Information Sciences 192 (2012) 174–192

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points

Akemi Gálvez, Andrés Iglesias *

Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros s/n, E-39005 Santander, Spain

ARTICLE INFO

Article history:
Available online 13 November 2010

Keywords:
Surface reconstruction
Reverse engineering
Surface parameterization
Surface fitting
Particle swarm optimization
NURBS surface

ABSTRACT

This work investigates the use of particle swarm optimization (PSO) to recover the shape of a surface from clouds of (either organized or scattered) noisy 3D data points, a challenging problem that appears recurrently in a wide range of applications such as CAD design, data visualization, virtual reality, medical imaging and movie industries. In this paper, we apply a PSO approach in order to reconstruct a non-uniform rational B-spline (NURBS) surface of a certain order from a given set of 3D data points. In this case, surface reconstruction consists of two main tasks: (1) surface parameterization and (2) surface fitting. Both tasks are critical but also troublesome, leading to a high-dimensional non-linear optimization problem. Our method allows us to obtain all relevant surface data (i.e., parametric values of data points, knot vectors, control points and their weights) in a shot and no pre-/post-processing is required. Furthermore, it yields very good results even in presence of problematic features, such as multi-branches, high-genus or self-intersections. Seven examples including open, semiclosed, closed, zero-genus, high-genus surfaces and real-world scanned objects, described in free-form, parametric and implicit forms illustrate the good performance of our approach and its superiority over previous approaches in terms of accuracy and generality.

1. Introduction

1.1. Swarm intelligence

Swarm intelligence (SI) has been defined as *the property of a system whereby the collective behaviors of (unsophisticated) agents or boids interacting locally with one another and with their environment cause coherent functional global patterns to emerge* [19]. A trademark of SI is the development of a collective behavior arising from decentralized systems comprised of (generally mobile) agents which communicate with each other (either directly or indirectly). Instead of a central behavior determining the evolution of the population, these local interactions between agents lead to the emergence of a global behavior for the swarm. Agents in a SI system obey very simple rules, have limited perception or intelligence and cannot individually carry out the task it intends to. A typical example of SI is the behavior of a flock of birds when moving all together following a common tendency in their displacements. Other examples from nature include ant colonies, animal herding, fish schooling and many others.

Nowadays, swarm intelligence is attracting increasing attention from researchers and practitioners because of its potential applications in several fields [59,76,78]. For instance, self-organizing swarm robots can potentially accomplish complex

* Corresponding author.
E-mail address: iglesias@unican.es (A. Iglesias).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.11.007

tasks and thus replace sophisticated and expensive robots by simple inexpensive drones [70], a research subfield usually referred to as *swarm robotics*. Military and civil applications of SI have also been reported in the literature with regards to the control of unmanned vehicles [71]. Applications of swarm intelligence range from crowd simulation in computer movies and video games to ant-based routing in telecommunication networks, and new exciting areas of research are constantly arising [10,32,49,68].

The objective of SI schemes is to model the simple behaviors of individual agents as well as their interactions with both the environment and their own neighbors in order to obtain more sophisticated behaviors that can be applied to solving complex problems, for instance optimization problems [53,83]. In fact, a very popular SI technique called particle swarm optimization (PSO) has been proposed to solve optimization problems and find solutions in a given problem space [43,44]. Basically, PSO is a global stochastic optimization algorithm for dealing with problems where potential solutions can be represented as vectors in a N-dimensional space (the *search space*). In PSO, particles representing potential solutions are distributed over such space and provided with an initial velocity and the capacity to communicate with other neighbor particles, even the entire swarm. Particles "flow" through the solution space and are evaluated according to some fitness function after each instance. Particles evolution is regulated by two memory factors: their memory of their own best position and knowledge of the global or their neighborhood's best. Particles of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. As the swarm iterates, the fitness of the global best solution improves so the swarm eventually reaches the best solution. Original PSO algorithm was first reported in 1995 by Kennedy and Eberhart in [43] (see also [15]). The reader is referred to the excellent book in [44]. See also [19] for a gentle analysis of PSO from a computational point of view and [1,11,14,75,81,82] for a comprehensive overview about some recent trends and applications of this technique.

1.2. Surface reconstruction

The problem of recovering the 3D shape of a surface, also known as *surface reconstruction*, has been a hot topic of research for the last 20 years or so, with several methods reported in the literature (see Section 2 for details). Given partial information about an unknown surface U, the goal of such methods is to construct, to the extent possible, a compact representation of a surface model S that approximates U. Depending on the available information about the surface (2D sections, clouds of points, grid of curves, interactive surface sketching, etc.), qualitatively different approaches can be formulated in order to tackle this issue. For instance, algorithms based on 2D cross-sections make use of the fact that data are organized into contours which lie, at their turn, in parallel planes. Other methods use a grid of isoparametric curves to infer the geometry and topology of the underlying surface.

In contrast, in this paper we only assume that the initial input is a (possibly massive) set of (noisy) 3D data points. This is a typical output obtained from some sorts of scanner devices, as commonly done in many reverse engineering applications. Our aim is to reconstruct the surface such data points come from by using a NURBS surface as fitting surface (where NURBS stands for non-uniform rational B-spline, by far the most common surface representation in real world applications). In this case, surface reconstruction comprises two main stages: (1) surface parameterization and (2) surface fitting. Both are critical in order to recover surface's geometry and topology and to obtain a good fitting to data points. A proper parameterization is essential in order to determine the relationships among neighbor points in the surface parametric domain, and hence surface's topology and boundaries. Furthermore, suitable surface fitting requires a good parameterization of data points. But as soon as the parameter values of data points are considered as variables, surface reconstruction becomes a complicated high-dimensional non-linear optimization problem. On the other hand, even with a good parameterization, there is no guarantee that the resulting fitting surface will be smooth enough. For instance, improperly chosen knot vectors might lead to a poor matching of data points. And it also remains the problem of obtaining the weights of control points.

1.3. Main contributions of this paper

In this paper, we address the above-mentioned problems by using a particle swarm optimization approach. In particular, given a set of 3D data points assumed to lie on an unknown NURBS surface of a certain order, PSO is applied to determine all relevant surface data (i.e., parametric values of data points, knot vectors, control points and their weights) thus actually returning the reconstructed NURBS surface. Main features of our method are:

- *Generality:* Our method is very general, since its input data consists exclusively of clouds of 3D points. We do not impose any constraint on the original surface those points come from. The method performs well regardless the points come from surfaces given in explicit, implicit or parametric form, or if they are polynomial, rational or belong to any other family of functions. Furthermore, as shown in Examples 5 and 6, surfaces can be of any genus, not necessarily of genus zero as it happens in many surface reconstruction methods.
- *Global method:* Most of surface reconstruction approaches require the combination of several methods or techniques in order to fully determine suitable values for all variables of the problem. In contrast, this method is global in the sense that it returns all data necessary to recover the surface without the addition of extra methods or further pre-/post-processing.

- *Good performance:* To test our method, we considered several point clouds from known surfaces of different types ranging from free-form Bézier polynomial surfaces to parametric surfaces or implicit surfaces of non-zero genus from which different point clouds are obtained. Notably, our method has been able to replicate the original surfaces (written in terms of NURBS surfaces) in all cases. For instance, for the cloud of points in Example 1 our method returns a NURBS surface with all unit weights (i.e., a polynomial B-spline) and non-periodic knot vectors with no interior knots, meaning that such a B-spline is actually a Bézier surface. This is a very remarkable result, since no data from the original surface (other than the data points) are used in our method.
- *Accuracy:* In the reported examples, the mean error for the data points is about 10^{-3} for the worst cases, but generally as low as 10^{-8}–10^{-15}. In other words, this method outperforms the previous ones in terms of numerical accuracy for the data points. Even more, our method is able to return the original surface with extremely high accuracy. Thus, for Example 1 our method determines the best order for the fitting NURBS surface, and returns all weights exactly equal to one, while the error in the control points is about 10^{-14}. In other words, we recover the original surface at full extent. Mean errors for data points coordinates are also of order 10^{-13}–10^{-15} in Examples 4–6, which correspond to complex-shaped, self-intersecting, high-genus surfaces. To our knowledge, no previous method is able to predict the shape of a surface at such extent from merely a cloud of points on it.
- *Robustness against noise:* Since the method is based on approximating the data points rather than interpolating them, it is robust against noise in data points. To test this issue, we introduced small random perturbations on some samples extracted from data points. As expected, global results were very similar, meaning that the method is not globally affected by the presence of isolated outliers below a prescribed threshold.

1.4. Structure of this paper

The structure of this paper is as follows: firstly, previous literature regarding surface reconstruction methods (including those based on evolutionary approaches) is reported in Section 2. Then, some basic concepts about NURBS surfaces and a gentle overview on the global PSO method are given in Sections 3 and 4, respectively. The core of the paper is in Section 5: it begins with a detailed explanation about the fitting surface problem. Then, the determination of relevant parameters for that problem via PSO along with the numerical procedures used in the process are discussed. Section 6 reports our experimental results. Some implementation issues, a detailed comparison with previous approaches and the analysis of the impact of the social and cognitive factors on our results are also given in that section. The paper closes with the main conclusions and our plans for future work.

2. Previous work

Surface reconstruction has been a topic of increasing attention 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 [12,23], statistics [13], numerical analysis [22,62], geometric modeling [24,79] and computer-aided geometric design (CAGD) [39,58]. On the applications side, surface reconstruction appears recurrently in a wide range of applications such as computer-aided manufacturing (CAM) [57,63], data visualization [64], cultural heritage preservation [48], virtual reality [47], medical imaging [2] and computer animation [67], to mention just a few.

In general, surface reconstruction methods are classified in terms of the available input (2D slices, iso-parametric curves, clouds of points, mixed information, etc.). For instance, authors in [3,4,38,54,56] address the problem of obtaining a surface model from a set of given cross-sections. This is a classical problem in medical science, biomedical engineering and CAD/CAM in which a volumetric object (such as a body inner organ, for instance) is typically defined by a sequence of 2D cross-sections or thin layers (acquired from computer tomography, magnetic resonance imaging, ultrasound imaging, etc.). Other classical input data include iso-parametric curves on the surface [28] and even mixed information, such as scattered points and contours [52,72] or iso-parametric curves and data points [39,40].

In most cases, however, available information about the surface is typically a dense set of (either organized or scattered) 3D data points obtained by using some sorts of scanner devices, the most common data point acquisition technique in reverse engineering applications (see, e.g., [30,31,35,36,50,51]). In such a case, the reconstructed surface can be described using three different representations providing different levels of accuracy, ranging from the coarsest (polygonal meshes) to the finest (NURBS surfaces):

- *Polygonal mesh:* In this model, data points are used as vertices connected by lines (edges) that work together to create a 3D model, comprised of vertices, edges and faces. This representation is the first choice for many computer graphics tasks, since it is very flexible and quicker to render and well suited for dealing with current graphical cards. However, *polygonal meshes are never a truly faithful representation of a smooth surface*. They are merely linear approximations of curved shapes; as such, they only provide a coarse representation of real objects: in a polygonal mesh, curves are approximated by linear segments, while surfaces are approximated by triangular or quadrilateral flat polygons. In this sense, polygonal meshes provide the lowest degree of accuracy, being mostly used for coarse geometry and fast rendering. Surface reconstruction methods with polygonal meshes can be found, for instance, in [20,35,36,48,55,64] and references therein.

- *CSG model:* In this model, 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, difference. Although often constructive solid geometry (CSG) presents a model or surface that appears visually complex, it is actually little more than a clever combination of simple objects by means of rather simple operations. As such, it is also limited in terms of the kind of objects it can represent: multi-branched, self-intersecting or high-genus objects are very hard (if not impossible) to be constructed with this technology. CSG models are barely applied to surface reconstruction because their extreme simplicity is a limiting factor in order to recover non-trivial shapes.
- *Free-form parametric surfaces:* They provide the most sophisticated and most accurate representation of smooth real-world objects. Among them, NURBS surfaces are the most powerful and most common free-form parametric surfaces because of their flexibility, versatility, and the fact that they represent well a wide variety of shapes in a very compact and intuitive mathematical form. This is the preferred form for detailed surfaces, since they faithfully represent real objects: using NURBS, a sphere is a true mathematical sphere, not a collection of flat polygons resembling a sphere.

Since our primary goal in this paper is about quality and accuracy, NURBS surfaces are clearly our ideal choice for surface reconstruction. In this context, our problem can be stated as follows: *given a set of sample points* **Q** *assumed to lie on an unknown surface* **U***, construct, to the extent possible, a NURBS representation of a surface model* **S** *that approximates* **U**. Depending on the nature of these data points, two different approaches are employed: interpolation and approximation. Approximation techniques are specially recommended when data are not exact, but subjected to measurement errors. Another important reason to choose approximation could be the great computational effort required to obtain surfaces by interpolating an infinite number of data, such as curve forms. An example is given in surface skinning problems, where we look for a smooth surface passing through a set of cross-sectional curves. In [60] it has been mentioned that, using the NURBS representation, interpolation of only a few cross-sections of various types may require as high a number as hundreds of thousands of surface control points. Furthermore, because industrial parts can easily contain hundreds of surfaces, interpolation for these parts becomes unrealizable in practice. Finally, in many applications, the data consist of a very large number of measurements, causing the number of basis functions to be very large as well. In addition, new measurement points may change the structure of the solution.

The obvious solution to these problems is to consider an approximation scheme that allows us significant saving of time and memory. This problem has been analyzed from several points of view, such as parametric methods [7], function reconstruction [21,73], implicit surfaces [50], artificial immune systems [77], genetic algorithms [25], etc. Artificial neural networks have also been applied [30,34], mostly for arranging the input data in case of unorganized points. After this pre-processing 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 [5]. Other sophisticated techniques include functional networks [16,40], a generalization of neural networks based on functional equations [8,9].

Reconstruction of freeform objects consists typically of two main stages: (1) surface parameterization and (2) surface fitting. Two basic approaches have been considered to solve these stages [79,86]: the first one uses an iterative process in which the region is approximated by an initial surface which is then smoothed iteratively [37,41,51]. The initial surface, usually called *base surface*, is chosen so that it reflects the distribution of points in the 3D space. Potential base surfaces are planes, cylinders or Coons patches. Sample points are projected onto this base surface in order to extract boundary features as well as a coarse 2D parameterization of the surface. These approaches fail in case data points cannot be projected in an unambiguous way. An alternative is to consider a 3D triangulation and then derive a topologically equivalent 2D triangulation [17,20]. These methods are, however, slow and computational expensive for large numbers of data points and find difficulties when the point cloud contains concavities or holes. The second approach uses a variational scheme that seeks to minimize a hybrid functional accounting simultaneously for good approximation and smoothness [17,29,33]. They tend to fail in case of uneven distributions of points over the region of interest, and present the problem that additional coefficients (such as the smoothness weighted factor) must be determined. Another problem is the correct determination of the length of knot vectors, as it affects the surface quality and the computational efficiency of the process. In general, a trade-off must be achieved between increasing the number of knots to increase the degrees of freedom and to reduce such a number in order to alleviate the computation time. All these methods also tend to fail for surfaces of non-zero genus. Our proposed method overcomes such limitations, as we will show later on.

2.1. Evolutionary approaches for surface reconstruction

Regarding evolutionary approaches for surface reconstruction, there are some references in the case of polygonal meshes using evolutionary algorithms (EA) [27,66], simulated annealing (SA) [74], an hybrid scheme with genetic algorithms (GA) and SA [45] and a combination of EA, SA and GA [84]. Those methods do not provide a high degree of accuracy and are limited to simple geometries. For instance, the approach in [84] reconstructs objects in 2D search space, making it difficult to be applied to real data. The same comment applies to [45].

On the contrary, there are very few works on surface reconstruction by using NURBS surfaces. This is not a surprising fact: NURBS surfaces are particularly challenging for evolutionary techniques, since they are based on non-linear basis functions involving a large number of parameters, so the problem becomes a high-dimensional non-linear problem. Among those rare

papers for surface reconstruction with NURBS by using evolutionary approaches, we mention [6,80,85]. In [85] authors propose a combination of CSG models and NURBS to overcome the limitation of CSG observed in previous works [42]. Yet, the reconstructed surfaces are still very simple in comparison with ours, errors are much higher and so are the reported computation times (see Section 6.3 for comparative details). The paper in [6] focuses on the design of evolutionary algorithms for surface reconstruction and has been extended in [80], but only very simple models are reconstructed even although the method does not compute the parametric values of data points (and hence only height map models can be handled). In fact, these methods make use of the concept of base surface discussed above (where data points are projected onto a plane along the vertical axis) and hence are affected by strong limitations (for instance, surfaces with concavities or holes cannot be reconstructed). In short, no evolutionary approach can be applied so far to reconstruct a general NURBS surface; they only work properly for very particular (and geometrically almost trivial) cases. This fact is not accidental, but a clear indication of the extreme difficulty to dealt with NURBS surfaces in the framework of evolutionary approaches.

3. NURBS surfaces

Let $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_{r-1}, s_r\}$ be a non-decreasing sequence of real numbers called *knots*. $\mathcal{S}$ is called the *knot vector*. The *ith B-spline basis function* $N_{i,k}(u)$ *of order* k (or equivalently, degree $k-1$) is defined by the recurrence relations

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

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 can be either an interval or reduce to a point, as knots s_i and s_{i+1} must not necessarily be different. If necessary, the convention $\frac{0}{0} = 0$ in Eq. (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 polinomials 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.

The number of times a knot appears in the knot vector is called the *multiplicity* of the knot and has an important effect on the shape and properties of the associated basis functions. Basically, knot vectors can be classified into two groups. The first one is the *uniform knot vector*: each knot appears only once and the distance between consecutive knots is always the same. As a consequence, each basis function $N_{i,k}(u)$ is plotted similarly to the previous one, $N_{i-1,k}(u)$, but shifted to the right according to such a distance. A qualitatively different behavior is obtained when any of the knots appears more than once (this case is usually referred to as *non-uniform knot vector*). The most common case of non-uniform knot vectors consists of repeating the end knots as many times as the order while interior knots appear only once (such a knot vector is called *non-periodic knot vector*).

With the same notation, given a set of three-dimensional points (called *control points* as they roughly determine the shape of the surface) $\{\mathbf{P}_{ij}\}_{i=0,\ldots,m;j=0,\ldots,n}$ (note that in this paper vectors are denoted in bold) 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) is a rational B-spline parametric surface given by:

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

where the $\{N_{i,k}(u)\}_i$ and $\{N_{j,l}(v)\}_j$ are the B-spline basis functions of order k and l, respectively, defined following (1) and (2), and $\{w_{i,j}\}_{i,j}$ are weights associated with the control points $\mathbf{P} = \{\mathbf{P}_{ij}\}_{i,j}$. Introducing now the piecewise rational basis functions:

$$R_{ij}(u, v) = \frac{w_{ij}N_{i,k}(u)N_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}N_{i,k}(u)N_{j,l}(v)}, \quad i = 0, \ldots, m;\ j = 0, \ldots, n \tag{4}$$

NURBS suface in (3) can be rewritten as:

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

Without loss of generality parameters u, v can be assumed to take values on the interval $[0, 1]$ [58]. For a proper definition of a NURBS surface in Eq. (3), the following relationships must hold (see [58]): $r = m + k$, $h = n + l$. In general, a B-spline surface does not interpolate any of its control points; interpolation only occurs for non-periodic knot vectors (in that case, the B-spline surface does interpolate the corner control points). Since they are the most common in computer graphics and industrial domains, in this work we will restrict ourselves to non-periodic knot vectors. Note however that our method does not preclude any other kind of knot vectors to be used at all.

4. Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic algorithm based on the evolution of populations for problem solving. In PSO the particle swarm simulates the social optimization commonly found in communities with a high degree of organization. For a given problem, some fitness function is needed to evaluate the proposed solution. In order to get a good one, PSO methods incorporate both a global tendency for the movement of the set of individuals and local influences from neighbors [43]. PSO procedures start by choosing a population (swarm) of random candidate solutions in a multidimensional space, called *particles*. Then they are displaced throughout their domain looking for an optimum taking into account global and local influences, the latest coming form the neighborhood of each particle. To this purpose, all particles have a position and a velocity and evolve all through the hyperspace according to two essential reasoning capabilities: a memory of their own best position and knowledge of the global or their neighborhood's best. The meaning of the "best" must be understood in the context of the problem to be solved. In a minimization problem (like in this paper) that means the position with the smallest value for the target function.

The dynamics of the particle swarm is considered along successive iterations, like time instances. Each particle modifies its position P_i along the iterations, keeping track of its best position in the variables domain implied in the problem. This is made by storing for each particle the coordinates P_i^b associated with the best solution (fitness) it has achieved so far along with the corresponding fitness value, f_i^b. These values account for the *memory* of the best particle position. In addition, members of a swarm can communicate good positions to each other, so they can adjust their own position and velocity according to this information. To this purpose, we also collect the best fitness value among all the particles in the population, f_g^b, and its position P_g^b from the initial iteration. This is a global information for modifying the position of each particle. Finally, the evolution for each particle i is given by:

$$V_i(k+1) = wV_i(k) + \gamma_1 R_1[P_g^b(k) - P_i(k)] + \gamma_2 R_2[P_i^b(k) - P_i(k)] \tag{6}$$

$$P_i(k+1) = P_i(k) + V_i(k) \tag{7}$$

where $P_i(k)$ and $V_i(k)$ are the position and the velocity of particle i at time k, respectively, w is called *inertia weight* and decide how much the old velocity will affect the new one and coefficients γ_1 and γ_2 are constant values called *learning factors*, which decide the degree of affection of P_g^b and P_i^b. In particular, γ_1 is a weight that accounts for the "social" component, while γ_2 represents the "cognitive" component, accounting for the memory of an individual particle along the time. Two random numbers, R_1 and R_2, with uniform distribution on $[0,1]$ are included to enrich the searching space. Finally, a fitness function must be given to evaluate the quality of a position. This procedure is repeated several times (thus yielding successive generations) until a termination condition is reached. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. Global PSO procedure is briefly sketched in Table 1.

The driving force of PSO is social interaction: particles learn from each other and evolve in order to become more similar to their "better" neighbors. This evolution is strongly related to the social structure of the swarm, usually referred to as the *communication topology*. Highly connected social networks in which most individuals communicate with each other have the advantage that good information quickly disseminates throughout the swarm. This means faster convergence to a solution than in less connected networks. At their turn, sparsely connected organizations are less susceptible to be trapped in local minima. Consequently, a trade-off between fast convergence and susceptibility to local minima must be achieved. As a result, several communication topologies have been developed. The first implementation of PSO was based on the *star* social structure, where all particles are connected with each other. In such a case, each particle is attracted towards the best solution found by the entire swarm. The resulting algorithm is described by Eqs. (6) and (7) and is usually called global best (*gbest*). Other social structures such as ring, wheel, pyramid, four clusters, von Neumann and so on are also possible. They lead to local best (*lbest*) models where the best fitness value and position of the swarm must be replaced by those of the neighborhood (and therefore Eqs. (6) and (7) must be modified accordingly). In this work we chose the star communication topology, since target value is known in advance, so we can check if we are trapped in local minima (something that did not happen in our experiments). As shown in Section 6, this *gbest* strategy yielded very good solutions for all our experiments. A careful analysis of other communication topology strategies in the context of surface reconstruction is part of our future work.

5. Our method

5.1. Surface fitting problem

Let us suppose that we want to fit a given set of three-dimensional data points arranged in a matrix as: $\mathbf{Q} = \{\mathbf{Q}_{\alpha,\beta} = (x_{\alpha,\beta}, y_{\alpha,\beta}, z_{\alpha,\beta})\}_{\alpha=0,\ldots,p;\beta=0,\ldots,q}$. We want now to fit such points with a NURBS surface $\mathbf{S}(u,v)$ of order (k,l) defined as above (where $r < p$, $h < q$) such that:

$$\mathbf{Q}_{\alpha,\beta} = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}R_{i,j}(u_\alpha, v_\beta) \quad \forall \alpha = 0,\ldots,p;\ \beta = 0,\ldots,q \tag{8}$$

Table 1
General structure of the particle swarm optimization algorithm.

```
Create a swarm of population Pop
begin
  k = 0
  initialization of individual positions P_i and velocities V_i in Pop(k)
  fitness evaluation of Pop(k)
  while (not termination condition) do
    Calculate best fitness particle P_g^b
    for each particle i in Pop(k) do
      Calculate particle position P_i^b with best fitness
      Calculate velocity V_i for particle i according to (6)
      while not feasible P_i + V_i do
        Apply scale factor to V_i
      end
      Update position P_i according to (7)
    end
    k = k + 1
  end
end
```

To do so, we must first make an association (parameterization) of parameter values (u_α, v_β) to each of our data points $\mathbf{Q}_{\alpha,\beta}$. Given this association and an adequate choice of weights $w_{i,j}$, surface fitting can be formulated from (4) and (5) as finding the surface which minimizes the following weighted least-squares expression:

$$E_{lsq} = \sum_{\alpha=0}^{p}\sum_{\beta=0}^{q}(\mathbf{Q}_{\alpha,\beta} - \mathbf{S}(u_\alpha, v_\beta))^2 = \sum_{\alpha=0}^{p}\sum_{\beta=0}^{q}\left(\mathbf{Q}_{\alpha,\beta} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}R_{i,j}(u_\alpha, v_\beta)\right)^2 \tag{9}$$

The case of unorganized points can be handled in a quite similar way by replacing the matrix by a vector. In other words, data points are now arranged along a vector $\{\mathbf{Q}_\mu = (x_\mu, y_\mu, z_\mu)\}_{\mu=0,\ldots,D}$, where D is the number of data points. In this case, we must associate suitable parameter values (u_μ, v_μ) and weights w_μ to each of our data points $\mathbf{Q}_\mu$ so the surface fitting problem is expressed as:

$$E_{lsq} = \sum_{\mu=0}^{D}(\mathbf{Q}_\mu - \mathbf{S}(u_\mu, v_\mu))^2 = \sum_{\mu=0}^{D}\left(\mathbf{Q}_\mu - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}w_\mu\mathbf{P}_\mu N_{i,k}(u_\mu)N_{j,l}(v_\mu)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_\mu N_{i,k}(u_\mu)N_{j,l}(v_\mu)}\right)^2 \tag{10}$$

As a result, an over-constrained system of linear equations can be generated from either (9) or (10) where $\mathbf{P}_{i,j}$ or $\mathbf{P}_\mu$ are the unknowns, respectively. Therefore, a least-squares solution can be performed in order to compute such control points. The problem is that in real applications, we are given neither parameter associations with data points nor the weights. In fact, even the number of control points and the number of knots are also unknown. Because the B-spline basis functions are non-linear functions, the problem requires to solve a non-linear optimization process, with a high number of unknowns for large sets of data points, a case that happens very often in practice.

Expression (8) can be rewritten as:

$$vec(\mathbf{Q}) = \mathbf{M}\cdot vec(\mathbf{P}) \tag{11}$$

where $vec(\cdot)$ is the vectorization operator (the linear transformation that converts the matrix into a column vector by stacking its columns on top of one another), and $\mathbf{M} = \{[[\mathbf{R}(u_\alpha, v_\beta)]], \alpha = 0,\ldots,p;\ \beta = 0,\ldots,q\}$ for $\mathbf{R}(\tilde{u}, \tilde{v}) = \{[[R_{i,j}(\tilde{u}, \tilde{v})]],\ i = 0,\ldots,m;\ j = 0,\ldots,n\}$, where $[[\mathbf{A}]] = vec(\mathbf{A}^T)$ for any matrix $\mathbf{A}$ and $(\cdot)^T$ means the transpose of the matrix. However, $vec(\mathbf{Q})$ is a vector of length $(p+1)\times(q+1)$ while $vec(\mathbf{P})$ has length $(m+1)\times(n+1)$ so system (11) is overdetermined. Pre-multiplication of both sides by $\mathbf{M}^T$ yields:

$$\mathbf{M}^T\cdot vec(\mathbf{Q}) = \mathbf{M}^T\cdot\mathbf{M}\cdot vec(\mathbf{P}) \tag{12}$$

By this way, the problem becomes a classical linear least-squares minimization. Computation of coefficients $\mathbf{P}_{i,j}$ will give the best fit in the discrete least-squares sense to the data points, thus actually reconstructing the fitting NURBS surface.

As a conclusion, obtaining a feasible solution of the surface fitting problem typically requires to perform four important tasks:

(1) Firstly, a careful choice of suitable parameters is needed in order to construct the B-spline basis functions in Eq. (3). Such parameters include the order of B-spline basis functions, the number of control points and the knot vectors.
(2) Secondly, weights associated with the control points must be obtained.
(3) Then, a proper parameterization of data points must be achieved.
(4) Finally, the coefficients of the least-squares problem in Eq. (12) are to be determined.

As show in next paragraphs, we solve these four sub-problems by applying a PSO procedure in order to determine all relevant parameters required from previous discussion. Our approach clearly shows that PSO is powerful enough to formulate all these related sub-problems in an unified and global way so no further problem subdivision is actually needed.

5.2. Parametric determination with PSO

To solve the previous problem, the *gbest* PSO algorithm described above has been applied. The initial input is given by:

- the collection of 3D data points, $\mathbf{Q}$,
- the order of the NURBS surface, (k,l), and
- the number of control points, (m,n).

The search space is a hyperspace of dimension:

$$4(m+1)(n+1)+(m-k+n-l+2)+2(p+1)(q+1) \tag{13}$$

where figures in this formula are obtained as follows:

- There are $(m+1)(n+1)$ control points to be calculated, each having three coordinates. Besides, each control point has its corresponding scalar weight, totaling $4(m+1)(n+1)$ data.
- There are two knot vectors of dimensions $m+k+1$ and $n+l+1$, respectively. This means $m+k+n+l+2$. But since in this paper we consider non-periodic knot vectors exclusively, $2k$ and $2l$ elements are already fixed in knot vectors, respectively, so total length is $m-k+n-l+2$.
- Finally, we have $(p+1)(q+1)$ data points, each with two parameter values (u,v), thus making $2(p+1)(q+1)$.

PSO parameters used in this paper are as follows: unless otherwise stated, an initial population size of 1000 individuals or particles, has been considered. All particles have been initialized with random uniform values on ranges according to the kind of data to be computed. For instance, inner knots take values on the interval $(0,1)$ while weights are valued on the interval $(0,2)$. The order of the fitting surface is chosen based on the visual complexity of data points; simple problems are assigned values between 2 and 5–8, and this value is increased for intrincated shapes. However, NURBS are piecewise functions, so this choice is not critical at all. More relevant is the location of control points, which are randomly placed at initial stage. A similar procedure is considered for data point parameterization. A critical issue in this method is the choice of the coefficients γ_1 and γ_2 accounting for the global (or social) and the local (or cognitive) influences, respectively. They have been set to 2.0 in this work. However, our experiments revealed that for the problem addressed in this paper their choice is much less critical than expected; in fact, it has negligible influence in our results (see Section 6.4 for details). Inertia coefficient was initially set to $w=1$ although several studies support that it is much better to change it dynamically, starting from a high value (let us say, $w=0.9$), which corresponds to a system where particles move in a low viscosity medium and perform extensive exploration, and gradually reducing it to a much lower value (e.g., $w=0.4$) where the system would be more dissipative and exploitative and would be better at homing into local optima [61]. We eventually turned to such approach, and results indeed improved reasonably both in accuracy and runtime. Finally, our termination criterion is that of not improving the solution after 10 consecutive iterations.

5.3. Numerical procedures

Regarding the numerical procedures, three different methods for solving either (11) or (12) have been used in this paper: the standard LU decomposition method, a non-iterative least squares fitting method accomplished by the singular value decomposition (SVD) method [65], and a modification of the LU decomposition optimized for non-squared sparse problems. Method LU was primarily chosen because of its theoretical simplicity, being applied directly to Eq. (12) rather than to Eq. (11) in order to increase the computational efficient. By this way, it can be computed by several algorithms (Doolittle, Crout, etc.) at a cost of $O(2p^3/3)$ excluding pivoting. However, LU was early discarded because it is less efficient than other methods and returns no results in some cases. We then moved to the SVD method, mostly because of its robustness and because it provides the best numerical answer in the sense of least-squares for those cases in which the exact solution is not possible [65]. To this aim, matrix $\mathbf{M}$ in Eq. (11) is decomposed as the matrix product $\mathbf{M}=\mathbf{U}\cdot\mathbf{\Phi}\cdot\mathbf{V}^T$ where $\mathbf{U}$ is a column-orthogonal matrix, $\mathbf{\Phi}$ is a diagonal matrix with positive or zero elements ϕ_k called the singular values and $\mathbf{V}$ is a square orthogonal matrix. Similar decomposition can be performed on $\mathbf{M}^T\cdot\mathbf{M}$ in case Eq. (12) is alternatively used. Furthermore, since $\mathbf{M}^T\cdot\mathbf{M}$ is square, its inverse can readily be obtained as: $\mathbf{V}\cdot\left[diag\left(\frac{1}{\phi_k}\right)\right]\cdot\mathbf{U}^T$. The SVD can be computed by a two-step procedure with a computational cost of $O(p\cdot q^2)$ for a problem of size $p\times q$. In general, SVD performs well and yields very accurate results, but does not take advantage of matrix sparsity derived from the local support of B-spline basis functions. To overcome this, we implemented a modification of LU decomposition for non-squared sparse problems [65] that outperformed the previous methods in our experiments.

6. Experimental results

In this section we report our experimental results. Some implementation issues are also briefly outlined. Finally, a comparison with some previous approaches and the analysis of the impact of the social and cognitive factors on our results are discussed.

6.1. Illustrative examples

Our algorithm has been tested on several examples from different families of functions. In this section, we consider seven of them: a bicubic Bézier surface, a right circular cylinder, a sphere, a shell surface, two parametric surfaces of non-zero genus and a scanned real-world object. The examples have been primarily chosen to reflect the diversity of situations our algorithm can be applied to, and because they provide a potential benchmark for future experiments. Thus, you can find open, semiclosed and closed surfaces in Examples 1–3 (originally defined in free-form, parametric and implicit form, respectively), while Examples 4–6 correspond to surfaces with very challenging features, such as complicated shapes, self-intersections and high-genus and Example 7 corresponds to a real-world object.

6.1.1. Example 1: a bicubic Bézier surface

As a first example, we consider a set of 400 data points from a bicubic Bézier surface fitted by a NURBS surface with orders varying from 2 to 5 for both u and v and control points ranging from 3 to 10 in both directions. Amazingly, we have been able to recover the original surface at full extent: the best fitting errors are obtained with a NURBS surface of order $(4,4)$ with 4×4 control points and with all unit weights. The *gbest* error in this case is as good as 5.18×10^{-14} with 11 iterations for a population of 60 particles and runtime of 4.85 s. We also compared the net of control points for the original and the recovered fitting surface; mean errors for the (x,y,z) coordinates are as low as $(2.23, 3.21, 2.57) \times 10^{-15}$. Knots vectors in this case have also been recovered; they are both $[0,0,0,0,1,1,1,1]$, meaning that the best fitting NURBS surface for the original surface is actually a bicubic Bézier surface, expressed in terms of the rational B-spline basis functions, with the same control points, non-periodic knot vectors and all weights equal to *1*. This example shows the great performance of our approach.

Very good results have also been obtained for all cases of fitting NURBS surfaces with order $(4,4)$ and $n \times n$ control points, n from 4 to 10. For instance, Fig. 1 shows the fitting surface for the case $n = 6$ along with the data points. This result can be interpreted as these fitting surfaces replicate the process of performing degree elevation on the original surface, such that the surface keeps exactly the same shape but additional degrees of freedom can be generated at the price of increasing the polynomial order.

6.1.2. Example 2: a right circular cylinder

Although cylinders turn out to be one of the most basic geometric shapes, they are also challenging with regards to NURBS surfaces, since the cylinder is a closed surface in one parametric direction but open in the other one, so the choice of knot vectors is not trivial. Thus, a cylinder is a good candidate to test the performance of our method against strictly rational geometric shapes.

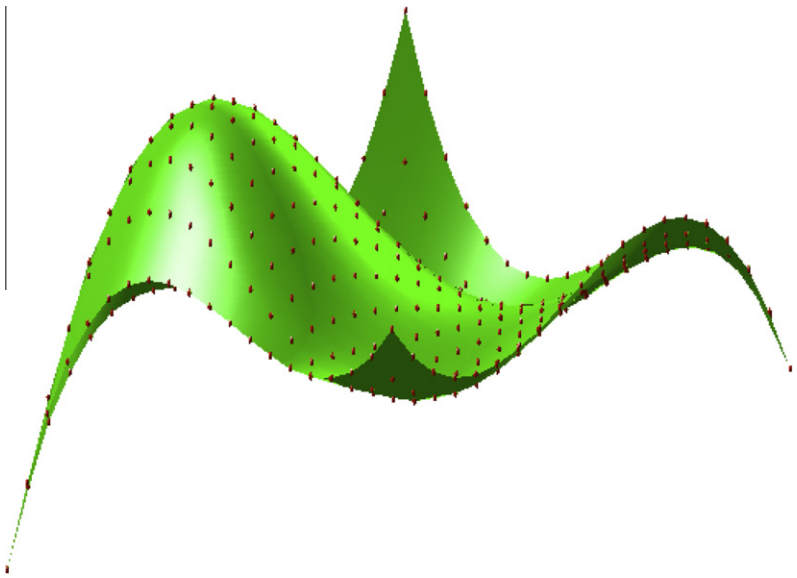

Fig. 1. Cloud of 3D data points and fitting NURBS surface.

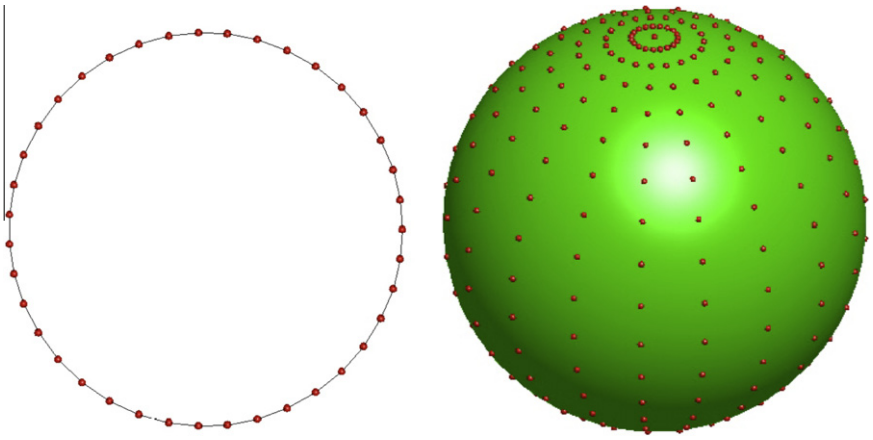

Fig. 2. Two examples of fitting surfaces: (left) upper view of a right circular cylinder; (right) a sphere.

We applied our method on a set of 2476 noisy data points (slightly perturbed with a small noise with uniform distribution to check the robustness of our reconstruction method) and a NURBS fitting surface with orders varying from 2 to 6. The number of control points also varies from 2 to 30 in both directions. The best fitting surface has been obtained for a (2,3)-order surface with a net of 2×24 control points, with a *gbest* error of 0.00037 and errors of the order of 10^{-3} in the data points, although other values for the order and number of control points also produced good errors. Note that order (2,3) means that the surface is linear in the vertical direction (actually a ruled surface) while it is a circle comprised of quadratic pieces in the horizontal direction. Fig. 2 (left) displays the surface as seen from the top, so that we can see the good fitting to point data along the most complicated direction (in fact, shape in vertical direction is linear and hence trivial).

6.1.3. Example 3: a sphere

Representing a sphere as a NURBS surface is not as easy as it seems to be at first glance. One way is revolving a NURBS curve about a vertical axis (thus treating the sphere as a revolution surface). Moreover, control points at the north and south poles must be repeated more than once and the knot vector for the NURBS curve is non-uniform, as opposed to our approach on focusing on non-periodic knot vectors exclusively. This means the sphere challenged our approach in several ways and for these reasons we include this example here. For our set of 420 data points generated from a sphere described implicitly, the best fitting NURBS surface is of (4,4) order with a net of 9×4 control points, with a *gbest* error of 0.003298, mean errors for the data points of the range of 10^{-3} and knot vectors $\mathcal{U} = [0,0,0,0.235,0.356,0.534,0.636,0.67,0.858,1,1,1]$ and $\mathcal{V} = [0,0,0,0,0.469,0.544,1,1,1,1]$. Fig. 2 (right) shows the fitting NURBS surface and the fitted data points.

6.1.4. Example 4: shell surface

Next example is a parametric surface known as *shell surface* and given by:

$$\begin{cases} x = \frac{1}{5}\left(1 - \frac{v}{2\pi}\right)cos(2v)(1 + cos(u)) + \frac{1}{10}cos(2v) \\ y = \frac{1}{5}\left(1 - \frac{v}{2\pi}\right)sin(2v)(1 + cos(u)) + \frac{1}{10}sin(2v) \quad u, v \in [0, 2\pi] \\ z = \frac{v}{2\pi} + \frac{1}{5}\left(1 - \frac{v}{2\pi}\right)sin(u) \end{cases}$$

Given a collection of 1395 data points, the best fitting surface obtained with our method is a (4,4)-order NURBS surface and 11×11 control points. The *gbest* error we got is 7.92×10^{-15} while mean errors for data points coordinates are 1.2×10^{-16}, 6.69×10^{-17} and 1.059×10^{-15}, respectively. The corresponding fitting surface is displayed in Fig. 3. Although this surface does neither exhibit self-intersections nor holes, it still has a complicated shape. In fact, we noticed that a number of surface reconstruction methods cannot reconstruct it accurately.

6.1.5. Example 5: Klein cycloid surface

In this example, we move further towards complexity by considering a multi-branched, high-genus, self-intersecting surface known as *Klein cycloid* surface, given in parametric form by:

$$\begin{cases} x = cos\left(\frac{u}{2}\right)cos\left(\frac{u}{3}\right)(10 + cos(v)) + sin\left(\frac{u}{3}\right)sin(v)cos(v) \\ y = sin\left(\frac{u}{2}\right)cos\left(\frac{u}{3}\right)(10 + cos(v)) + sin\left(\frac{u}{3}\right)sin(v)cos(v) \\ z = -sin\left(\frac{u}{3}\right)(10 + cos(v)) + cos\left(\frac{u}{3}\right)sin(v)cos(v) \end{cases}$$

where $u \in [0,6]$, $v \in [0,4\pi]$. We applied our method to a set of 9150 data points from this surface and, once again we obtained satisfactory results. The best fitting surface in this case was a (9,9)-order NURBS surface (two views of this surface are depicted in Fig. 4) with 21×21 control points, a *gbest* error of 2.30×10^{-10} and mean errors for data points coordinates of

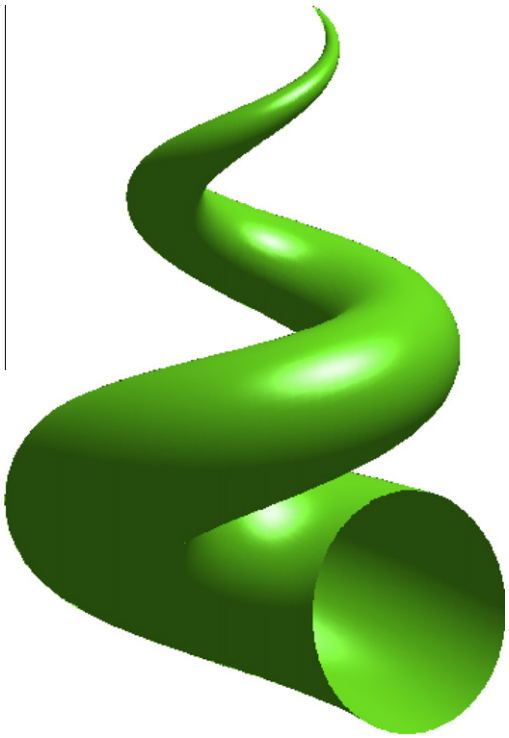

Fig. 3. Fitting NURBS surface of the shell surface.

2.21×10^{-12}, 3.05×10^{-13} and 1.50×10^{-14}, respectively. We remark the excellent results obtained by the method even in presence of self-intersections and the perfect fitting of different branches to data points matching the original surface extremely well.

6.1.6. Example 6: tranguloid trefoil surface

Last example corresponds to a very complicated shape aimed at illustrating the wide range of applicability of our method as well as its good performance. We consider a cloud of 4132 data points from a parametric surface known as *tranguloid trefoil*, described as:

$$\begin{cases} x = \frac{2sin(3u)}{2+cos(v)} \\ y = 2\frac{sin(u)+2sin(2u)}{2+cos\left(v+\frac{2\pi}{3}\right)} \\ z = \frac{1}{4}(cos(u) - 2cos(2u))(2 + cos(v))\left(2 + cos\left(v + \frac{2\pi}{3}\right)\right) \end{cases} \qquad -\pi \leqslant u, v \leqslant \pi$$

Best fitting surface is a (7,7)-order NURBS surface with 15×16 control points, *gbest* error 2.38×10^{-13} and mean error for data points coordinates $(2.06, 2.27, 0.405) \times 10^{-14}$ (although similar results have been obtained for different number of control points). Fig. 5 shows (from top to bottom) the initial cloud of data points, and front and upper views of the fitting NURBS surface. Note that data points are organized in slices as usually obtained from scanner devices. However, no raster preferential direction can actually be chosen since those slices are non-parallel.

6.1.7. Example 7: cell phone

Last example corresponds to a real-world object: a cell phone, scanned by Geomagic [26] and made publicly available years ago. Fig. 6 shows the bicubic NURBS surface fitting the cloud of 5400 data points, with *gbest* error 7.43×10^{-8} and mean error for data points $(3.35, 2.61, 3.89) \times 10^{-6}$. We remark that, in spite of the apparent visual simplicity of the model, it is actually complex from the geometrical standpoint. In fact, industrial IGES file of this model is comprised of six NURBS surfaces with 18×18 control points each, and very complicated non-uniform knot vectors.

6.2. Implementation issues

All computations in this paper have been performed on a 2.4 GHz Intel Core 2 Duo processor with 4 GB of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program

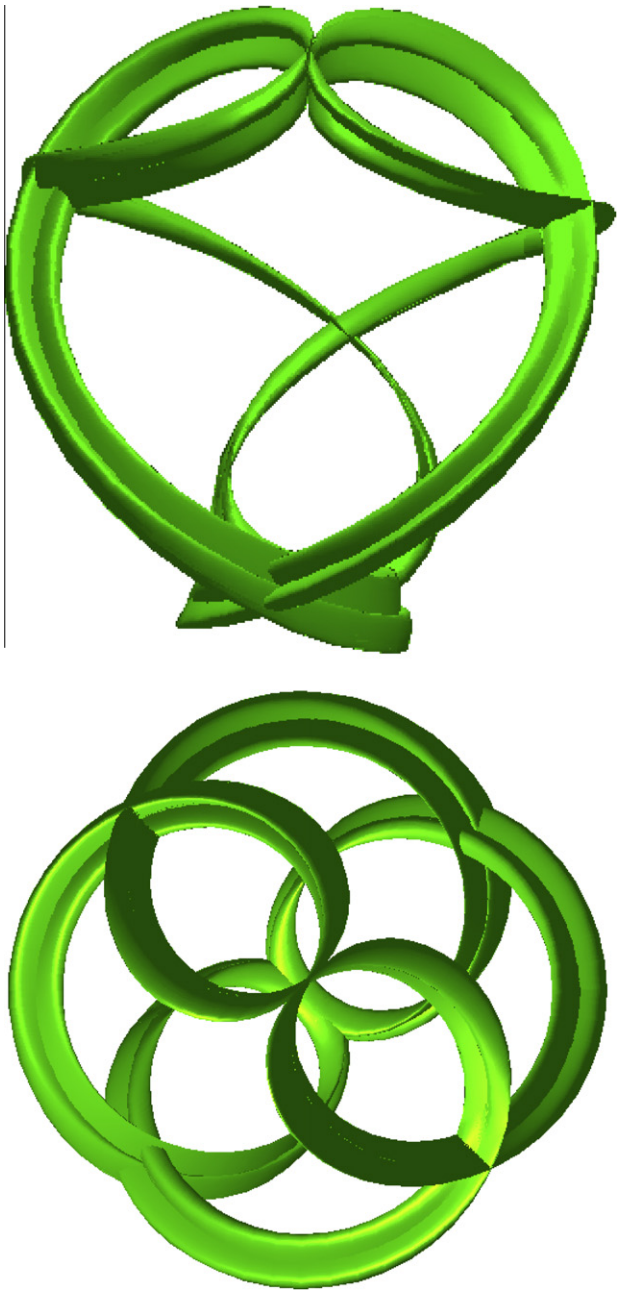

Fig. 4. Two views of a fitting NURBS surface of the Klein cycloid surface.

Matlab, version 7.0. In our opinion, Matlab is a very suitable tool for this task: it is fast and provides reliable, well-tested routines for efficient matrix manipulations. It also contains a bulk of resources regarding the solving of systems of equations. Although we decided to implement such numerical routines by ourselves, this feature proved to be valuable in order to test our early results and for comparison purposes at later stages of this work. Besides, Matlab provides excellent graphical options and optimized code for input/output interaction and high performance computations.

An important issue concerns the computation time. First observation is that our approach is not well suited for real-time applications; the emphasis is on accuracy. In our examples, each execution takes from some tens of seconds to several minutes, depending on the complexity of the surface (existence of holes, concavities, self-intersections, several branches), the number of initial data points, the initial population of the swarm and the numerical procedure employed. Of all those values, the most critical one for runtime is the population size. In our experiments, we changed this value from 60 to 1000, and corresponding runtimes become much longer as the population size increased. As a general rule, we talk about tens of seconds

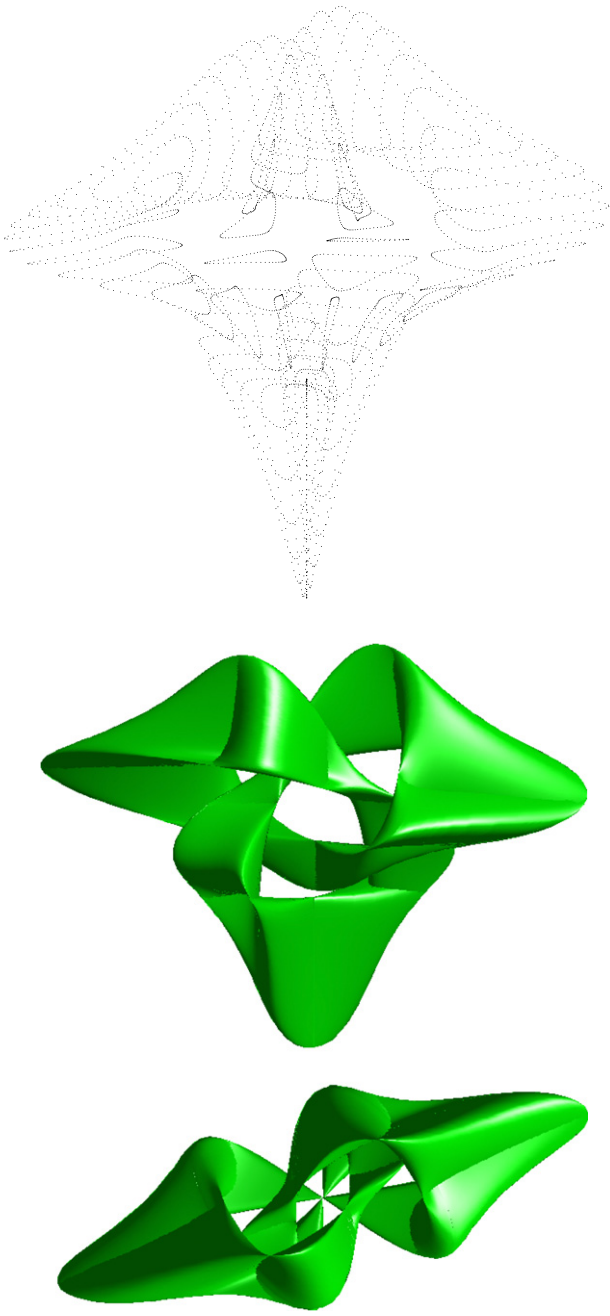

Fig. 5. Fitting the tranguloid trefoil surface: (top) cloud of data points; (middle) fitting NURBS surface: front view; (bottom) fitting NURBS surface: upper view.

for swarms of 60 particles and minutes for swarms of 1000 individuals. On the contrary, some PSO parameters such as the values of (γ_1, γ_2) do not influence the execution times significantly. Other parameters (such as the order of the NURBS surface and the number of control points and knots) and the intrinsic stochastic nature of the method also influence the computation time, making it hard to obtain a general rule for determining *a priori* how long does it take for a given example to be solved. Further analysis on this question is part of our future work.

6.3. Comparison with other approaches

As stated, our PSO approach performs very well for the problems analyzed above and others not included here because of limitations of space. Compared with other approaches reported in the literature, our method outperforms them in terms of

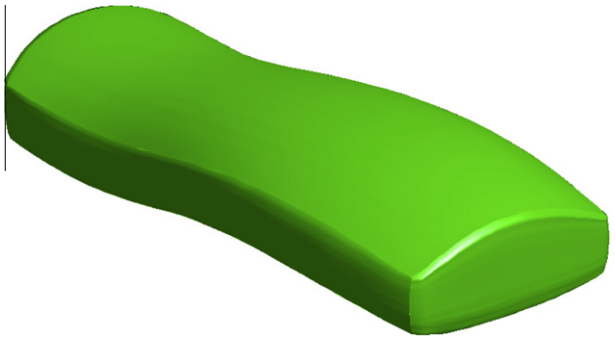

Fig. 6. Fitting NURBS surface of a cell phone model.

accuracy and flexibility. To support this claim, a careful comparison with recent alternative surface reconstruction methods that use smooth piecewise surfaces and are based on evolutionary techniques has been carried out. This gives two methods [80,85]. A powerful method for piecewise surfaces close to ours but not based on evolutionary techniques has been included to enrich the discussion [5]. Methods considering polygonal meshes fall out of the scope of this paper and therefore should not be considered for comparison. However, for illustrative purposes we decided to include a recent method for polygonal meshes [27].

Table 2 summarizes our main results. Compared methods are arranged in rows and sorted so that methods not based on evolutionary algorithms appear first [5]; then, evolutionary-based methods for polygonal meshes [27] and piecewise surfaces [80,85] follow; finally, our method is reported. For each method, the following issues are discussed: technique employed in the method, number of data points in examples reported in the corresponding entries, population size (when applicable), output of the method and number of parameters involved, runtime, reported error, behavior of the method with respect to our Examples 1–7 and some additional comments.

Table 2
Comparison of the proposed method with other alternative surface reconstruction methods.

Authors and year	Method	# Data points	Population	Output (parameters)	Runtime	Error	Examples 1–7	Additional comments
Not evolutionary approach								
Barak and Fischer [5] (2001)	Neural network SOM and PDE with gradient descent algorithm (GDA)	10^4	Not applicable here	Cubic polynomial B-spline	3–6 h	Data points: 10^{-1}–10^{-2}	Only Ex. 1 works properly. Ex. 2–7 fail	Requires projection of data points onto a base surface
Evolutionary approach								
Goinski [27] (2008)	Evolutionary algorithms (EA)	Not reported	30 (sphere) 30 (fractal) 50 (head)	Polygonal mesh (triangulation)	Tens of minutes to hours	Not reported	Ex. 1, 2, 3 and 7 work properly. Ex. 4–6 fail	Not actual surface is reconstructed (polygonal mesh)
Weinert et al. [85] (2001)	Evolutionary search (ES) and genetic programming (GP)	192	500	Polynomial B-spline (90 parameters)	24 h	Data points: 10^{-2}	Only Ex. 1 works properly. Ex. 2–7 fail	Neither parameterization nor weights are computed. Pre-processing required
Wagner et al. [80] (2007)	Multi-objective evolutionary and genetic algorithm (MOEA)	823–17,307	20	Polynomial B-spline (150 parameters)	Tens of hours to days	Data points: 10^{-2}	Only Ex. 1 works properly. Ex. 2–7 fail	Neither parameterization nor weights are computed. Pre-processing required
Our method (2010)	Particle swarm optimization (PSO)	400 (Ex. 1) 2476 (Ex. 2) 420 (Ex. 3) 1395 (Ex. 4) 9150 (Ex. 5) 4132 (Ex. 6) 5400 (Ex. 7)	60–1000	NURBS (902) NURBS (5313) NURBS (1073) NURBS (3462) NURBS (20,412) NURBS (9557) NURBS (57,660)	Tens of seconds to minutes	Data points: 10^{-8}–10^{-15} (worst cases: 10^{-3})	All work properly	All relevant data (knot vectors, data points parameterization, control points and weights) are computed. No pre-/post-processing is required

First method compared is that by Barhak and Fischer [5] where two different schemes, self-organizing map (SOM) neural network and partial differential equations (PDEs), are used to cope with the surface parameterization issue, and gradient descent algorithm (GDA) is used to create a 3D base surface that is iteratively modified to get the reconstructed surface. Such a surface is described in terms of cubic polynomial B-splines with errors of about 10^{-1}–10^{-2} for sets of about 10^4 data points. However, this method requires data points to be projected onto the base surface and therefore, only examples of neither self-intersecting nor high-genus surfaces are properly handled. Control points in this method have no weights either. As a result, the method fails for Examples 2–7 because data points cannot be unambiguously projected.

Next method by Goinski [27] is hard to be compared, since its output is a polygonal mesh instead of a real mathematical surface and neither the number of data points nor the errors are reported. In that paper, evolutionary algorithms (namely, evolutionary search with soft selection and $1 + \lambda$) are used to recover the shape of tessellated surfaces such as a sphere, a fractal surface and a head (the Igea model). For a population size of 30–50 particles, a polygonal mesh is obtained. The method is able to reconstruct Examples 1, 2, 3 and 7, which are topologically equivalent to some examples in that paper. Insurmountable problems arise, however, for Examples 4–6, where a proper triangulation cannot be obtained. Moreover, since the surface is only linearly approximated, points on the reconstructed surface different from data points exhibit very large errors, several orders of magnitude larger than ours.

The method by Weinert et al. [85] combines NURBS with Constructive Solid Geometry in a hybrid evolutionary algorithm/genetic programming approach. In practice, however, the method is much simpler and less powerful than ours since no parameterization of data points is actually computed. Instead, it is assumed that the NURBS patch represents a surface of a solid which is infinite in the negative direction of the z-axis. This constraint, designed to avoid the parameterization pro-

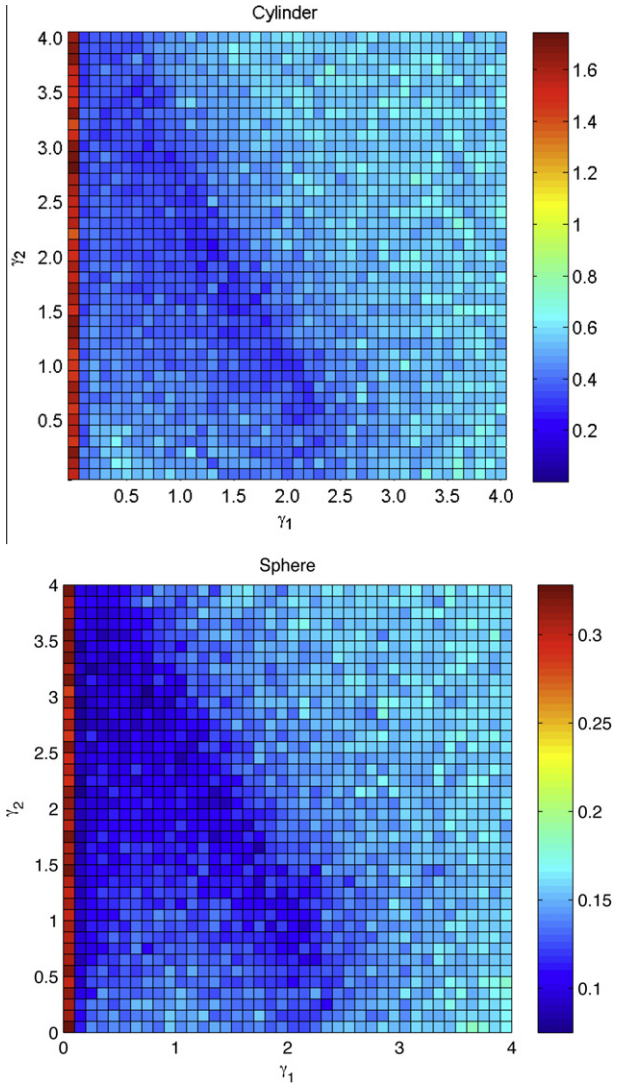

Fig. 7. Variation of accuracy for different values of the social and cognitive factors for the cylinder (above) and sphere (below) surfaces.

cess, has proved to be very limiting: Examples 2–7 of this paper cannot be reconstructed with this method. In fact, the method is limited to very simple examples in which a planar base surface can be used. And even in this case, reported errors for data points are quite large, of order 10^{-2} for an almost trivial example.

A very recent approach is given in [80], where a multi-objective evolutionary algorithm (MOEA) approach is applied to reconstruct a simple smooth surface with different sets of data points: the initial set of 17,307 data points was reduced to a set of 823 data points in order to decrease the runtime from several days to 9 h for errors of magnitude 10^{-2}. The method is also limited in several ways: on one hand, authors assume that all weights are equal to one and the degree is 3, so the NURBS surface actually becomes a cubic polynomial B-spline surface. On the other hand, it is assumed that sampled points are uniformly spaced in the (u, v) domain, so clouds of scattered data points cannot be reconstructed. Finally, data points should also be projected in order to perform the evaluation within the MOEA. As a consequence, the method is restricted to rather simple examples. In fact, the only reported example (which takes only seconds with our method for errors of about 10^{-14}) is by far much simpler than our surfaces. Also, Examples 2–7 of this paper cannot be reconstructed with this method.

To summarize, no other surface reconstruction approach described in the literature reported fitting errors as low as those achieved in this work. Besides, the examples analyzed here are, by far, much more complex and challenging that those found in previous papers. Our method also exhibits a remarkable flexibility, being able to adapt to a wide range of situations. In fact, this is the first evolutionary-based surface reconstruction method able to deal with NURBS surfaces at full extent, in the sense that our method returns all relevant data (data points parameterization, knot vectors, control points and weights), while previous methods introduce some assumptions instead of calculating all them. This is the reason why none of previous methods was able to reconstruct our Examples 2–7 with NURBS surfaces.

Regarding the computation times, our method is not applicable to real-time simulations. This fact can be attributed to the large amount of data required by the method. A careful analysis of Eq. (13) reveals that for a dense set of data points (e.g., about hundreds of thousands of points), search space dimension might be as large as 10^6, so we do not expect to obtain accurate results in a very short span (except for cases of points with a quite simple structure). However, the method is still faster than previous methods for a similar level of complexity. The paper in [5] reported times of about 200–350 min, while in [18] the fitting of tens of thousands of points took hundreds of minutes on a HP 735 workstation. However, this comparison is not totally fair, since those papers were written time ago and hardware and software have improved dramatically since then. A more pertinent comparison of runtimes can be done with recent entries [27,80]; the former reported runtimes from minutes to hours even although the output is a polygonal mesh, whereas the later reported times of days for sets of data points comparable to ours. It must also be noticed that all those methods reported much worse fitting errors than our method. This observation is important, since most of previous methods include an optimization stage in addition to the fitting [5,46,69]. Not by chance, such optimization stage is typically the time bottleneck [5] meaning that fitting error enhancement may demand significantly longer even for modest improvement factors.

6.4. Analysis of social and cognitive factors

In this section we report an empirical analysis of the role of social and cognitive factors on the accuracy of our approach. To this purpose, we applied our method to Examples 1–7 in this paper for values of γ_1 and γ_2 from 0 to 4 with step-size 0.1. This generates a matrix of 41×41 couples (γ_1, γ_2). For each, we carried out 10 executions and computed the mean value. The results are depicted in Fig. 7 for Examples 2 (the cylinder, above) and 3 (the sphere, below). Results for other examples are very similar and hence are omitted for shortness. Each figure represents the matrix of 41×41 couples (γ_1, γ_2) colored according to the percentage of variance with respect to the best result. Thus, dark blue[1] squares in Fig. 7 (above and below, respectively) means that the variance is less than (0.2% and 0.1%), respectively, for that particular choice of (γ_1, γ_2), while dark red means that variance is about (1.6% and 0.3%), respectively. Two important conclusions arise from these figures: firstly, there are some combinations that perform better than others and they follow a similar pattern for all our examples. Secondly, and most important, the accuracy of results is barely affected by our choice of (γ_1, γ_2). The worst cases produce a variance less than 2%, meaning that the reported results in this paper are robust against variations of (γ_1, γ_2) values.

7. Conclusions and future work

In this paper we introduce a general and efficient evolutionary-based NURBS surface reconstruction method from clouds of noisy 3D data points. The method relies heavily on the PSO approach to obtain all relevant parameters in order to construct the NURBS surface that fits the data points better. Major features of this method are a proper parameterization of data points and the very accurate fitting of the surface to the cloud of points. Our results show that the proposed method yields very good results even in presence of problematic features, such as multi-branches, high-genus or self-intersections. Seven examples including open, semiclosed, closed, zero-genus, high-genus surfaces and real-world scanned objects, described in free-form, parametric and implicit forms, illustrate the good performance of our approach. Our experiments show that our fitting errors are astonishingly small when compared to other previous approaches. Our proposal outperforms them in terms of accuracy and flexibility, being able to reconstruct very complicated shapes for which all previous methods fail.

[1] For interpretation of color in Fig. 7, the reader is referred to the web version of this article.

Although these results are quite encouraging and the approach is promising overall, there is still a long walk ahead. It is yet an open question how to optimize the process in order to reduce its computational time and complexity. It turns out that an adequate choice of relevant PSO parameters (such as inertia, social and cognitive factors, initial population and so on) may open the door for further improvement. While it is clear that the population size is a major factor in what concerns runtime (and all our experiments confirmed it), the influence of other PSO parameters in both accuracy and runtime is less obvious. Our empirical results show that the accuracy of results is barely affected by our choice of (γ_1, γ_2). However, further research is still needed to fully determine the optimal values of other PSO parameters in our approach.

Other future works include the parallelization of some processes, which might lead to some time improvement, although we are currently doubtful at which extent. Optimization of numerical routines used to solve the system of equations and the analysis of other communication topology strategies are also part of our future work. On the other hand, the reported method is very general and therefore can be applied to a wide range of practical problems. Consequently, we expect to apply our scheme to some problems reported in the literature. This task will give us very valuable feedback from real-world application settings towards the potential improvement of our current approach.

Acknowledgements

This research has been supported by the Computer Science National Program of the Spanish Ministry of Education and Science, Project Ref. #TIN2006-13615 and the University of Cantabria. Special thanks are owe to the anonymous reviewers for their encouraging comments and very helpful feedback that allowed us to improve our paper significantly. The revised version of this paper has been written after the unexpected death of second author's father on March 3, 2010. Authors would like to devote this paper to his memory. We also thank the Editor-in-Chief, Prof. Wiltod Pedrycz, and his staff for their support during those tough days.

References

[1] C. Ahn, J. An, J. Yoo, Estimation of particle swarm distribution algorithms: combining the benefits of PSO and EDAs, Information Sciences 192 (2012) 109–119.
[2] C.V. Alvino, A.J. Yezzi, Tomographic reconstruction of piecewise smooth images, Proceedings of Computer Vision and Pattern Recognition – CVPR'04, vol. 1, IEEE Computer Society Press, Los Alamitos, CA, 2004.
[3] C. Bajaj, F. Bernardini, G. Xu, Automatic reconstruction of surfaces and scalar fields from 3D scans, in: Proc. SIGGRAPH'95, 1995, pp. 109–118.
[4] C. Bajaj, E.J. Coyle, K.N. Lin, Arbitrary topology shape reconstruction from planar cross-sections, Graphical Models and Image Processing 58 (1996) 524–543.
[5] J. Barhak, A. Fischer, Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques, IEEE Transactions on Visualization and Computer Graphics 7 (1) (2001) 1–16.
[6] T. Beielstein, J. Mehnen, L. Schnemann, H.P. Schwefel, T. Surmann, K. Weinert, D. Wiesmann, Design of evolutionary algorithms and applications in surface reconstruction, in: H.P. Schwefel, I. Wegener, K. Weinert (Eds.), Advances in Computational Intelligence – Theory and Practice, Springer, Berlin, 2003, pp. 164–193.
[7] R.M. Bolle, B.C. Vemuri, On three-dimensional surface reconstruction methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (1) (1991) 1–13.
[8] E. Castillo, A. Iglesias, Some characterizations of families of surfaces using functional equations, ACM Transactions on Graphics 16 (3) (1997) 296–318.
[9] E. Castillo, A. Iglesias, R. Ruiz-Cobo, Functional Equations in Applied Sciences, Elsevier Science, Amsterdam, 2005.
[10] F. Chang, H. Huang, A refactoring method for cache-efficient swarm intelligent algorithms, Information Sciences 192 (2012) 39–49.
[11] J. Chang, P. Shi, Using investment satisfaction capability index based particle swarm optimization to construct a stock portfolio, Information Sciences 181 (2012) 2989–2999.
[12] M.G. Cox, Algorithms for spline curves and surfaces, in: L. Piegl (Ed.), Fundamental Developments of Computer-Aided Geometric Design, Academic Press, London, San Diego, 1993, pp. 51–76.
[13] N.R. Draper, H. Smith, Applied Regression Analysis, third ed., Wiley–Interscience, 1998.
[14] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Sciences 178 (15) (2008) 3096–3109.
[15] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 81–86.
[16] G. Echevarría, A. Iglesias, A. Gálvez, Extending neural networks for B-spline surface reconstruction, Lectures Notes in Computer Science 2330 (2002) 305–314.
[17] M. Eck, J. Hadenfeld, Local energy fairing of B-spline curves, in: G. Faring, H. Hagen, H. Noltemeier (Eds.), Computing, vol. 10, Springer-Verlag, 1995.
[18] M. Eck, H. Hoppe, Automatic reconstruction of B-spline surfaces of arbitrary topological type, in: Proc. SIGGRAPH'96, 1996, pp. 325–334.
[19] A.P. Engelbretch, Fundamentals of Computational Swarm Intelligence, John Wiley and Sons, Chichester, England, 2005.
[20] M.S. Floater, Parameterization and smooth approximation of surface triangulations, Computer Aided Geometric Design 14 (1997) 231–250.
[21] T.A. Foley, Interpolation to scattered data on a spherical domain, in: J.C. Mason, M.G. Cox (Eds.), Algorithms for Approximation, vol. II, Chapman and Hall, London, New York, 1990, pp. 303–310.
[22] D.R. Forsey, R.H. Bartels, Surface fitting with hierarchical splines, ACM Transactions on Graphics 14 (1995) 134–161.
[23] R.H. Franke, L.L. Schumaker, A bibliography of multivariate approximation, in: C.K. Chui, L.L. Schumaker, F.I. Utreras (Eds.), Topics in Multivariate Approximation, Academic Press, New York, 1986.
[24] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, J. Espinola, Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation, Lectures Notes in Computer Science 4706 (2007) 680–693.
[25] A. Gálvez, A. Iglesias, J. Puig-Pey, Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction, Information Science 182 (2012) 56–76.
[26] Geomagic. <http://www.geomagic.com>.
[27] A. Goinski, Evolutionary surface reconstruction, in: Proc. IEEE Conference on Human System Interactions, Krakow, Poland, 2008, pp. 464–469.
[28] W.J. Gordon, Spline-blended surface interpolation through curve networks, Journal of Mathematics and Mechanics 18 (10) (1969) 931–952.
[29] G. Greiner, Variational design and fairing of spline surfaces, Computer Graphics Forum 13 (3) (1994) 143–154.
[30] P. Gu, X. Yan, Neural network approach to the reconstruction of free-form surfaces for reverse engineering, Computer Aided Design 27 (1) (1995) 59–64.

[31] B. Guo, Surface reconstruction from points to splines, Computer Aided Design 29 (4) (1997) 269–277.
[32] Q. Guo, M. Zhang, A novel approach for multi-agent-based intelligent manufacturing system, Information Sciences 179 (18) (2009) 3079–3090.
[33] H. Hagen, P. Santarelli, Variational design of smooth B-spline surfaces, in: H. Hagen (Ed.), Topics in Surface Modeling, SIAM, 1992, pp. 85–94.
[34] M. Hoffmann, L. Varady, Free-form surfaces for scattered data by neural networks, Journal for Geometry and Graphics 2 (1998) 1–6.
[35] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: Proc. of SIGGRAPH'92, Computer Graphics, vol. 26(2), 1992, pp. 71–78.
[36] H. Hoppe, Surface Reconstruction from Unorganized Points, Ph.D. Thesis, Department of Computer Science and Engineering, University of Washington, 1994.
[37] J. Hoschek, Smoothing of curves and surfaces, Computer Aided Design 2 (1985) 97–105.
[38] M. Jones, M. Chen, A new approach to the construction of surfaces from contour data, Computer Graphics Forum 13 (3) (1994) 75–84.
[39] A. Iglesias, A. Gálvez, A new artificial intelligence paradigm for computer aided geometric design, Lectures Notes in Artificial Intelligence 1930 (2001) 200–213.
[40] A. Iglesias, G. Echevarría, A. Gálvez, Functional networks for B-spline surface reconstruction, Future Generation Computer Systems 20 (8) (2004) 1337–1353.
[41] E. Kaufman, R. Klass, Smoothing surfaces using reflection lines for families of splines, Computer Aided Design 20 (1988) 312–316.
[42] R.E. Keller, W. Banshaf, J. Mehnen, K. Weinert, CAD surface reconstruction from digitized 3D point data with a genetic programming/evolution strategy hybrid, Advances in Genetic Programming, vol. 3, MIT Press, Cambridge, MA, USA, 1999.
[43] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[44] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufman, San Francisco, 2001.
[45] T. Kodama, X. Li, K. Nakahira, D. Ito, Evolutionary computation applied to the reconstruction of 3-D surface topography in the SEM, Journal of Electron Microscopy 54 (5) (2005) 429–435.
[46] P. Laurent, M. Mekhilef, Optimization of a NURBS representation, Computer Aided Design 25 (11) (1993) 699–710.
[47] M.C. Leu, X. Peng, W. Zhang, Surface reconstruction for interactive modeling of freeform solids by virtual sculpting, CIRP Annals – Manufacturing Technology 54 (1) (2005) 131–134.
[48] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk, The digital Michelangelo project: 3D scanning of large statues, in: SIGGRAPH 2000, New Orleans, 2000, pp. 131–144.
[49] M. Li, Z. Wang, A hybrid coevolutionary algorithm for designing fuzzy classifiers, Information Sciences 179 (12) (2009) 1970–1983.
[50] C. Lim, G. Turkiyyah, M. Ganter, D. Storti, Implicit reconstruction of solids from cloud point sets, in: Proc. of 1995 ACM Symposium on Solid Modeling, Salt Lake City, Utah, 1995, pp. 393–402.
[51] W.Y. Ma, J.P. Kruth, Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces, Computer Aided Design 27 (1995) 663–675.
[52] I. Maekawa, K. Ko, Surface construction by fitting unorganized curves, Graphical Models 64 (2002) 316–332.
[53] E. Mendel, R.A. Krohling, M. Campos, Swarm algorithms with chaotic jumps applied to noisy optimization problems, Information Sciences 181 (2011) 4494–4514.
[54] D. Meyers, S. Skinnwer, K. Sloan, Surfaces from contours, ACM Transactions on Graphics 11 (3) (1992) 228–258.
[55] C. Oblonsek, N. Guid, A fast surface-based procedure for object reconstruction from 3D scattered points, Computer Vision and Image Understanding 69 (2) (1998) 185–195.
[56] H. Park, K. Kim, Smooth surface approximation to serial cross-sections, Computer Aided Design 28 (12) (1997) 995–1005.
[57] N.M. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer Verlag, 2002.
[58] L. Piegl, W. Tiller, The NURBS Book, Springer-Verlag, Berlin, Heidelberg, 1997.
[59] D. Picard, A. Revel, M. Cord, An application of swarm intelligence to distributed image retrieval, Information Sciences 192 (2012) 71–81.
[60] L. Piegl, W. Tiller, Algorithm for approximate NURBS skinning, Computer Aided Design 28 (9) (1997) 699–706.
[61] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization – an overview, Swarm Intelligence 1 (2007) 33–57.
[62] H. Pottmann, S. Leopoldseder, M. Hofer, Approximation with active B-spline curves and surfaces, in: Proceedings of Pacific Graphics, SIEEE Computer Society Press, 2002, pp. 8–25.
[63] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, W. Wang, Industrial geometry: recent advances and applications in CAD, Computer Aided Design 37 (2005) 751–766.
[64] M. Prasad, A. Zisserman, A.W. Fitzgibbon, Single view reconstruction of curved surfaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society Press, Los Alamitos, CA, 2006.
[65] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes, second ed., Cambridge University Press, Cambridge, 1992.
[66] G. Prestifillipo, J. Sprave, Optimal triangulation by means of evolutionary algorithms, in: Proc. Second Int. Conference on Genetic Algorithms in Engineering Systems, Glasgow, Scotland, 1997, pp. 492–497.
[67] J. Rossignac, B. Borrel, Multi-resolution 3D approximations for rendering complex scenes, Geometric Modeling in Computer Graphics, Springer-Verlag, 1993. pp. 455–465.
[68] M. Saleem, M. Farooq, G.A. Di Caro, Swarm intelligence based routing protocol for wireless sensor networks: survey and future directions, Information Sciences 181 (2011) 4597–4624.
[69] B. Sarkar, C.H. Menq, Parameter optimization in approximating curves and surfaces to measurement data, Computer Aided Geometric Design 8 (1991) 267–290.
[70] J.A. Sauter, R. Matthews, H.V.D. Parunak, S.A. Brueckner, Evolving adaptive pheromone path planning mechanisms, in: Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Bologna, Italy, 2002.
[71] J.A. Sauter, R. Matthews, H.V.D. Parunak, S.A. Brueckner, Effectiveness of digital pheromones controlling swarming vehicles in military scenarios, Journal of Aerospace Computing Information and Communication 4 (5) (2007) 753–769.
[72] V. Savchenko, A. Pasko, O. Okunev, T. Kunii, Function representation of solids reconstructed from scattered surface points and contours, Computer Graphics Forum 14 (4) (1995) 181–188.
[73] S. Sclaroff, A. Pentland, Generalized implicit functions for computer graphics, in: Proc. of SIGGRAPH'91, Computer Graphics, vol. 25(4), 1991, pp. 247–250.
[74] S. Sen, S.Q. Zheng, Near-optimal triangulation of a point set by simulated annealing, in: Proc. of 1992 ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990s, Kansas City, USA, 1992, pp. 1000–1008.
[75] Y. Shi, H. Liu, L. Gao, G. Zhang, Cellular particle swarm optimization, Information Sciences 181 (2011) 4460–4493.
[76] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Information Sciences 180 (11) (2010) 3182–3191.
[77] E. Ulker, A. Arslan, Automatic knot adjustment using an artificial immune system for B-spline curve approximation, Information Sciences 179 (10) (2009) 1483–1494.
[78] A. Unler, A. Murat, R.B. Chinnam, mr^{2}PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification, Information Sciences 181 (2011) 4625–4641.
[79] T. Varady, R. Martin, Reverse engineering, in: G. Farin, J. Hoschek, M. Kim (Eds.), Handbook of Computer Aided Geometric Design, Elsevier Science, 2002.
[80] T. Wagner, T. Michelitsch, A. Sacharow, On the design of optimizers for surface reconstruction, in: Proceedings of the 2007 Genetic and Evolutionary Computation Conference—GECCO2007, London, England, 2007, pp. 2195–2202.

[81] Y. Wang, Y. Yang, Particle swarm optimization with preference order ranking for multi-objective optimization, Information Sciences 179 (12) (2009) 1944–1959.
[82] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences 181 (2011) 4515–4538.
[83] Z. Wang, C. Chang, M. Li, Optimizing least-significant-bit substitution using cat swarm optimization strategy, Information Sciences 192 (2012) 98–108.
[84] K. Weinert, J. Mehnen, F. Albersmann, P. Dreup, New solutions for surface reconstruction from discrete point data by means of computational intelligence, in: Proceedings of the Intelligent Computation in Manufacturing Engineering—ICME'98, Capri, Italy, 1998, pp. 431–438.
[85] K. Weinert, T. Surmann, J. Mehnen, Evolutionary surface reconstruction using CSG-NURBS-hybrids, in: Proceedings of the 2001 Genetic and Evolutionary Computation Conference—GECCO2001, San Francisco, USA, 2001, pp. 1456–1463.
[86] V. Weiss, L. Or, G. Renner, T. Varady, Advanced surface fitting techniques, Computer Aided Geometric Design 19 (2002) 19–42.