ELSEVIER

Computer-Aided Design 36 (2004) 607–623

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Parameterization of clouds of unorganized points using dynamic base surfaces

Phillip N. Azariadis*

Department of Product and Systems Design Engineering, University of the Aegean, Ermoupolis, Syros 84100, Greece

Received 25 July 2002; received in revised form 21 May 2003; accepted 2 June 2003

Abstract

In this paper a new method for parameterizing clouds of unorganized points is presented. The proposed method introduces the notion of *dynamic base surfaces* (DBS) which are dynamically adapted to the three-dimensional shape implied by the clouds of points. The only assumption regarding the cloud of points is the existence of a boundary defined by a closed path of four curves. The proposed method is based on an iterative procedure where a DBS is gradually improved approximating more faithfully the fundamental geometry of the cloud of points. Parameterization is achieved by orthogonally projecting the cloud of points onto the DBS. An application of the introduced parameterization method to the well-known surface least-squares fitting is presented which illustrates the effectiveness and the efficiency of the proposed approach.

Keywords: Parameterization; Reverse engineering; Surface fitting; Base surfaces

1. Introduction

Reverse engineering is a well-known process utilized to construct geometrical models of existing objects. Nowadays, a large area of applications of reverse engineering has been developed. Examples include the production of a copy of a part of an existing object when the original drawings are not available. In other cases, one may need to re-engineer an existing part, when more analysis and/or modifications are required to construct a new improved product. Reverse engineering is required, in cases where stylists rely more often on evaluating real 3D objects than on working with viewing projections of objects on high resolution 2D screens at reduced scale.

A generic approach of reverse engineering comprises of the basic phases illustrated in the flowchart in Fig. 1. Using data acquisition devices, a cloud of points is captured. Depending on the data acquisition method, this cloud of points can be organized, where topological information for each point is known, or unorganized otherwise. Segmentation divides the cloud of points into subsets which correspond to certain regions of the object surface where a single surface can be fitted. A parameterization process is invoked to assign to each point of the cloud adequate parameter values which will be utilized during surface fitting. Usually, a well segmented and parameterized cloud of points ensures accurate surface fitting. A detailed review of the different reverse engineering phases is given in Ref. [1].

This paper introduces a new method for the parameterization of a cloud of unorganized points, which has been segmented into 'rectangular' subsets bounded by a closed path of curves. More specifically, given four boundary curves of b_p order

$$\mathbf{C}_\ell(u) = \sum_i N_{i,bp}(u)\mathbf{P}_i^\ell, \qquad \ell = 0, 1 \tag{1}$$

$$\mathbf{C}_k(v) = \sum_j N_{j,bp}(v)\mathbf{P}_j^k, \qquad k = 0, 1 \tag{2}$$

and a set of unorganized points

$$\mathbf{p}_\mu = (x_\mu, y_\mu, z_\mu) \in \Re^3, \qquad \mu = 0, \ldots, N-1$$

which is provided after segmentation, this paper addresses the problem of assigning adequate parameter values

$$(u_\mu, v_\mu) \in [0,1] \times [0,1] \subset \Re^2, \qquad \mu = 0, \ldots, N-1$$

that best represent the points' position within the rectangular domain in the plane. Contrarily to existing methods, there is no restriction regarding the density and the homogeneity of

* Tel.: +30-22810-97129; fax: +30-22810-97009.
E-mail address: azar@aegean.gr (P.N. Azariadis).

0010-4485/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0010-4485(03)00138-6

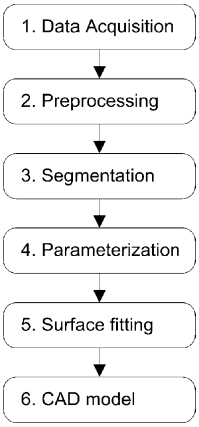

Fig. 1. The basic phases of reverse engineering.

the cloud of points. This implies that in planar areas of the model surface, the density of the cloud of points can be significantly reduced and vice versa. Such a cloud of points can be regarded as the result of an 'ideal 3D scanner' described in Ref. [1]. Fig. 2 describes the bounded cloud of points of a half shoe last. It is obvious that the density of the cloud of points varies significantly; there are dense regions, which correspond to high-curvature areas, and coarse regions, which correspond to almost flat areas. This introduces many technical hurdles which should be avoided during parameterization.

This paper is organized as follows. In Section 2 an overview of previous approaches for the parameterization problem is presented. In Section 3, the proposed parameterization method is introduced. An application to the well-known surface least-squares (LSQ) fitting is described in Section 4. Various examples and test cases are illustrated and discussed in Section 5, and the paper concludes with a summary of the results and hints for potential future work in Section 6.

2. Previous work

Many authors have proposed methods for parameterizing organized points [2–16]. For parameterization of irregularly spaced and randomly distributed data points, usually a simple surface (called as *base surface*) is created, which roughly reflects the global shape of the cloud of points. This might be a plane, a cylinder, or a bilinearly blended Coons patch. Ma and Kruth [10] used interactively defined section curves together with the four boundary curves to obtain a base surface, which roughly approximates the shape of the cloud of points. Parameter values are then computed by projecting data points onto the base surface. There are a variety of methods to construct the base surface; these may fail if there are irregularly oriented highly curved subregions or the data points cannot be projected in an unambiguous way.

Several methods start from an underlying 3D triangulation of the cloud of points. Generally, through iterative steps, a topologically identical 2D triangulation is obtained, which defines the parameter values of the vertices in the domain plane. These methods differ in how the transformation is performed from 3D to 2D. Harmonic parameterizations minimize the squared distances of the edge lengths [3], Floater's shape preserving approach locates the new vertices of the interior triangles using barycentric mappings [4]. In both cases, first an appropriate convex domain needs to be defined to map the corner points of the 3D point region. Greiner and Hormann [6] suggested to project the boundary points onto a best fit plane and minimise the energy of springs associated with the internal edges of the triangulation. Later in Ref. [7], the same authors suggested an improved technique—called the Most Isometric Parameterization—where the boundaries do not need to be fixed in advance and evolve in a more natural way. In Ref. [8], Hormann et al. utilized hierarchical representations of triangulated surfaces in order to speed up the global parameterization problem. Through this process, the global parameterization problem is represented by a hierarchical parameterization of different levels of detail.

Other approaches can be utilized which are based on mapping a triangulated surface onto the plane by minimizing an energy function which leads to an approximate locally isometric planar development of the 3D surface [11–15]. In Ref. [14], it has been shown that when

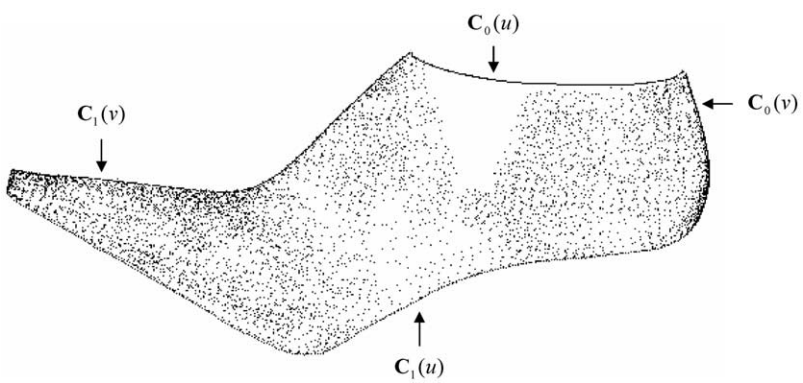

Fig. 2. A example of a bounded cloud of unorganized points ($N = 7132$).

the surface is approximated with a sufficient number of triangular elements, the shape of the corresponding planar development converges to a steady stage, e.g. the insertion of more triangles will not disturb the shape (and thus the mapping onto the planar domain) of the final planar development. Several of these techniques have been utilized successfully for texture mapping with minimal distortions [11–13,15]. These approaches, however, seem to be slow if the number of data points is large and somewhat limited if there are concave parts and hole loops within the cloud of points.

However, common to all the aforementioned methods is the assumption that the points are either structured in some kind of mesh or topological information regarding their spatial position is known. A generalization of the method presented in Ref. [4] is proposed by Floater and Reiners [17], for parameterizing unorganized points by solving a global linear system. The linear equations arise from demanding that each interior parameter point be some convex combination of some neighbouring ones. Hoppe et al. [18], introduced a method for producing 'simplicial' surfaces approximating an unorganized set of points in $\Re^3$. This method demands the density of the cloud of points to be under a predefined threshold ρ, which means that each point in the cloud should have at least one neighbouring point at maximum Euclidean distance ρ.

Pottmann et al. [19] proposed a method similar in spirit with the method introduced in this paper. In particular, the authors developed a scheme where a so-called 'active' curve or surface approximates a model shape through an iterative procedure where the quasi-Newton optimization procedure is employed to minimize in each iteration a quadratic objective function. The objective function is a convex combination of the squared distance function of the model shape and a smoothing functional in order to avoid unwanted loops or crossovers during the development of the active curve or surface. Although their method avoids the parameterization problem during fitting, it is fundamentally dissimilar to the proposed one since the utilized functionals and the overall approach is quite different.

Piegl and Tiller [20] proposed a method for parameterizing unorganized clouds of points by projecting them onto a base surface, which is fitted to the given boundary curves and to some of the data points. However, it is not clear how many points to select and how to parameterize them to construct the base surface. The authors propose to use up to 30% of the data points to produce a relatively accurate base surface, which leads to long computational times especially with large datasets. Also, the selection of the data points implies a well sampling of the cloud of points with an even density factor.

Thus, we conclude that there is no safe, 'universal' method for a good initial parameterization. One may obtain some parameterization, which often needs to be improved before the actual surface fitting process can start. A new method for producing parameterizations of clouds of unorganized points with variable sampling density is proposed in this paper, where the notion of DBS is introduced.

3. The proposed parameterization method

3.1. Outline of the proposed algorithm

The basic idea of the proposed method is to construct a base surface which is dynamically refined using an iterative scheme where section curves are projected along certain directions onto the cloud of points until a terminating criterion is satisfied. Initially the *dynamic base surface* (DBS) is fitted to the four boundary curves. Section curves are selected, comprising a so-called *grid*, which correspond to base surface isoparametric curves. The grid is projected onto the cloud of points in a matter that minimizes a combined error function. The base surface is fitted to the projected grid and it is validated to ensure that no unwanted self-loops or crossovers exist.

Contrarily to other approaches [10], the base surfaces introduced in this paper are *dynamic* and they evolve starting from a rough to a very accurate approximation of the fundamental geometry of the cloud of points allowing for good parameterizations without solving LSQ systems. The final DBS approximates faithfully the cloud of points. In some cases, there is no further need to fit a surface under the least-squares sense, since the fit tolerance of the final DBS lies within the nominal accuracy of the measuring device or the data acquisition method. The flowchart of the proposed parameterization method is shown in Fig. 3 and it is analytically described in the following paragraphs.

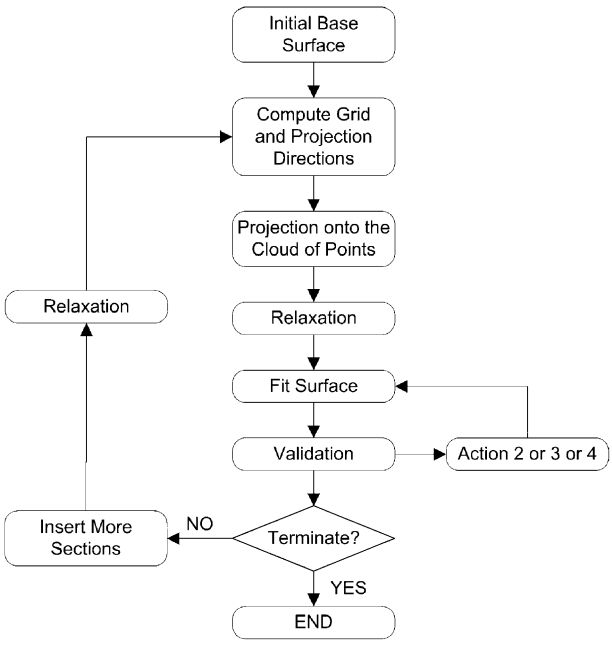

Fig. 3. The flowchart of the proposed parameterization method.

3.2. Creation of the initial dynamic base surface

Given the four boundary curves $\mathbf{C}_\ell(u)$, $\mathbf{C}_k(v)$ ($\ell = 0, 1$ and $k = 0, 1$), an initial DBS is created using a NURBS representation of a Coons bilinear blended patch, which merely approximates the cloud of points, and is given by Ref. [21, p. 304]

$$\mathbf{S}(u, v) = \mathbf{R}_1(u, v) + \mathbf{R}_2(u, v) - \mathbf{T}(u, v) \tag{3}$$

where $u, v \in [0, 1]$, $\mathbf{R}_1$ and $\mathbf{R}_2$ are ruled surfaces along the u and v parametric directions, respectively, and $\mathbf{T}$ is a bilinear tensor product surface defined by the four corner points of the given boundary. Coons' surfaces provide a good and very fast initial approximation to the cloud of points.

3.3. Projection of section curves onto the cloud of points

Let $\mathbf{S} = \mathbf{S}(u, v)$ be the current DBS (which is derived by the last iteration or after the previous step in the case of the first iteration), $\mathbf{R}_i = \mathbf{R}_i(v) = \mathbf{S}(u_i, v)$ be n v-isoparametric curves defined at n discrete values of the u parameter (e.g. $u = u_i \in [0, 1]$, $u_{i-1} < u_i$), and $\mathbf{R}_j = \mathbf{R}_j(u) = \mathbf{S}(u, v_j)$ be m u-isoparametric curves defined at m discrete values of the v parameter (e.g. $v = v_j \in [0, 1]$, $v_{j-1} < v_j$), where $i = 0, \ldots, n-1, j = 0, \ldots, m-1$, and $\mathbf{R}_0(v) \equiv \mathbf{C}_0(v)$, $\mathbf{R}_{n-1}(v) \equiv \mathbf{C}_1(v)$, $\mathbf{R}_0(u) \equiv \mathbf{C}_0(u)$, $\mathbf{R}_{m-1}(u) \equiv \mathbf{C}_1(u)$. The grid is constructed in such a way that each grid point corresponds to $\mathbf{p}_{i,j} = \mathbf{R}_j(u) \cap \mathbf{R}_i(v) = \mathbf{S}(u_i, v_j)$. Totally, there are $n \times m$ grid points which are associated with surface normal vectors defined by[1]

$$\mathbf{n}_{i,j} = \frac{\mathbf{S_u}(u_i, v_j) \times \mathbf{S_v}(u_i, v_j)}{\|\mathbf{S_u}(u_i, v_j) \times \mathbf{S_v}(u_i, v_j)\|}$$

where

$$\mathbf{S_u}(u_i, v_j) = \left.\frac{\partial \mathbf{S}(u, v)}{\partial u}\right|_{\substack{u=u_i \\ v=v_j}}$$

and

$$\mathbf{S_v}(u_i, v_j) = \left.\frac{\partial \mathbf{S}(u, v)}{\partial v}\right|_{\substack{u=u_i \\ v=v_j}}.$$

Note that the grid can be regarded in a dual manner: as a set of points $\mathbf{p}_{i,j}$ or as a set of isoparametric curves $\mathbf{R}_i$ or $\mathbf{R}_j$ which correspond to a 'column'- or to a 'row'-section, respectively. In the former case, the grid is usually called as *grid of points* while in the latter case we refer to a *grid section curve* or *grid section* for simplicity (it should be cleared by the context whether it is a 'row' or a 'column' section). Each grid section is composed by a subset of grid points.

Three methods are proposed in the following for projecting the grid onto the cloud of points following the associated directions in such a way that an adequate *error* with respect to the cloud of points is minimized.

3.3.1. Squared distances error

Following the notation used in the previous paragraphs, let $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N-1}$ be the cloud of points with corresponding positive weights $a_0, a_1, a_2, \ldots, a_{N-1}$. Each point has coordinates $\mathbf{p}_\mu = (x_\mu, y_\mu, z_\mu)$ and let $\mathbf{p}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$ be the point of interest, which corresponds to one particular grid point. The weighted sum of the squared distances from the specified point to the cloud of points is given as

$$\begin{aligned} E(\mathbf{p}_{i,j}) &= \sum_{\mu=0}^{N-1} a_\mu \|\mathbf{p}_{i,j} - \mathbf{p}_\mu\|^2 \\ &= \sum_{\mu=0}^{N-1} a_\mu [(x_{i,j} - x_\mu)^2 + (y_{i,j} - y_\mu)^2 + (z_{i,j} - z_\mu)^2] \end{aligned} \tag{4}$$

Equivalently, for the given cloud of points, a five-dimensional vector $\mathbf{c} = (c_0, c_1, c_2, c_3, c_4)$ is computed:

$$\begin{aligned} &c_0 = \sum_{\mu=0}^{N-1} a_\mu,\ c_1 = \sum_{\mu=0}^{N-1} a_\mu x_\mu,\ c_2 = \sum_{\mu=0}^{N-1} a_\mu y_\mu, \\ &c_3 = \sum_{\mu=0}^{N-1} a_\mu z_\mu,\ c_4 = \sum_{\mu=0}^{N-1} a_\mu (x_\mu^2 + y_\mu^2 + z_\mu^2) \end{aligned} \tag{5}$$

Using vector $\mathbf{c}$ one can quickly calculate $E(\mathbf{p}_{i,j})$ by [22]

$$E(\mathbf{p}_{i,j}) = c_0(x_{i,j}^2 + y_{i,j}^2 + z_{i,j}^2) - 2(c_1 x_{i,j} + c_2 y_{i,j} + c_3 z_{i,j}) + c_4 \tag{6}$$

Now, let $\mathbf{p}_{i,j}^* = (x_{i,j}^*, y_{i,j}^*, z_{i,j}^*)$ be the result of the projection of the point $\mathbf{p}_{i,j}$ onto the cloud of points following an associated direction $\mathbf{n}_{i,j} = (n_{i,j}^x, n_{i,j}^y, n_{i,j}^z)$. Then $\mathbf{p}_{i,j}^*$ can be written as

$$\mathbf{p}_{i,j}^* = \mathbf{p}_{i,j}^*(t) = \mathbf{p}_{i,j} + t\mathbf{n}_{i,j} \tag{7}$$

where t is the unknown parameter. Substituting Eq. (7) into Eq. (6) yields

$$\begin{aligned} E(\mathbf{p}_{i,j}^*(t)) &= c_0[(x_{i,j}^*(t))^2 + (y_{i,j}^*(t))^2 + (z_{i,j}^*(t))^2] \\ &\quad - 2(c_1 x_{i,j}^*(t) + c_2 y_{i,j}^*(t) + c_3 z_{i,j}^*(t)) + c_4 \end{aligned} \tag{8}$$

Proposition 1. *The minimum of function* $E : \Re \to \Re$ (*Eq.* (8)) *with respect to* t, *is defined at*

$$t = \frac{\lambda - \mathbf{p}_{i,j} \cdot \mathbf{n}_{i,j}}{\|\mathbf{n}_{i,j}\|^2}$$

where

$$\lambda = \frac{c_1 n_{i,j}^x + c_2 n_{i,j}^y + c_3 n_{i,j}^z}{c_0}$$

and $\cdot$ *denotes dot product.*

[1] The Euclidean norm of vector $\mathbf{a}$ is denoted by $\|\mathbf{a}\|$. The Euclidean distance between $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\|\mathbf{a} - \mathbf{b}\|$.

Proof. The necessary condition for the minimization of Eq. (8) with respect to t is

$$E'(\mathbf{p}^*_{i,j}(t)) = 0 \Rightarrow t = \frac{\lambda - \mathbf{p}_{i,j}\cdot\mathbf{n}_{i,j}}{\|\mathbf{n}_{i,j}\|^2} \tag{9}$$

and since $E''(\mathbf{p}^*_{i,j}(t)) = 2c_0\|\mathbf{n}_{ij}\|^2 > 0$, the parameter t defined by Eq. (9) corresponds to the minimum of Eq. (8). □

Utilizing Eqs. (7) and (9), an efficient algorithm for projecting the grid of points $\mathbf{p}_{i,j}$ onto the cloud of points following the directions defined by $\mathbf{n}_{i,j}$ is implemented. During our experiments two different weight factors have been used

$$a_\mu = \frac{1}{\|\mathbf{p}_{i,j} - \mathbf{p}_\mu\|^4} \tag{10}$$

and

$$a_\mu = \mathrm{e}^{-\frac{\|\mathbf{p}_{i,j}-\mathbf{p}_\mu\|^2}{D}} \tag{11}$$

where $D = \max\{\|\mathbf{p}_{i,j} - \mathbf{p}_\mu\|^2 | \mu = 0, \ldots, N-1\}$. However, it has been noticed that both weight factors produce almost identical results. For this reason, it is chosen to adapt Eq. (10) due to superior speed performance. In the rare case where the points $\mathbf{p}_{i,j}$ and $\mathbf{p}_{\mu 0}$ coincide for a $\mu_0 \in \{0, \ldots, N-1\}$ then we simply set $t = 0$, i.e. the projected grid point is $\mathbf{p}_{\mu 0}$. The combination of Eqs. (7), (9) and (10) is considered as an operator, namely SDE(*,*), which is used for projecting a grid point $\mathbf{p}_{i,j}$ associated with a direction $\mathbf{n}_{i,j}$ to the given cloud of points, i.e. $\mathbf{p}^*_{i,j} = \mathrm{SDE}(\mathbf{p}_{i,j}, \mathbf{n}_{i,j})$.

3.3.2. Minimizing the length of the projected grid sections

It must be ensured that the final DBS will be smooth, without unwanted crossovers or self-loops which can be produced by an inadequate grid. Both requirements can be achieved to some degree by minimizing the length of the grid sections during their projection onto the cloud of points.

3.3.2.1. Projection of a grid section onto the cloud of points. First, a modification to the SDE(*,*) operator is developed in order to project one grid section (either a 'row' or a 'column' of points) to the cloud of points. Without loss of generality, let $\mathbf{p}_{i0,j}$ be a 'row' of m points and $\mathbf{n}_{i0,j}$ ($j = 0, \ldots, m-1$) be the corresponding directions, where $0 \le i_0 \le n-1$. The first and last point, namely $\mathbf{p}_{i0,0}$ and $\mathbf{p}_{i0,m-1}$, belong to the given boundary and can not be written in the form of Eq. (7). Thus, there are totally $(m-2)$ points in each row-section that are projected to the cloud of points. The overall error of each section with respect to the cloud of points is given as a function

$$f : \Re^{m-2} \to \Re,$$

$$\text{with } f(t_1, t_2, \ldots, t_{m-2}) = \sum_{j=1}^{m-2} E(\mathbf{p}_{i0,j}(t_j)) \tag{12}$$

Remark 1. The minimum of function $f : \Re^{m-2} \to \Re$ (Eq. (12)) with respect to $\mathbf{t} = (t_1, t_2, \ldots, t_{m-2})$ is defined at

$$t_j = \frac{\lambda_j - \mathbf{p}_{i0,j}\cdot\mathbf{n}_{i0,j}}{\|\mathbf{n}_{i0,j}\|^2}$$

for $j = 1, \ldots, m-2$. In vector form, $\mathbf{t}$ is given as

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_{m-2} \end{pmatrix} = \begin{pmatrix} \dfrac{\lambda_1 - \mathbf{p}_{i0,1}\cdot\mathbf{n}_{i0,1}}{\|\mathbf{n}_{i0,1}\|^2} \\ \dfrac{\lambda_2 - \mathbf{p}_{i0,2}\cdot\mathbf{n}_{i0,2}}{\|\mathbf{n}_{i0,2}\|^2} \\ \vdots \\ \dfrac{\lambda_{m-2} - \mathbf{p}_{i0,m-2}\cdot\mathbf{n}_{i0,m-2}}{\|n_{i0,m-2}\|^2} \end{pmatrix} \tag{13}$$

where $\lambda_j = (c^1_{i0,j}n^x_{i0,j} + c^2_{i0,j}n^y_{i0,j} + c^3_{i0,j}n^z_{i0,j})/c^0_{i0,j}$. The scalars $c^0_{i0,j}, c^1_{i0,j}, c^2_{i0,j}, c^3_{i0,j}$ are defined through Eq. (5) for each $\mathbf{p}_{i0,j}$ and $\mathbf{n}_{i0,j} = (n^x_{i0,j}, n^y_{i0,j}, n^z_{i0,j})$ for $j = 1, \ldots, m-2$. The Remark 1 is a direct consequence of Proposition 1.

Eq. (13) can be written in matrix form as

$$\mathbf{A}\mathbf{t} = \mathbf{b} \tag{14}$$

where $\mathbf{A}$ is the $(m-2)\times(m-2)$ identity matrix and $\mathbf{b}$ is the right-hand vector of Eq. (13). The squared chord lengths function of the projection of the row-section onto the cloud of points is given by

$$f_s : \Re^{m-2} \to \Re,$$

$$\text{with } f_s(t_1, t_2, \ldots, t_{m-2}) = \sum_{j=0}^{m-2} \|\mathbf{p}^*_{i0,j} - \mathbf{p}^*_{i0,j+1}\|^2 \tag{15}$$

The minimization of f_s with respect to $\mathbf{t} = (t_1, t_2, \ldots, t_{m-2})$ leads to the aforementioned request of computing the projection with minimal length. It holds

$$\mathbf{p}^*_{i0,0} \equiv \mathbf{p}_{i0,0},\ \mathbf{p}^*_{i0,m-1} \equiv \mathbf{p}_{i0,m-1} \text{ (boundary points) and}$$

$$\mathbf{p}^*_{i0,j} = \mathbf{p}^*_{i0,j}(t_j) = \mathbf{p}_{i0,j} + t_j\mathbf{n}_{i0,j}, \quad \text{for } j = 1, \ldots, m-2 \tag{16}$$

and

$$\frac{\partial \mathbf{p}^*_{i0,j}}{\partial t_j} = \mathbf{n}^*_{i0,j}, \qquad \text{for } j = 1, \ldots, m-2 \tag{17}$$

A necessary condition for minimizing f_s is the vanishing of the first derivatives, i.e. $\nabla f_s = \mathbf{0}$, which yields the linear system

$$\mathbf{A}_s\mathbf{t} = \mathbf{b}_s \tag{18}$$

where $\mathbf{A}_s$ is a $(m-2)\times(m-2)$ tridiagonal and symmetric matrix. The three non-zero elements of the jth row of $\mathbf{A}_s$ are

$$\mathbf{A}^{j,j-1}_s = -\mathbf{n}_{i0,j-1}\cdot\mathbf{n}_{i0,j}, \qquad \mathbf{A}^{j,j}_s = 2,$$
$$\mathbf{A}^{j,j+1}_s = -\mathbf{n}_{i0,j}\cdot\mathbf{n}_{i0,j+1} \tag{19}$$

for $j = 2, \ldots, m-3$. The two special cases for $j = 1$ and $j = m-2$ correspond to

$$\mathbf{A}_s^{1,1} = 2, \qquad \mathbf{A}_s^{1,2} = -\mathbf{n}_{i0,1} \cdot \mathbf{n}_{i0,2} \tag{20}$$

$$\mathbf{A}_s^{m-2,m-3} = -\mathbf{n}_{i0,m-2} \cdot \mathbf{n}_{i0,m-3}, \qquad \mathbf{A}_s^{m-2,m-2} = 2 \tag{21}$$

In matrix form $\mathbf{A}_s$ is written as

$$\mathbf{A}_s = \begin{pmatrix} 2 & -\mathbf{n}_{i0,1} \cdot \mathbf{n}_{i0,2} & & & \\ -\mathbf{n}_{i0,1} \cdot \mathbf{n}_{i0,2} & 2 & -\mathbf{n}_{i0,2} \cdot \mathbf{n}_{i0,3} & & \\ \vdots & \vdots & \vdots & & \\ & & -\mathbf{n}_{i0,m-4} \cdot \mathbf{n}_{i0,m-3} & 2 & -\mathbf{n}_{i0,m-2} \cdot \mathbf{n}_{i0,m-3} \\ & & & -\mathbf{n}_{i0,m-2} \cdot \mathbf{n}_{i0,m-3} & 2 \end{pmatrix} \tag{22}$$

The jth element of $\mathbf{b}_s$ vector is given by

$$\mathbf{b}_s^j = (\mathbf{p}_{i0,j-1} - \mathbf{p}_{i0,j}) \cdot \mathbf{n}_{i0,j} + (\mathbf{p}_{i0,j} - \mathbf{p}_{i0,j+1}) \cdot \mathbf{n}_{i0,j} \tag{23}$$

for $j = 1, \ldots, m-2$.

Taking the above into consideration the following proposition is stated.

Proposition 2. *The minimum of* $f_s : \Re^{m-2} \rightarrow \Re$ (*Eq.* (15)) *with respect to* $\mathbf{t} = (t_1, t_2, \ldots, t_{m-2})$ *is given at* $\mathbf{t} = \mathbf{A}_s^{-1}\mathbf{b}_s$.

3.3.2.2. Combined projection onto the cloud of points. Eqs. (12) and (15) express two different objective functions with respect to the projection of a grid section onto the cloud of points. Therefore, a convex combination of these objective functions is given by combining Eqs. (14) and (18)

$$[(1-\kappa)\mathbf{A} + \kappa\mathbf{A}_s]\mathbf{t} = (1-\kappa)\mathbf{b} + \kappa\mathbf{b}_s, \tag{24}$$

with $\kappa \in [0, 1]$

Then one derives the solution

$$\mathbf{t} = [(1-\kappa)\mathbf{A} + \kappa\mathbf{A}_s]^{-1}[(1-\kappa)\mathbf{b} + \kappa\mathbf{b}_s], \tag{25}$$

with $\kappa \in [0, 1]$

which corresponds to the combined projection of the grid section onto the cloud of points with respect both to the squared distances error and the section's length. A smoother DBS with larger deviations from the cloud of points will be produced by a big κ than the DBS that which will be produced by a small κ. Thus, one can manipulate κ in such a way that a smaller value is assigned when the DBS has captured the fundamental geometry of the cloud of points in order to accelerate the adaptation speed and improve local accuracy.

Eq. (24) represents a system of m linear equations with a m unknowns expressed by the tridiagonal and symmetric matrix $[(1-\kappa)\mathbf{A} + \kappa\mathbf{A}_s]$. This system can be efficiently solved through decomposition into lower and upper triangular components and a forward/backward substitution utility [23,24; p. 31], which is an $O(m)$ process. The combination of Eqs. (12), (15) and (25) is considered as a new operator, namely SSDE(*,*), which is used for projecting a row-section (resp. column-section) composed of grid points $\mathbf{p}_{i0,j}$ with associated directions $\mathbf{n}_{i0,j}$, for $j = 1, \ldots, m-2$, to the given cloud of points, i.e. $\mathbf{p}_{i0,j}^* = \text{SSDE}(\mathbf{p}_{i0,j}, \mathbf{n}_{i0,j})$.

3.3.3. Gross error

For simplicity, let again $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N-1}$ be the cloud of points, and $\mathbf{p}_{i,j}$ the point of interest, which corresponds to one particular grid point and is associated with one direction vector $\mathbf{n}_{i,j}$. According to Fig. 4, the error of point $\mathbf{p}_{i,j}$ with respect to $\mathbf{p}_\mu$ of the cloud of points is defined by

$$\begin{aligned} E_\mu(\mathbf{p}_{i,j}) &= [E_\mu^1(\mathbf{p}_{i,j})]^2 + E_\mu^2(\mathbf{p}_{i,j})E_\mu^3(\mathbf{p}_{i,j}) \\ &= [E_\mu^1(\mathbf{p}_{i,j})]^2[1 + E_\mu^2(\mathbf{p}_{i,j})] + [E_\mu^2(\mathbf{p}_{i,j})]^3 \end{aligned} \tag{26}$$

where

$$E_\mu^1(\mathbf{p}_{i,j}) = \|(\mathbf{p}_\mu - \mathbf{p}_{i,j}) \times n_{i,j}\|^2 \tag{27}$$

$$E_\mu^2(\mathbf{p}_{i,j}) = [(\mathbf{p}_\mu - \mathbf{p}_{i,j}) \cdot \mathbf{n}_{i,j}]^2 \tag{28}$$

$$E_\mu^3(\mathbf{p}_{i,j}) = \|\mathbf{p}_\mu - \mathbf{p}_{i,j}\|^2 \tag{29}$$

In other words, utilizing Eq. (26), one can search for the $\mathbf{p}_\mu$ point that satisfies two conditions: (a) its distance from the axis of projection is minimal, and (b) the length of the projection of the vector $(\mathbf{p}_\mu - \mathbf{p}_{i,j})$ onto the axis of the projection is minimal. This is achieved through a greedy search where each point $\mathbf{p}_\mu$ is tested against each $\mathbf{p}_{i,j}$ according to Eq. (26). This process can be accelerated by defining subsets of the cloud of points in the areas near

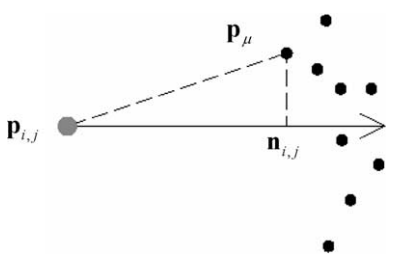

Fig. 4. The projection of a grid point (e.g. $\mathbf{p}_{i,j}$) onto the cloud of points using the 'gross' error function.

the grid points. If $\mathbf{p}^*_{i,j} = (x^*_{i,j}, y^*_{i,j}, z^*_{i,j})$ is the result of the projection of point $\mathbf{p}_{i,j}$ onto the cloud of points following the associated direction $\mathbf{n}_{i,j}$, then $\mathbf{p}^*_{i,j}$ is given by

$$\mathbf{p}^*_{i,j} = \mathbf{p}_{i,j} + t\mathbf{n}_{i,j} \tag{30}$$

where $t = (\mathbf{p}_\mu - \mathbf{p}_{i,j})\cdot\mathbf{n}_{i,j}$ corresponds to the $\min_\mu E_\mu(\mathbf{p})$. In a similar fashion, the combination of Eqs. (26) and (30) is considered as an operator GE(*,*) which is used for projecting a point $\mathbf{p}_{i,j}$ associated with a direction $\mathbf{n}_{i,j}$ to the given cloud of points, i.e. $\mathbf{p}^*_{i,j} = \mathrm{GE}(\mathbf{p}_{i,j}, \mathbf{n}_{i,j})$. GE(*,*) operator yields best results when the overall error between the grid and the cloud of points is under a small threshold. For this reason, it is possible to use GE(*,*) as a final refinement step in the parameterization algorithm in order to increase accuracy. Examples are illustrated and discussed in Section 5.

3.4. 'Relaxation' of the projected grid

After the projection of the grid onto the cloud of points a 'relaxation' procedure is employed to derive a grid with almost evenly distributed points. This procedure is comprised of a set of interpolations along the u and v direction of the projected grid. Fixing the $n \times m$ points of the projected grid, n interpolations are performed along the u direction and m interpolations along the v direction. A cubic B-spline with an optimal knot vector based on a routine described in De Boor [25] is utilized for interpolating the projected grid rows $\mathbf{p}^*_{i0,j}$ and columns $\mathbf{p}^*_{i,j0}$ $(0 \le i_0 \le n-1,\ 0 \le j_0 \le m-1)$.

Let G^u (with points denoted as $\mathbf{g}^u_{i,j}$) and G^v (with points denoted as $\mathbf{g}^v_{i,j}$) be the two resulting grids after the interpolation along the u and v direction, respectively. Then the final grid is derived through averaging, e.g.

$$\mathbf{p}^{**}_{i,j} = \frac{\mathbf{g}^u_{i,j} + \mathbf{g}^v_{i,j}}{2}$$

for $i = 0, \ldots, n-1$ and $j = 0, \ldots, m-1$. The effect of the relaxation method in the grid is discussed further in Section 5.

3.5. Fitting the DBS to the grid

Given the set of $n \times m$ grid points, $\mathbf{p}^{**}_{i,j}$, $i = 0, \ldots, n-1$ and $j = 0, \ldots, m-1$, a non-rational (p, q)th-degree tensor product B-spline surface is constructed interpolating these grid points, i.e.

$$\mathbf{p}^{**}_{i,j} = \mathbf{S}(u_i, v_j) = \sum_{r=0}^{n-1} \sum_{c=0}^{m-1} N_{r,p}(u_i) N_{c,q}(v_j) \mathbf{P}_{r,c} \tag{31}$$

The unknown parameters are the points of the control net $\mathbf{P}_{r,c}$, $r = 0, \ldots, n-1$ and $c = 0, \ldots, m-1$. A global surface fitting interpolation algorithm based on a method described in Ref. [26, §9.2.5] is used. The basic steps are:

1. Compute adequate parametric values $(\bar{u}_i, \bar{v}_j)$ for the given grid points $\mathbf{p}^{**}_{i,j}$, using the chord length or centripetal method [27] and by *averaging* [26, p. 365].
2. Compute the knot vectors U and V by *averaging*.
3. Using U and $\bar{u}_i$, do m curve interpolations through $\mathbf{p}^{**}_{0,j}, \ldots, \mathbf{p}^{**}_{n,j}$ for $j = 0, \ldots, m-1$; this yields an intermediate net $\mathbf{R}_{r,j}$.
4. Using V and $\bar{v}_j$, do n curve interpolations through $\mathbf{R}_{r,0}, \ldots, \mathbf{R}_{r,m}$ for $r = 0, \ldots, m-1$; this yields control net $\mathbf{P}_{r,c}$.

In the intermediate steps, a compatible global curve interpolation method is used which is equivalent to solving a series of tridiagonal linear systems, which is very efficient.

3.6. Validation process

Given the final grid of $\mathbf{p}^{**}_{i,j}$ points derived according to the last step, the next task is to fit the DBS to that grid and check its validity. That is, in order to generate an adequate and even parameterization it is important that the derived DBS to be free of unwanted crossovers and self-loops. In the following, a DBS with crossovers is considered invalid. In any other case it is valid.

3.6.1. Checking for crossovers

In Ref. [28], testing for crossovers is performed using the following two steps:

1. Orthogonal projection of the triangles defined by the surface control net $\mathbf{P}_{r,c}$ $(r = 0, \ldots, n-1$ and $c = 0, \ldots, m-1)$ onto the base surface.
2. Check whether the domain images of the projected triangles keep their original orientation or not.

The aforementioned technique is based on the assumption that the control net is near to the actual surface. In our case, however, the above criterion is not adequate for checking the validity of the projected grid $\mathbf{p}^{**}_{i,j}$ for two reasons:

1. The control net is not close to the actual surface (especially in the early stages of the proposed algorithm).
2. The surface control net and the projected grid $\mathbf{p}^{**}_{i,j}$ differ significantly, especially in the early stages of the proposed algorithm.

Thus, the aforementioned technique may result in wrong estimations about the existence of unwanted crossovers and/or self-loops in the DBS. Instead, a more generic approach is used which checks whether there are crossovers among the actual grid section curves (e.g. column-sections) since this is a sufficient condition for deriving invalid DBSes. The proposed technique is based on the notion of half-spaces. In particular, let $\mathbf{p}^{**}_{i,j0}$ and $\mathbf{p}^{**}_{i,j1}$ be two consecutive grid column-sections $(i = 0, \ldots, n-1)$. We

shall check whether section $\mathbf{p}^{**}_{i,j1}$ lies entirely in the positive (or negative) half-space defined by a plane passing through $\mathbf{p}^{**}_{i,j0}$ as in the following:

For each point of the $\mathbf{p}^{**}_{i,j0}$ section, i.e. for $i = 1,\ldots,n-1$, do:

a. Compute the plane H_i defined by $\mathbf{p}^{**}_{i-1,j0}$, $\mathbf{p}^{**}_{i,j0}$ and $\mathbf{n}^{**}_{i,j0}$ (where $\mathbf{n}^{**}_{i,j0}$ is the new normal vector of the DBS at $\mathbf{p}^{**}_{i,j0}$).
b. If $i = 1$, set an initial *flag* f (positive or negative) indicating whether the point $\mathbf{p}^{**}_{1,j1}$ of the next column-section lies in the positive or negative half-space with respect to H_i.
c. If $i > 1$, check whether the flag of point $\mathbf{p}^{**}_{i,j1}$ with respect to H_i equals f. If not, then return a flag indicating *failure*.

Although the proposed crossover test does not guarantee that crossovers will always be detected, in practice it has been proved enough robust. This is justified if we take into account that in the context of the proposed algorithm: (a) neighbouring sections do not have significant differences in their length, (b) the relaxation process produces a grid of almost evenly distributed sections, and (c) the projection of the grid sections onto the cloud of points incorporates a smoothing term which minimizes the length of the projected sections. Thus, it is almost impossible for the crossover test to fail to indicate an existing crossover. If such a case occurs, then it is more likely that the fundamental geometry of the cloud of points is too complex in order to dynamically fit a unique surface to it.

If the *failure* flag is retuned from the aforementioned process, then a crossover between successive sections has been detected. There are four alternative actions to be followed:

1. *Terminate the algorithm*. The DBS approximates the cloud of points with a reasonable accuracy and thus an adequate parameterization of the cloud of points can be achieved.
2. *Compute geodesic grid sections*. Based on the DBS and the grid of the previous iteration, *geodesic sections* are computed which approximate geodesic curves on that DBS. The DBS is re-fitted to the new grid.
3. *Increase smoothing term*. It is possible that an increase in the smoothing term κ (Eq. (25)) would result in a valid grid. Thus, the last valid grid with the corresponding DBS is recovered and the smoothing term κ is increased.
4. *Remove the grid sections which are stamped as invalid*. The grid sections, which failed to satisfy the half-spaces criterion, are removed. Then either geodesic sections are computed or the smoothing term is increased.

Usually, the first three choices are followed in the majority of the problematic cases and they are automatically performed in the present implementation. However, the last choice requires the user's decision and it is used as an uttermost effort to derive a valid DBS. From the above four choices, the second and third attempt to 'repair' an invalid DBS. More details concerning the second action are given in next paragraph.

3.6.1.1. Computing geodesic sections. Fitting a DBS to a grid composed of geodesic sections—that is surface isoparametric curves that connect points along corresponding boundaries with minimal length—minimizes the risk to derive an invalid DBS. This condition does not guarantee that crossovers will not occur in any type of surface, but during our experiments, it has been proved very effective since it gave the required solution in all our test cases. The technique consists of the following main steps:

1. Recover the last grid of points $\mathbf{p}^{\xi-1}_{i,j}$, $i = 0,\ldots,n_{\xi-1}-1$ and $j = 0,\ldots,m_{\xi-1}-1$ (with $n_{\xi-1}\times m_{\xi-1}$ number of points) which correspond to a valid DBS without crossovers.[2]
2. For each row-section $\mathbf{p}^{\xi-1}_{i0,j}$ (resp. column-section), $j = 0,\ldots,m_{\xi-1}-1$, compute the geodesic curve connecting the curve boundary points $\mathbf{p}^{\xi-1}_{i0,0}$ and $\mathbf{p}^{\xi-1}_{i0,m_{\xi-1}-1}$. This yields a $\mathbf{g}^{\xi-1}_{i,j}$ grid ($i = 0,\ldots,n_{\xi-1}-1$ and $j = 0,\ldots,m_{\xi-1}-1$).
3. Re-fit the DBS to the $\mathbf{g}^{\xi-1}_{i,j}$ grid, increase the number of sections such as $n_{\xi-1}\rightarrow n$ and $m_{\xi-1}\rightarrow m$, and derive the new 'repaired' grid $\mathbf{p}^{**}_{i,j}$ ($i = 0,\ldots,n-1$ and $j = 0,\ldots,m-1$).

It is clear, that the proposed algorithm should make use of the last 'good' grid, since it is meaningless to search for geodesic sections in an invalid surface with crossovers. For this reason, the last valid grid is recovered (iteration number: $\xi-1$) and refined in order to prevent an invalid surface from occurring in the next iteration (iteration number: ξ). In the second step of the above algorithm, the computation of geodesic sections takes place. Each grid row-section $\mathbf{p}^{\xi-1}_{i0,j}$ (resp. column-section) is approximated by a non-rational second-degree B-spline defined in the parametric domain [0,1] as

$$\mathbf{C}_g = \mathbf{C}_g(t) = \mathbf{C}_g(u(t), v(t)) = \mathbf{C}_g\left(\sum_{\sigma=0}^{3} N_{\sigma,2}(t)\mathbf{P}_\sigma\right) \tag{32}$$

All control points $\mathbf{P}_\sigma$ are defined in the parametric domain $[0,1]\times[0,1]$. The control points $\mathbf{P}_0$ and $\mathbf{P}_3$ are fixed in the boundary, while the control points, $\mathbf{P}_1$ and $\mathbf{P}_2$, are computed in such a way that the arc length L of $\mathbf{C}_g(t)$ is minimized, where [29, chap. 11]

$$L = \int_0^1 \left[E\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)^2 + 2F\frac{\mathrm{d}u}{\mathrm{d}t}\frac{\mathrm{d}v}{\mathrm{d}t} + G\left(\frac{\mathrm{d}v}{\mathrm{d}t}\right)^2 \right]^{1/2} \mathrm{d}t \tag{33}$$

The coefficients E, F and G are the quantities of the first

[2] The superscript or subscript ξ denotes the current iteration of the proposed algorithm.

Table 1
Summary of the obtained results of all the tests carried out in Section 5

Cloud of points				DBS			Least-squares solution				Projection (cloud to surface)
Object	No points	Sampling accuracy (mm)	Bounding box $W \times H \times D$ (mm)	Grid dimensions	No geodesic sections evaluation	Mean square error (E_{mean})	Time (s)	Control net	Smoothness coefficient, a (%)	Mean square error (E_{mean})	Time (s)	Time (s)
Shoe last (Fig. 5)	7132	3×10^{-1}	$260 \times 45 \times 119$	20×20	1	0.149	4.628	20×20	1	0.112	10.557 (QR factorization)	0.636
Boot last (Fig. 6)	12,454	3×10^{-1}	$265 \times 80 \times 138$	22×35	1	0.239	14.152	22×35	1	0.193	75.765 (QR factorization)	1.208
Car	19,579	10^{-2}	$123 \times 300 \times 60$	17×17	0	0.392	3.051	17×17	1	0.067	20.401 (QR factorization)	1.941
Face	11,469	3×10^{-1}	$174 \times 183 \times 241$	39×39	0	0.757	12.860	39×39	1	0.537	474.070 (QR factorization)	1.408
Face	11,469	3×10^{-1}	$174 \times 183 \times 241$	39×39	0	0.757	12.860	39×39	1	0.505	157.467 (conjugate gradient)	1.408
Toy	21,120	3×10^{-1}	$90 \times 15 \times 38$	29×29	0	0.195	8.198	29×29	1	0.162	388.085 (QR factorization)	1.933
Toy	21,120	3×10^{-1}	$90 \times 15 \times 38$	29×29	0	0.195	8.198	29×29	1	0.136	189.222 (conjugate gradient)	1.933
Heel	11,448	3×10^{-1}	$48 \times 50 \times 82$	14×14	0	0.223	2.093	14×14	1	0.167	6.272 (QR factorization)	1.244
Torus part	62.5K	–	$94 \times 127 \times 300$	17×17	0	10^{-3}	5.006					5.087

Note: The values of the mean square error (E_{mean}) in the columns DBS and least-squares solution have been computed using Eq. (34).

fundamental form of the DBS. Only four control points are used in the present implementation, since a fast computation of the geodesic sections is more important than a high-accurate one. Although in the majority of our experiments, the proposed approach yielded satisfactory results, one may use more elaborated approaches, i.e. insert more knots, fixing the 'valid' control points and setting the rest 'invalid' control points as unknown parameters.

3.7. Terminating criterion

The final step of the proposed parameterization method is to assert whether to terminate the described iterative procedure. There are three criterions:

- The DBS approximates the cloud of points with an accepted accuracy.
- The dimension of the grid has reached a predefined threshold.
- The maximum number of iterations is surpassed.

The first criterion is satisfied when the mean error of the fitting accuracy is under a predefined tolerance *tol*, i.e.

$$E_{\text{mean}} = \frac{1}{N} \sum_{\mu=0}^{N-1} \|\mathbf{S}(u_\mu, v_\mu) - \mathbf{p}_\mu\|^2 \leq \text{tol} \tag{34}$$

However, Eq. (34) is expensive to be computed in each iteration of the proposed algorithm. On the other hand, Eq. (6) or Eq. (26) can be utilized to compute a good estimation of the current surface fitting accuracy. For example, let the error function based on Eq. (6) at the ξ-iteration be

$$E^{\xi}_{\text{mean}} = \sum_{i=1}^{n-2} \sum_{j=1}^{m-2} E(\mathbf{p}_{i,j}(t_j)) \tag{35}$$

Then the terminating condition is written as

$$\frac{E^{\xi}_{\text{mean}}}{E^{\xi-1}_{\text{mean}}} \to 1 \tag{36}$$

The second terminating condition is based on the fact that during the iterative adaptation of the DBS, the dimension of the grid of points is increased by inserting more curve sections, i.e. $n \to n+1$ and $m \to m+1$. Thus, the algorithm terminates when the number of rows and columns has surpassed the predefined thresholds n_{max} and $m_{\max}$, respectively, i.e.

$$n > n_{\max}, \qquad m > m_{\max} \tag{37}$$

If none of the above conditions is satisfied, more column- and/or row-sections are inserted and the algorithm continues with a new grid. In the opposite case, the cloud of points is orthogonally projected onto the surface obtaining the required (u_μ, v_μ) parameters.

4. Application in surface fitting

The proposed parameterization method is utilized mainly in the final stage of reverse engineering: surface fitting. However, other uses of the proposed algorithm include the derivation of 'good' initial guesses for surface flattening, meshing and optimal triangulation, non-distorted texture mapping, and many others. In this paper, the results of the proposed parameterization method are applied to the widely used surface LSQ fitting. The outline of the surface fitting algorithm is described briefly in the sequel.

4.1. Surface LSQ fitting

A common approach for surface fitting followed by many researchers is to compute a smooth surface $\mathbf{S}$, which approximates each point $\mathbf{p}_\mu$ of the points cloud within tolerance ε_μ

$$\|\mathbf{S}_{\mathbf{p}_\mu} - \mathbf{p}_\mu\| \leq \varepsilon_\mu, \qquad \mu = 0, \ldots, N-1 \tag{38}$$

$\mathbf{S}_{\mathbf{p}_\mu}$ denotes the point on the surface associated to the data point $\mathbf{p}_\mu$. For free-form shapes, B-spline surfaces are most favourite. This is due to their particular unlimited degree of geometric freedom and their simple mathematical model, which leads to fast and stable computational algorithms. Having computed the final DBS, the parameters (u_μ, v_μ) are assigned by orthogonally projecting the cloud of points onto that base surface, which is achieved in the present implementation by solving a local optimization problem through a modified Newton method and an analytically supplied Hessian. The initializing values of this local optimization process correspond to the parameters of the grid point, which is nearest to the projected cloud point. Then surface fitting is formulated as a LSQ problem

$$\min\left\{\sum_{\mu=0}^{N-1} \|\mathbf{S}(u_\mu, v_\mu) - \mathbf{p}_\mu\|^2\right\} \tag{39}$$

Using the Euclidean norm and fixing the knot vectors and the parameter values the optimization problem, Eq. (39) can be generally solved. The unknown quantities are the control points of the B-spline surface. It is obvious that the minimum solution will not necessarily meet the tolerance criterion and nothing guarantees the smoothness of the surface. In order to increase fitting accuracy, more degrees of freedom are provided by inserting more knots. The aesthetic requirements are met by incorporating various smoothness functionals in the general LSQ problem.

In the present implementation, surface fitting is formulated under the LSQ sense as an optimization problem defined as

$$\min_{\mathbf{x}\in R^{n\times m\times 3}} \{E_{\text{lsq}}(\mathbf{x})\} \text{ with } E_{\text{lsq}}(\mathbf{x}) = E_d(\mathbf{x}) + aE_s(\mathbf{x}) \tag{40}$$

where $0 \leq a \leq 1$ is the smoothness weighting factor, which determines the relation between the LSQ E_d and the smoothing E_s terms. A big a will produce a smoother surface with larger deviations from the data points than

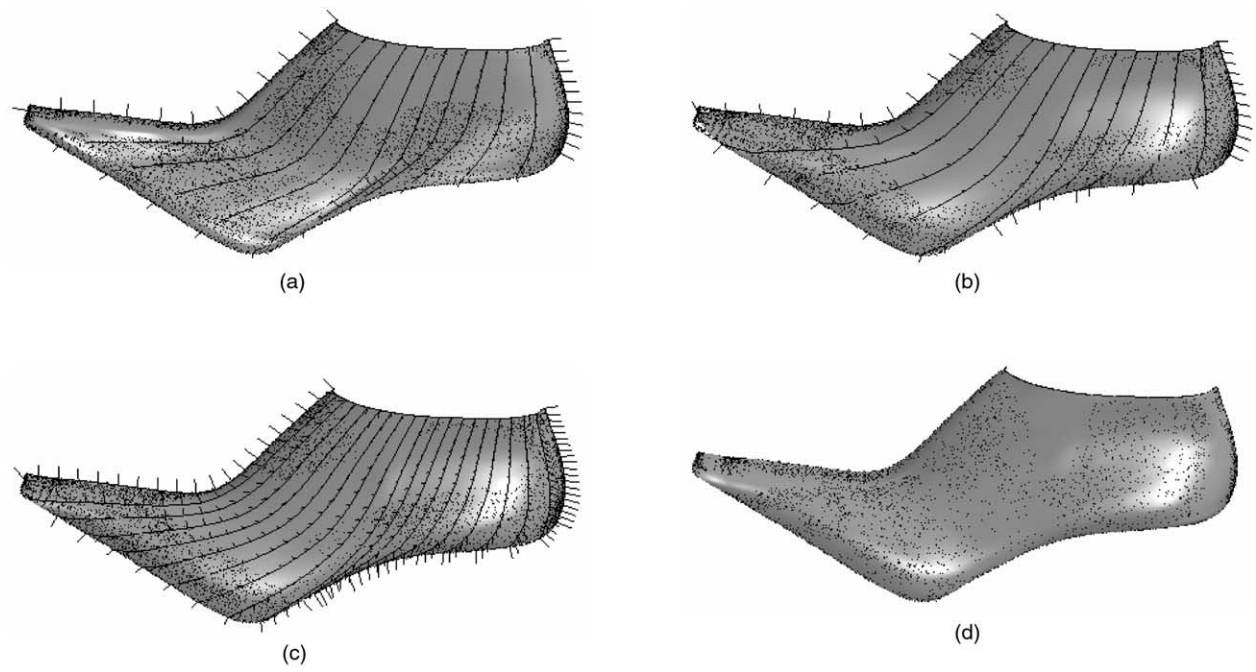

Fig. 5. The complete reverse engineering example of the outer half of the shoe last ($N = 7132$). (a) The initial DBS derived as a Coons bilinear patch. (b) The initial grid projected onto the cloud of points using the SSDE(*,*) operator. The DBS is immediately improved: it is smoother and closer to the cloud of points. (c) The final DBS (grid: 20 rows × 20 columns) after eight iterations. (d) The final surface computed through LSQ.

the surface that will be produced with a small a. It is assumed that the surface is defined by a control net $\mathbf{P}_{r,c}$ ($r = 0,\ldots,n-1$ and $c = 0,\ldots,m-1$) with totally $n \times m$ points, which are the actual unknown parameters represented by a matrix $\mathbf{x} \in \Re^{M\times 3}$, where $M = n \times m$. The two functionals E_d and E_s are given as

$$E_d(\mathbf{x}) = \sum_{\mu=0}^{N-1} \|\mathbf{S}(u_\mu, v_\mu) - \mathbf{p}_\mu\|^2 \tag{41}$$

and

$$E_s(\mathbf{x}) = \frac{1}{2}\sum_{c=0}^{n-1}\sum_{r=0}^{m-2} \|\mathbf{P}_{r+1,c} - \mathbf{P}_{r,c}\|^2 + \frac{1}{2}\sum_{c=0}^{m-2}\sum_{r=0}^{n-1} \|\mathbf{P}_{r,c+1} - \mathbf{P}_{r,c}\|^2 \tag{42}$$

Note that $\mathbf{x}_k = \mathbf{P}_{r(k),c(k)}$, where $r(k) = \lfloor k/m \rfloor$ and $c(k) = k \times \mathrm{mod}\, m$. E_s is used as smoothness functional to improve the quality of the surface by minimizing the 'tension' of the control net. This yields a surface with minimal wrinkles. Since the functionals E_d and E_s are positive and quadratic in the control points $\mathbf{x}$, the necessary and sufficient condition for their minimization is the vanishing of the partial derivatives with respect to the control points $\mathbf{x}$. In matrix formulation, this leads to the linear system

$$(\mathbf{K}_d + \alpha\mathbf{K}_s)\mathbf{x} = \mathbf{b} \qquad \text{or} \qquad \mathbf{K}_{\text{comp}}\mathbf{x} = \mathbf{b} \tag{43}$$

where

$$\mathbf{K}_d = \mathbf{N}^{\mathrm{T}}\mathbf{N} \qquad \text{and} \qquad \mathbf{b} = \mathbf{N}^{\mathrm{T}}\mathbf{p} \tag{44}$$

$\mathbf{K}_s$ is a tridiagonal $M \times M$ matrix, which is composed by tridiagonal and symmetric submatrices $\mathbf{K}_s^\lambda \in \Re^{n\times n}$, $\lambda = 0,\ldots,m-1$, where the diagonal of each submatrix $\mathbf{K}_s^\lambda$

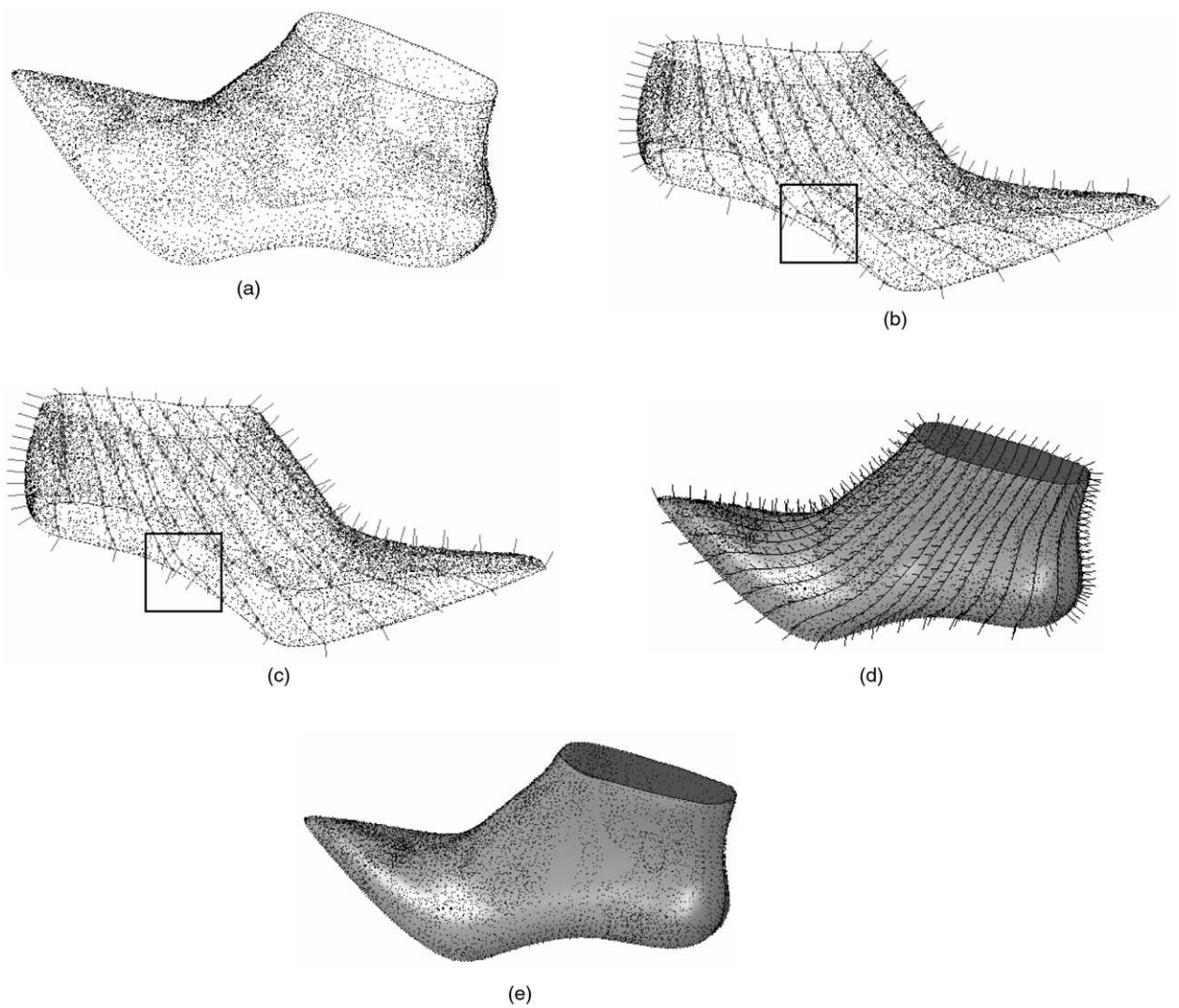

Fig. 6. Reverse engineering of a boot last ($N = 12,454$). (a) The cloud of points. (b) Crossovers between grid curve sections: should be repaired to avoid deriving an invalid DBS. (c) The repaired grid after the evaluation of geodesic sections. (d) The final DBS (grid: 22×35) computed using the SSDE(*,*) operator and GE(*,*) for the last iteration. (e) The final surface (22×35 control net) corresponding to the solution of the LSQ problem.

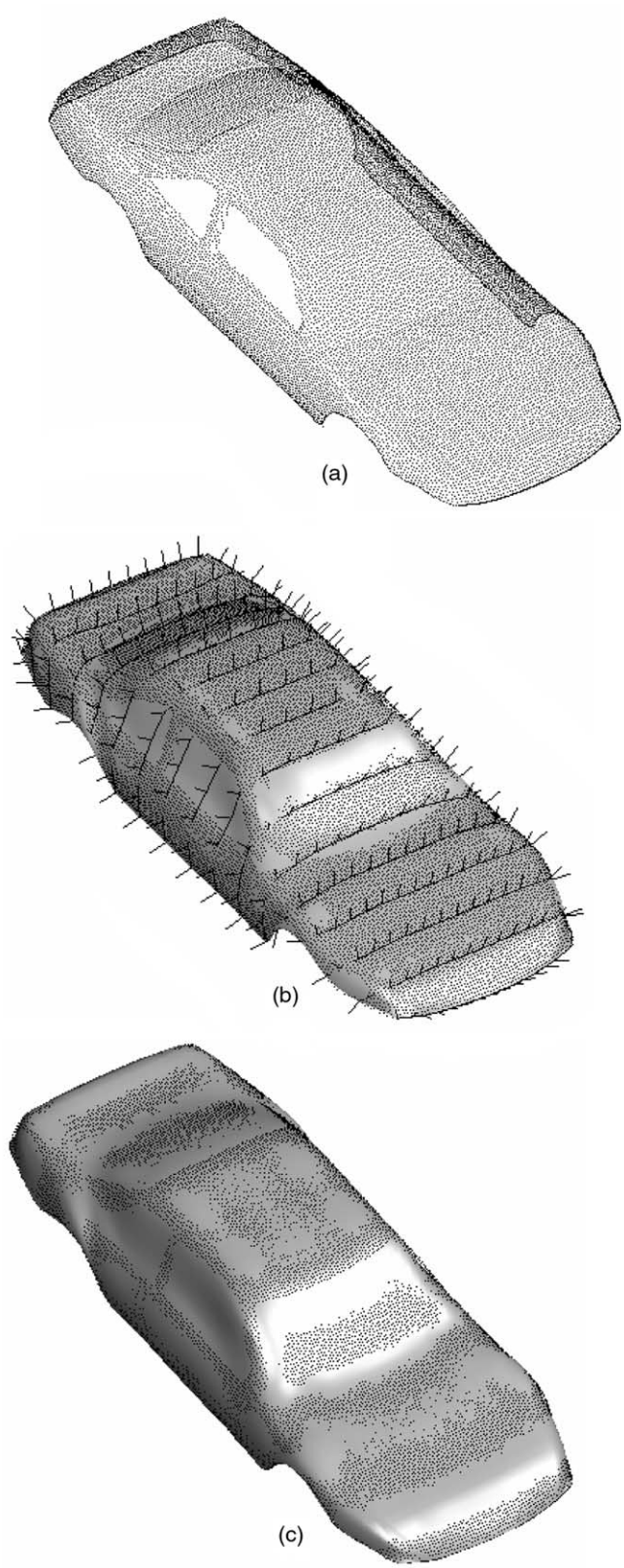

Fig. 7. Reverse engineering of a car body ($N = 19,579$). (a) The cloud of points with 'empty' areas corresponding to window seats. (b) The final DBS (grid: 17×17) computed using the SSDE(*,*) operator and GE(*,*) for the last iteration. (c) The final surface (17×17 control net) corresponding to the solution of the LSQ problem.

coincides with the diagonal of $\mathbf{K}_s$. Matrices $\mathbf{K}_s^\lambda$ have the following form:

$$\mathbf{K}_s^\lambda = \begin{pmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{pmatrix} \tag{45}$$

The matrix of the base function values is $\mathbf{N} = (N_k(u_\mu, v_\mu))_{\mu,k} \in \Re^{N\times M}$ where $N_k(u_\mu, v_\mu) = N_{r(k),p}(u_\mu)N_{c(k),q}(v_\mu)$, and the matrix of the data point coordinates is $\mathbf{p} \in \Re^{N\times 3}$. In general, when the number of the data points exceeds the number of control points, the composite matrix $\mathbf{K}_{\text{comp}}$ is regular and the control points can be computed through $\mathbf{x} = \mathbf{K}_{\text{comp}}^{-1}\mathbf{b}$.

The aforementioned surface LSQ fitting problem has been solved through four different approaches:

1. Direct solution of Eq. (43), by computing $\mathbf{K}_d = \mathbf{N}^{\mathrm{T}}\mathbf{N}$ and $\mathbf{K}_{\text{comp}}^{-1}$. However, both operations require $O(n^3)$ operations and can be extremely time consuming (especially the multiplication $\mathbf{N}^{\mathrm{T}}\mathbf{N}$) when the number of data points $N \gg 1$ or the number of control points $M \gg 1$.
2. Through QR decomposition of $\mathbf{N}$ using Householder transformations [30]: $\mathbf{NP} = \mathbf{QR}$. $\mathbf{P}$ is a permutation matrix ($\mathbf{P}^{-1} = \mathbf{P}^{\mathrm{T}}$), $\mathbf{Q}$ is an orthogonal matrix ($\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{T}}$) and $\mathbf{R}$ is an upper trapezoidal matrix. Thus, Eq. (43) can be replaced by

$$(\mathbf{R}^{\mathbf{T}}\mathbf{R}\mathbf{P}^{\mathbf{T}} + a\mathbf{K}_s)\mathbf{x} = \mathbf{R}^{\mathbf{T}}(\mathbf{Q}^{\mathbf{T}}\mathbf{b}) \tag{46}$$

3. Using conjugate gradient method [31] and analytic gradient vector to solve Eq. (40). This approach yields good results and it is faster than the first approach when the number of data points $N \gg 1$ or the number of control points $M \gg 1$.
4. Using a modified Levenberg–Marquardt algorithm and analytic Jacobian [32,33]. This approach is based on nonlinear LSQ formulation and it is by far the most expensive one with respect to the computational time.

Generally, in the current implementation, the first solution is used when $N < 2000$ and the number of control points $M < 100$, the second when $N < 2 \times 10^4$ and $M < 1500$, and the third when $N > 2 \times 10^4$.

5. Examples—discussion

Several examples are presented in this section to illustrate the effectiveness of the proposed parameterization method. It must be pointed out that the presented

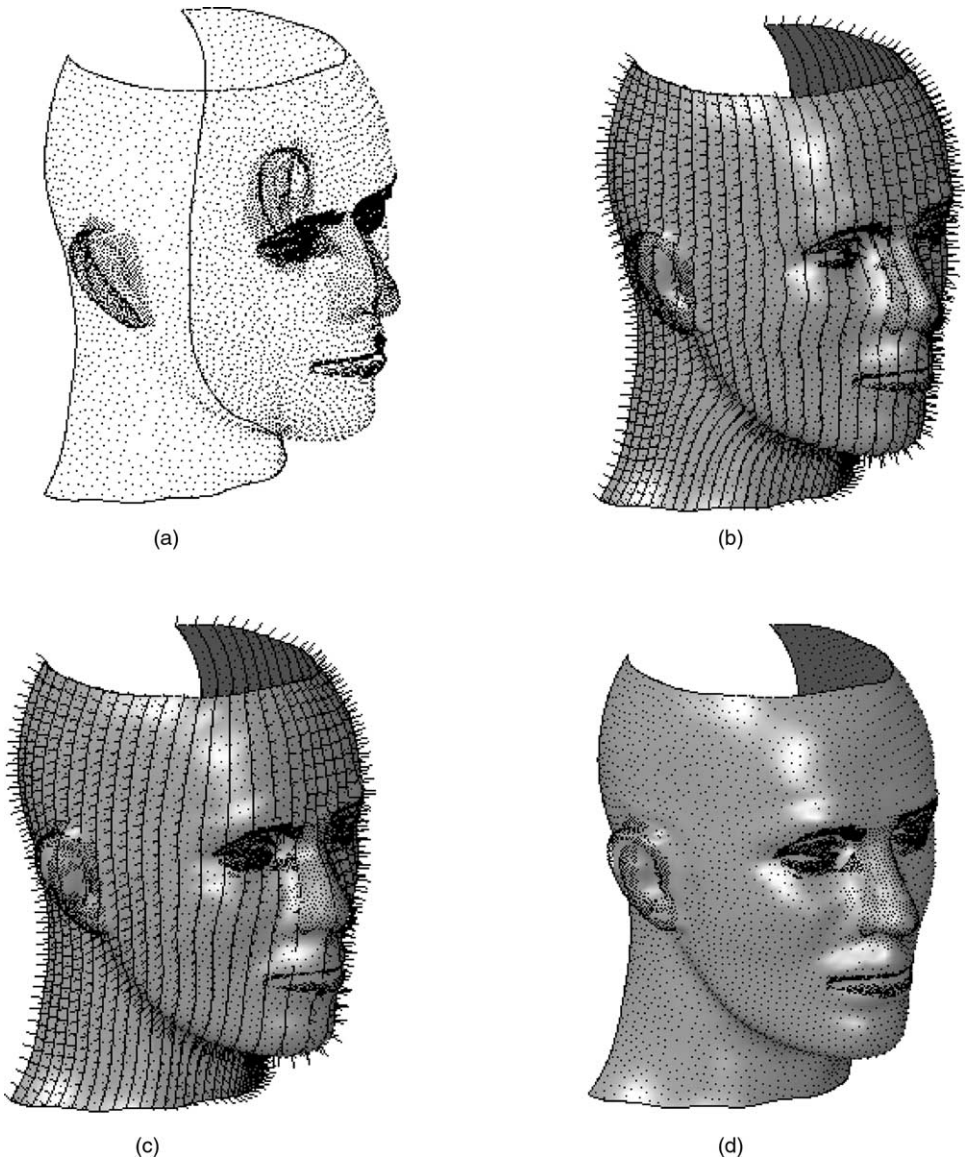

Fig. 8. Reverse engineering of the face ($N = 11,469$). (a) The cloud of points. (b) The final DBS (grid: 39×39) computed using the SSDE(*,*) operator and GE(*,*) for the last iteration. (c) The final DBS (grid: 39×39) computed without relaxation. The 'stretching' of the grid in areas near the noise and the forehead, and the 'shrinking' in the vicinity of the neck are apparent. (d) The final surface (39×39 control net) corresponding to the solution of the LSQ problem.

examples are developed using a generic implementation of the proposed method. In other words, the use of heuristics and enhancement techniques is kept to the minimum. For example, the number of grid section curves is increased iteratively without taking into account the areas with high curvature or large fitting errors. We followed this approach in order to derive uniform and comparable results for all the presented examples. In practice, the proposed algorithm could be fine tuned in order to be more efficient and effective in industrial scale utilization. Thus, in more elaborated approaches section curves should be inserted in predefined areas of the DBSes.

In all the presented examples, we initiated the algorithm setting the smoothing term κ (used in Eq. (25)) equal to 0.5. Then, we let the influence of the smoothing term fade out in the later iterations. In this way, unwanted crossovers or loops of the DBS can be avoided, especially in the beginning of the adaptation process. As the DBS comes closer to the cloud of points the smoothing term gets less importance. In every figure, we present the DBSes with the corresponding direction vectors at each vertex of the grid. Detailed results of all the examples are listed in Table 1, where the last column provides the time required to compute the parameters of the cloud of points after the construction of the final DBS.[3]

The overall process of reverse engineering the outer half of the shoe last (Fig. 2) is described in Fig. 5a–d.

[3] All the presented experiments have been conducted on a Pentium 4: 1.4 GHz, 256 MB RAM.

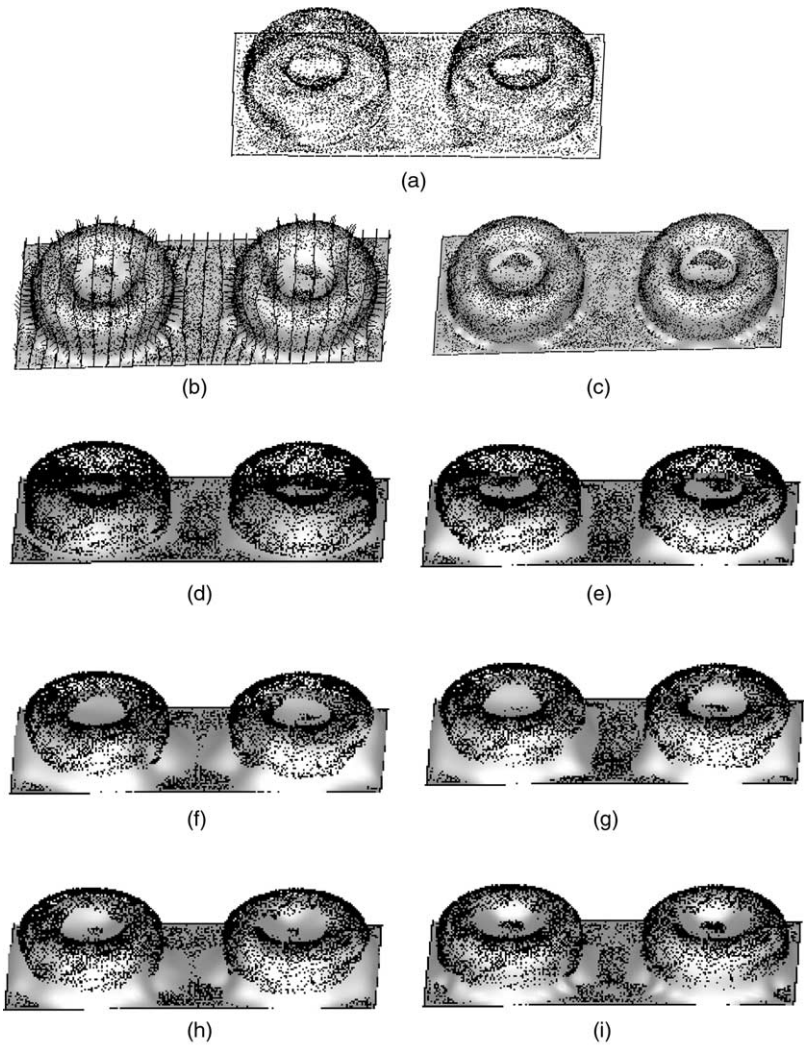

Fig. 9. Reverse engineering of the toy part ($N = 21,120$). (a) The cloud of points. (b) The final base surface (29×29) computed using the SSDE(*,*) operator and GE(*,*) for the last iteration. (c) The final surface (29×29 control net) corresponding to the solution of the LSQ problem. (d) Grid: 17×17. (e) Grid: 18×18. (f) Grid: 19×19. (g) Grid: 20×20. (h) Grid: 24×24. (i) Grid: 26×26.

Fig. 5a shows the initial DBS fitted to the four boundary curves, and the initial grid (12 rows by 12 columns) with the associated normal vectors. The initial DBS approximates roughly the fundamental geometry of the cloud of points. The application of the SSDE(*,*) operator to the initial grid is shown in Fig. 5b. The DBS is immediately improved since it is smoother and approximates the cloud of points more faithfully. The DBS is a tensor product (3,3)-degree B-spline surface fitted to the projected grid. Fig. 5c displays the final DBS which is a (3,3)-degree B-spline surface fitted to the final grid with 20 rows and 20 columns. Coarse areas of the cloud are approximated with almost flat regions and high-curvature areas are approximated with satisfying accuracy (see also Table 1). The final surface is a (3,3)-degree B-spline surface fitted using LSQ and the parameters obtained by orthogonally projecting the cloud of points onto the DBS (Fig. 5d).

Fig. 6a shows the cloud of points of a boot last with a sharp forepart ($N = 12,454$). During the iterative evolution of the DBS, crossovers between successive grid curve sections occurred (Fig. 6b). This problematic case was resolved through the evaluation of geodesic sections as it is shown in Fig. 6c allowing for the construction of the final valid DBS shown in Fig. 6d. Fig. 6e shows the (3,3)-degree B-spline surface fitted in the LSQ sense to the cloud of points using the parameters obtained by orthogonally projecting the cloud of points onto the DBS.

The cloud of points of a car body is shown in Fig. 7a. Usually a good segmentation should have divided this cloud into subsets where simpler surfaces could be fitted. Notice the ‘empty’ areas without points (car window seats) which hinder the fitting of a single surface significantly. Though, in order to illustrate the effectiveness of the proposed method, we attempted to fit a single DBS to the cloud of points. The proposed method succeeded to adapt a smooth DBS, which

approximates faithfully the fundamental geometry of the cloud of points (Fig. 7b). Furthermore, the smoothing functional (42) used for LSQ fitting overcomes the 'weak control points' problem [28] which is caused by the 'empty' areas and it could cause unpredictable results. The parts of the surface corresponding to that 'empty' areas are stretched due to the functional (42), which provides smoothness to the surface and guarantees that weak control points are handled appropriately (Fig. 7c).

Fig. 8a displays the cloud of points of a human face which is approximated by a DBS through a grid with 39 rows and 39 columns (Fig. 8b). The DBS has being expanded in curved areas such as the noise and the ears in order to capture the underlying geometry. The surface in Fig. 8b is constructed using the proposed relaxation technique resulting to an almost even distribution of the grid points between successive grid sections. On the other hand, Fig. 8c has been obtained without relaxation. The 'stretching' of the grid in areas near the noise and forehead and the 'shrinking' near the vicinity of the neck are apparent. In general, the 'stretching' and 'shrinking' of the grid may cause crossovers between successive grid sections during their projection onto the cloud of points and therefore, the grid should be relaxed at every iteration of the algorithm. The LSQ problem has been solved using QR factorization in 474.070 s (Table 1). However, Conjugate Gradient method proved much more efficient since it required only 157.467 s to solve the optimization problem giving in parallel slightly better results (Fig. 8d). In this example, the memory utilization for the formulation of the solution based on QR factorization incorporating smoothing functionals exceeded 340 MB, justifying the increased computational time. If extra smoothness is not required, then such a problem can be solved using QR factorization in a reasonable computational time.

Fig. 9a shows the cloud of points of a toy part. Again, under pragmatic circumstances, the cloud of points would be divided through segmentation into much simpler subsets. After 12 iterations of the proposed method, an adequate DBS has been computed and it is shown in Fig. 9b. Conjugate Gradient method gave a relatively fast and accurate solution to the final LSQ problem incorporating smoothing functionals (Fig. 9c). Figs. 9d–i show the adaptation of the DBS onto the cloud of points. The DBS captures the global shape of the cloud of points very fast, especially in the early stages of the adaptation process.

Another, test case with complicated geometry is the cloud of points of a shoe heel (Fig. 10a). The large difference between the width of heel base (top) and the width of heel bottom hinders significantly the development of the DBS, since a 'shrinking' of the grid in the bottom part and a 'stretching' at the upper part is unavoidable. The DBS is computed after eight iterations in 2.093 s and according to Table 1 approximates the fundamental geometry with high accuracy (Fig. 10b). The LSQ solution corresponds to the B-spline surface shown in Fig. 10c, which is computed in 6.272 s. A final example is shown in Fig. 11. It is actually a benchmark of the proposed parameterization method with large datasets (62.5×10^3 points), where the cloud of points of the torus quadrant has been

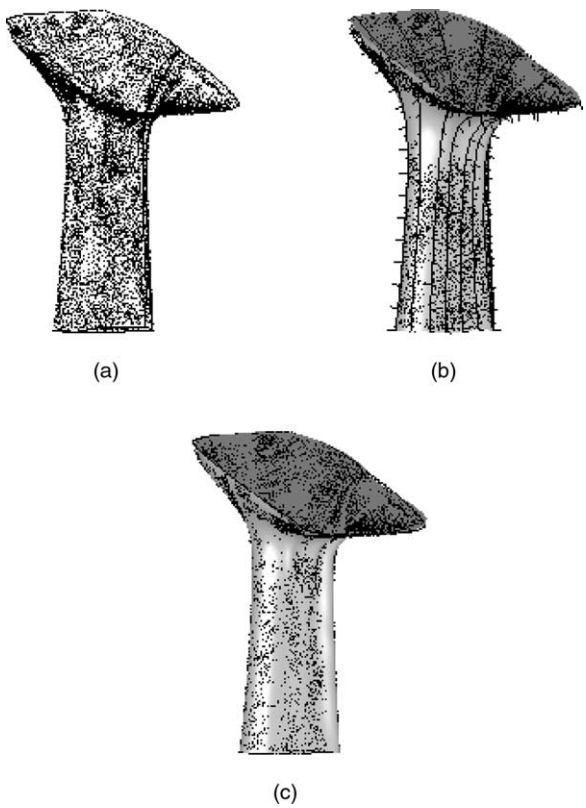

Fig. 10. Reverse engineering of the shoe heel ($N = 11,448$). (a) The cloud of points. (b) The final (grid: 14×14) computed using the SSDE(*,*) operator and GE(*,*) for the last iteration. (c) The final surface (14×14 control net) corresponding to the solution of the LSQ problem.

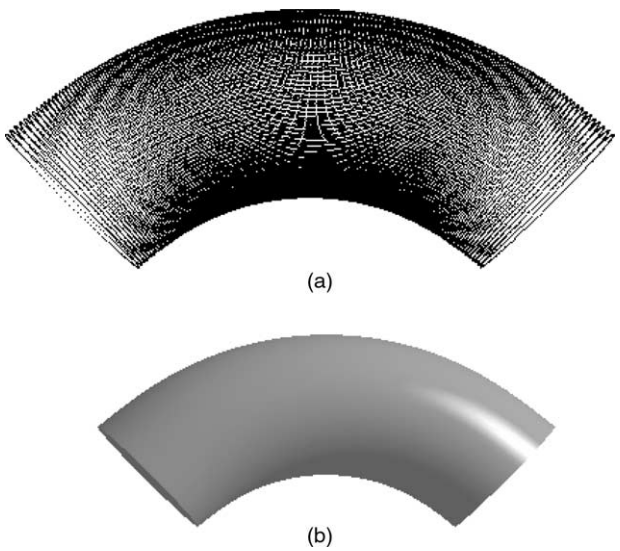

Fig. 11. Benchmarking the proposed parameterization method with large datasets ($N = 62.5K$). (a) The cloud of points. (b) The final DBS (17×17) computed using the SSDE(*,*) operator and GE(*,*) in the last iteration in 5.006 s.

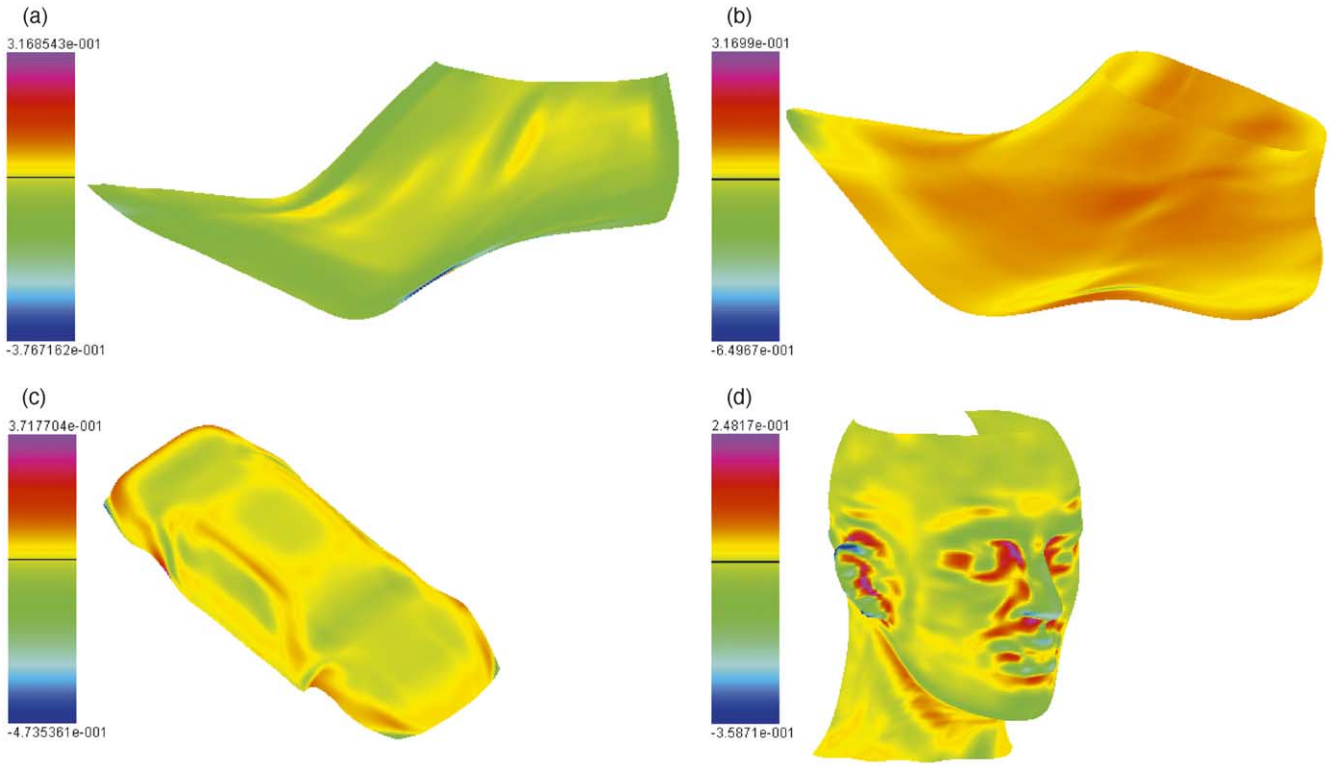

Fig. 12. Mean curvature plots of the DBSes computed for the (a) shoe last (Fig. 5), (b) boot last (Fig. 6), (c) car body and (d) human face.

algorithmically created. The DBS is computed after seven iterations of the proposed method in 5.006 s, providing ample justification concerning its efficiency with respect to the computational time.

Finally, in order to illustrate further the quality of the computed DBSes, four representative mean curvature plots are presented in Fig. 12 corresponding to the *shoe last* (Fig. 5), *boot last* (Fig. 6), *car* and *human face*. According to Fig. 12a, the computed DBS is smooth while an increase in the mean curvature near the featherline is due to the sharpness of the surface of the last in the corresponding areas. Fig. 12b verifies the smoothness of the computed DBS of the boot last surface. Again, an increase of the mean curvature is apparent near the vicinity of the featherline due to the geometry of the last surface. Fig. 12c displays the mean curvature of the car DBS which is a very smooth surface without wrinkles and without rapid changes in the mean curvature. This is also due to the higher sampling accuracy of the measured cloud of points. Finally, Fig. 12d shows the mean curvature plot of the DBS which corresponds to the human face. Although, there are some areas with increased mean curvature, this is due to the complexity of the fundamental shape of the cloud of points.

6. Conclusions

A new method for parameterizing clouds of unorganized points has been presented. The proposed method introduces the notion of Dynamic Base Surfaces, which are dynamically fitted to various types of geometries implied by the clouds of points. A large range of examples has been presented to illustrate the effectiveness and the efficiency of the proposed method. Even if segmentation has not been invoked in this paper, well-segmented data sets would be a lot easier to be processed with the proposed parameterization method. The implementation of the methods presented in this paper took place by following a generic approach and avoiding the incorporation of heuristics. Although this paper does not claim to solve the parameterization problem, it is our opinion that it provides the engineer/designer a set of tools for obtaining adequate parameters for randomly distributed points.

Many areas of future work and potential applications are envisaged. First of all, the novel idea of projecting points onto the cloud of points can be utilized to develop algorithms for direct 3D drawing on the cloud of points, which could result to a fully point-based CAD system. Also, the same technique can be tested for producing meshes of cloud of points. A preliminary research revealed that both areas are very promising. In addition, it seems reasonable to search for other weight functions in order to replace Eq. (10) with one which is more accurate or with a function that is computationally less expensive. Regarding, the overall algorithm: future research could be focused at the development of techniques for estimating more accurately the directions of projection. It seems possible to improve the projection accuracy by first creating a triangulation of

the cloud of points and then estimating the real normal directions at each vertex of the cloud of points. Finally, the application of the proposed parameterization method in curve fitting is straightforwards.

References

[1] Várady T, Martin RR, Cox J. Reverse engineering of geometric models: an introduction. Comput-Aided Des 1997;29(4):255–68.

[2] Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. Proc SIGGRAPH 95 1995;173–82.

[3] Eck M, Hadenfeld J. Local energy fairing of B-spline curves. In: Farin G, Hagen H, Noltemeier H, editors. Computing, vol. 10.; 1995.

[4] Floater MS. Parametrization and smooth approximation of surface triangulations. Comput-Aided Geom Des 1997;14(3):231–50.

[5] Floater MS. Parametric tilings and scattered data approximation. Int J Shape Model 1998;4:165–82.

[6] Greiner G, Hormann K. Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines. In: Le Méhauté A, Rabut C, Schumaker LL, editors. Surface fitting and multiresolution methods. Nashville: Vanderbilt University Press; 1997. p. 163–72.

[7] Hormann K, Greiner G. MIPS: an efficient global parameterization method. In: Laurent P-J, Sablonnière P, Schumaker LL, editors. Curve and surface design. Nashville: Vanderbilt University Press; 2000. p. 153–62.

[8] Hormann K, Greiner G. Hierarchical parameterization of triangulated surfaces. Proc Vis, Model Visualization 1999;219–26.

[9] Lévy B, Mallet JL. Non-distorted texture mapping for sheared triangulated meshes. Proc SIGGRAPH 98 1998;343–52.

[10] Ma W, Kruth JP. Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. Comput-Aided Des 1995;27(9):663–75.

[11] Maillot J, Yahia H, Verroust A. Interactive texture mapping. Proc SIGGRAPH 93 1993;27–34.

[12] Ma SD, Lin H. Optimal texture mapping. Proc EUROGRAPHICS 88 1988;421–8.

[13] Bennis C, Vezien J-M, Iglesias G. Piecewise surface flattening for non-distorted texture mapping. Comput Graph 1991;25(4):237–46.

[14] Azariadis PN, Aspragathos NA. Geodesic curvature preservation in surface flattening through constrained global optimization. Comput-Aided Des 2001;33(8):581–91.

[15] Azariadis PN, Aspragathos NA. On using planar developments to perform texture mapping on arbitrarily curved surfaces. Comput Graph 2000;24(4):539–54.

[16] Lee AWF, Sweldens W, Schröder P, Cowsar L, Dobkin D. MAPS: multiresolution adaptive parameterization of surfaces. Proc SIGGRAPH 98 1998;95–104.

[17] Floater MS, Reiners M. Meshless parameterization and surface reconstruction. Comput Aided Geom Des 2001;18(2):77–92.

[18] Hoppe H, DeRose T, DuChamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. Comput Graph 1992;26(2):71–8.

[19] Pottmann H, Leopoldseder S, Hofer M. Approximation with active B-spline curves and surfaces. Proc Pac Graph 02, IEEE Comput Soc 2002;8–25.

[20] Piegl LA, Tiller W. Parameterization for surface fitting in reverse engineering. Comput-Aided Des 2001;33:593–603.

[21] Zeid I. CAD/CAM theory and practice. New York: McGraw-Hill; 1991.

[22] Erikson C, Manocha D. GAPS: general and automatic polygonal simplification. Seventh ACM Symposium on Solid Modeling and Applications, Tutorial T4: Handling Large Geometric Datasets; 2002.

[23] Du Croz J, Mayes P, Radicati G. Factorization of band matrices using level-3 BLAS. Proceedings of CONPAR 90 VAPP IV. Lecture notes in computer science, Berlin: Springer; 1990. p. 222–231.

[24] Press W, Flannery B, Teukolsky S, Vetterling W. Numerical recipes in C. Cambridge, UK: Cambridge University Press; 1988.

[25] De Boor C. A practical guide to splines. New York: Springer; 1978.

[26] Piegl L, Tiller W. The NURBS book. Berlin/Heidelberg: Springer; 1997.

[27] Lee ETY. Choosing nodes in parametric curve interpolation. Comput-Aided Des 1989;21(6):363–70.

[28] Weiss V, Andor L, Renner G, Várady T. Advanced surface fitting techniques. Comput Aided Geom Des 2002;19(1):19–42.

[29] Lipschutz MM. Differential geometry. USA: McGraw-Hill; 1969.

[30] Dongarra JJ, Bunch JR, Moler CB, Stewart GW. LINPACK users' guide. Philadelphia: SIAM; 1979.

[31] Powell MJD. Restart procedures for the conjugate gradient method. Math Program 1977;12:241–54.

[32] Levenberg K. A method for the solution of certain problems in least squares. Q Appl Math 1944;2:164–8.

[33] Marquardt D. An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 1963;11:431–41.

Dr. Phillip N. Azariadis has received a mathematics degree from the Department of Mathematics (1990–1994) and a PhD from the Mechanical Engineering & Aeronautics Department (1995–1999) of the University of Patras, Greece. Currently his is instructor at the Department of Product & Systems Design Engineering of the University of the Aegean and also director of the Research & Technology Department of the Greek research institute ELKEDE Technology & Design Centre SA. His background and research activities are focused in the areas of Computer-Aided Design and Design for Manufacture, Reverse Engineering, Computer Graphics and Robotics. His work has been published in leading international scientific journals and conference proceedings.