Parameterization and Reconstruction from 3D Scattered Points Based on Neural Network and PDE Techniques

J. Barhak and A. Fischer, *Member*, *IEEE*

Abstract—Reverse engineering ordinarily uses laser scanners since they can sample 3D data quickly and accurately relative to other systems. These laser scanner systems, however, yield an enormous amount of irregular and scattered digitized point data that requires intensive reconstruction processing. Reconstruction of freeform objects consists of two main stages: 1) parameterization and 2) surface fitting. Selection of an appropriate parameterization is essential for topology reconstruction as well as surface fitness. Current parameterization methods have topological problems that lead to undesired surface fitting results, such as noisy self-intersecting surfaces. Such problems are particularly common with concave shapes whose parametric grid is self-intersecting, resulting in a fitted surface that considerably twists and changes its original shape. In such cases, other parameterization approaches should be used in order to guarantee non-self-intersecting behavior. The parameterization method described in this paper is based on two stages: 1) 2D initial parameterization and 2) 3D adaptive parameterization. Two methods were developed for the first stage: Partial Differential Equation (PDE) parameterization and neural network Self Organizing Maps (SOM) parameterization. PDE parameterization yields a parametric grid without self-intersections. Neural network SOM parameterization creates a grid where all the sampled points, not only the boundary points, affect the grid, leading to a uniform and smooth surface. In the second stage, a 3D base surface was created and then adaptively modified. To this end, the Gradient Descent Algorithm (GDA) and Random Surface Error Correction (RSEC), both of which are iterative surface fitting methods, were developed and implemented. The feasibility of the developed parameterization methods and fitting algorithms is demonstrated on several examples using sculptured free objects.

Index Terms—Reverse engineering, laser scanner, PDE parameterization, neural network SOM parameterization, surface reconstruction.

1 INTRODUCTION

Currently, the laser scanner is the most common method used in reverse engineering (RE) applications because it is fast and robust relative to other methods. Scanned data from a laser provides explicit 3D points from which a 3D model can be reconstructed [24]. The 3D data generated by 3D laser scanners is, however, not suitable for direct incorporation into CAD systems. Laser scanner systems yield an enormous amount of irregular and scattered digitized points. This data requires intensive processing in order to reconstruct the surface of the object while preserving its topology and shape. This problem of surface reconstruction is a nonlinear optimization problem that has been addressed in [1], [14], [18], [23]. The methods reviewed, however, were found to be inadequate due to the relatively large amount of memory resources consumed for data based on tens to hundreds of thousands of points. Other advanced methods [2], [12], [13] use geometric representations that are not standard in CAD systems, such as subdivision surfaces, triangular Bezier-Burnstein patches and manifold surfaces. Eck and Hoppe [4] achieve surface reconstruction using B-Spline surfaces. However, the mesh is nonregular and has problems with samples of varying densities.

Surface reconstruction in CAD systems has been divided into two main stages: 1) parameterization and 2) surface approximation. The parameterization stage is essential in order to establish the neighboring relations between the points in the surface's parametric domain, thus actually determining the topology, geometrical shape, and boundary of the surface. Selecting an appropriate parameterization is the key to obtaining good surface fitting. Different parameterization strategies can be used to assign parametric values: equally spaced, chordal, centripetal, geometric, and affinity invariant parameters [14]. These strategies are based on the assumption that the points are arranged in a grid and are therefore not suitable for scattered digitized points.

Methods that deal with scattered points use another strategy, which is based on projecting the points onto a parametric plane and calculating the parameterization with respect to the 2D projected boundary [14]. This 2D parameterization does not, however, always reflect the relative position among the 3D sampled points. As a result, the topology of the reconstructed surface is not preserved. Kruth and Ma [17] overcome this problem by proposing a 3D base surface parameterization. The main advantage of their method in contrast to other 2D methods is that the base surface is not necessarily a plane. It is, in fact, a rough approximation of the final fitted surface. First, an initial parametric base surface is determined. Then, the

• *The authors are with the CMSR Laboratory for Computer Graphics and CAD, Department of Mechanical Engineering, Technion, Haifa, Israel 32000. E-mail: {merjac,meranath}@techunix.technion.ac.il.*

Manuscript received 3 Sept. 1999; revised 12 July 2000; accepted 14 Sept. 2000.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 110540.

parameterization of each sampled point is determined according to the parametric value of the nearest point on the surface. A new surface is approximated according to the sampled points and their new parametric values. This approximated surface can be used as a base surface for the next iteration. This process is repeated until the base surface converges to the final surface, with the resolution details increasing adaptively in each iteration.

Projecting the sampled points onto the 3D base surface results in better matching between the sampled points and their parametric values. For general cases, however, a base surface may have concave boundaries that might lead to undesired results, such as a self-intersecting surface [6]. For example, a Coons Patch is commonly used as a parametric surface [14], [17]. The concave boundaries of a Coons surface sometimes produce a parameterization that maps points outside the boundary to the inside of the surface. Moreover, it may also connect the points in an erroneous topology, leading to a self-intersecting surface.

The self-intersection problem can be avoided if a proper curve network is created, as suggested in [17]. In such a case, determining the boundary curves is straightforward and can be achieved using vision techniques [7], [20]. However, determining the interior curves is difficult. Another solution to the self-intersection problem using conformal mapping is proposed by [3], but the model is nonlinear and time consuming.

In addition, preserving details throughout the fitting process is desirable. Kruth and Ma [17] deal with this issue by adjusting the knot vector of the surface. This final grid is highly dependent on the parametric orientation of the initial base surface. In general, the parametric orientation is determined primarily by the four corner points. This will lead to a problematic grid whose density will not reflect the density of the sampled points. Although this phenomenon does not always constitute a serious problem, it may result in a wasteful representation of the surface. This occurs as a result of bunching many control points in areas where point density is low. Therefore, proper initial point parameterization can significantly affect the initial base surface.

In this paper, we have focused on creating an initial base surface through an appropriate initial parameterization without segmenting the range image. The proposed base surface differs significantly from those proposed in other papers, where the initial base surface could have been self-intersecting [14], [17] or where it does not reflect the sampled density [3], [14]. In order to overcome these problems, we have developed two methods: PDE parameterization and neural network SOM parameterization.

The proposed PDE (partial differential equation) numerical method avoids iso-parametric curve intersections. The initial parameterization is calculated on a plane from which a 3D parametric-based surface is created. The process is based on solving a PDE Laplacian equation twice, once in the *u* direction and once in the *v* direction. As a result, a grid of 2D non-self-intersecting iso-curves is created.

The proposed neural network method is based on the Self-Organizing Map (SOM) suggested by Kohonen in [16]. Self-Organizing Maps are nodes represented by neurons, where each neuron contains information about the node coordinates and the neighborhood relations. The entire neural map is trained by the sample data in random order and is modified into the shape of the sampled points. The advantage of a SOM grid is that all the sampled points, not only the boundary points, participate in grid creation. Using SOM parameterization, it is possible to accurately detect the orientation of the parametric domain, which will reflect the density of the sampled points. The initial parameterization of the 3D sampled points is then determined and a proper 3D base surface is reconstructed.

In this paper, the base surface is represented by a B-Spline surface, defined as:

$$\mathbf{S}(\mathrm{u},\mathrm{v}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{B_i}(\mathrm{u}) \cdot \mathrm{B_j}(\mathrm{v}) \cdot \mathbf{V_{ij}}. \tag{1}$$

The surface is defined uniquely by its order k, an ordered set of n*m control points $\{V_{ij}\}$, and knot vectors $\{\mathrm{u}_1 \dots \mathrm{u}_{\mathrm{n+k}}\}$ and $\{\mathrm{v}_1 \dots \mathrm{v}_{\mathrm{m+k}}\}$ in the *u* and *v* direction, respectively. An adaptive process is then applied, the final parametric values of the digitized points are determined, and a 3D B-Spline surface is fitted.

The paper is organized as follows: Section 2 describes the reconstruction approach, Section 3 details the PDE parameterization method, and Section 4 specifies the neural network SOM parameterization method. Section 5 compares the PDE and SOM methods and discuses initial parameterization, while Section 6 describes base surface fitting methods. Section 7 discusses the adaptive base surface parameterization and Section 8 demonstrates the feasibility of the parameterization and fitting methods on sculptured surfaces. A summary and conclusions are given in Section 9.

2 The Reconstruction Approach

The stages of the proposed reconstruction method are as follows (see Fig. 1):

1. Projecting the sample points onto a given plane and extracting the 2D boundary.
2. Constructing a 2D parameterization grid by PDE parameterization or neural network SOM parameterization methods.
3. Parameterizing the sampled data according to the 2D parametric grid.
4. Creating an initial 3D parametric base surface using the 3D digitized points and their 2D parameterization that was calculated in stage 3.
5. Projecting the sampled points onto the 3D base surface and calculating their new parameterization.
6. Correcting the B-Spline surface according to the new parameterization of the 3D digitized points.
7. Using the resulting surface from the previous stage as a parametric base surface. Stage 5 is repeated until the surface approximation error satisfies a given tolerance.

Stages 1-4 deal with the initial parameterization, either PDE or neural network SOM, from which the initial base surface is calculated. Stages 5-7 adaptively improve the

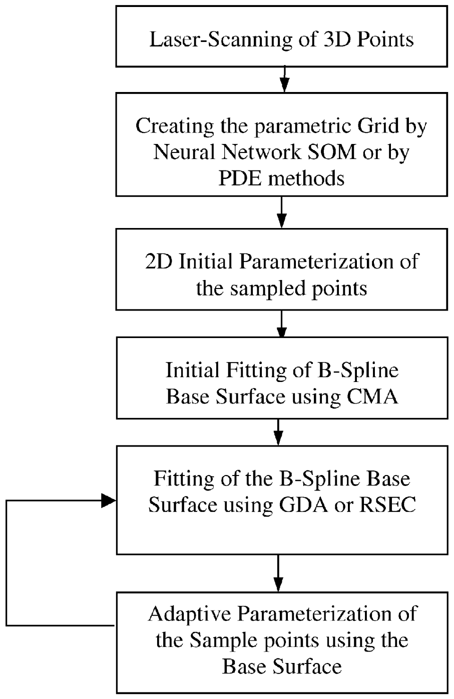

Laser-Scanning of 3D Points
Creating the parametric Grid by Neural Network SOM or by PDE methods
2D Initial Parameterization of the sampled points
Initial Fitting of B-Spline Base Surface using CMA
Fitting of the B-Spline Base Surface using GDA or RSEC
Adaptive Parameterization of the Sample points using the Base Surface

Fig. 1. General block diagram of the proposed approach.

reconstructed base surface until it converges to the sampled surface under a given tolerance.

Each of the above stages is described in detail in the following sections.

2.1 Criteria for a Parametric Grid

A good parametric grid is a grid that satisfies the following criteria:

1. Neighborhood relations—Digitized points that are neighbors will fall on the same or neighboring cells in the 2D grid.
2. Boundary conditions—Grid boundaries will coincide with the sample projection boundary.
3. Non-self-intersection—Two different grid cells do not share the same area. This criterion is satisfied if two different iso-curves, for u iso-value and v iso-value, do not intersect and do not split (Figs. 2 and 3) [9], [15].
4. Uniform parametric density—The projected digitized points will be distributed as uniformly as possible through the cells.

The PDE and neural network SOM methods both fulfill the first two grid requirement criteria. The PDE method also fulfills the criterion of non-self-intersection. Boundary subdivision in PDE has an impact upon fulfilling the fourth criterion, while the neural network SOM method fulfills the fourth criterion of uniform parametric density with a given tolerance.

3 PDE Implementation

3.1 Determining the Subboundaries of the PDE Grid

In order to determine the subboundaries of the PDE grid, the 3D points are first projected onto a plane perpendicular to the viewing direction of the laser. This direction is determined according to the laser position and the center of mass of the 3D digitized points. The 2D boundary is extracted using edge detection techniques.

Then, four boundaries are defined as initial conditions for the PDE solver and the 2D boundary is accordingly divided into four subboundaries. The subdivision can be applied according to the following criteria:

1. Arbitrary subdivision [14], leading to subboundaries that may be extremely short or long.
2. Curve criteria: Boundary curve division is usually applied according to length, or curvature criteria, or according to mixed criteria that integrate these criteria with a given rate [15]. The boundary subdivision criteria determine the behavior and orientation of the resulting parametric grid. When using the PDE parameterization method, the method for selecting the four boundary curves did not significantly affect the global surface accuracy and the PDE method always created a non-self-intersecting grid. Selecting the four boundary curves did

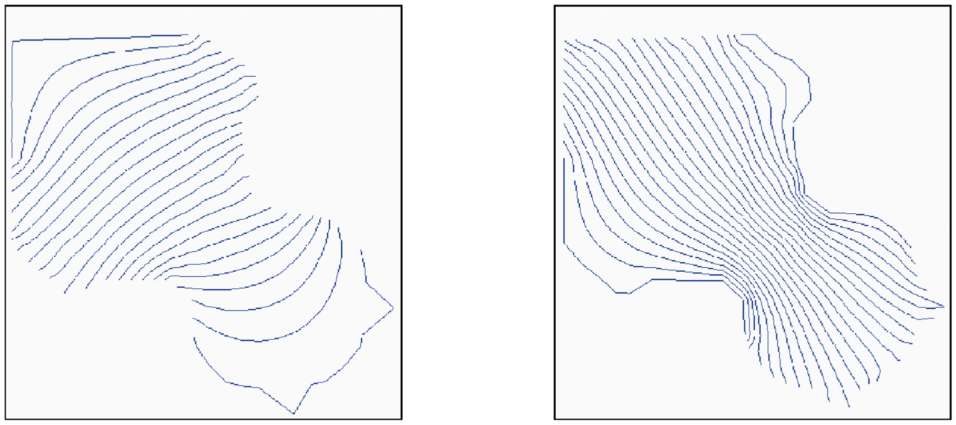


Fig. 2. The iso-curves of a projected object: (a) iso-u parametric curves; (b) iso-v parametric curves.

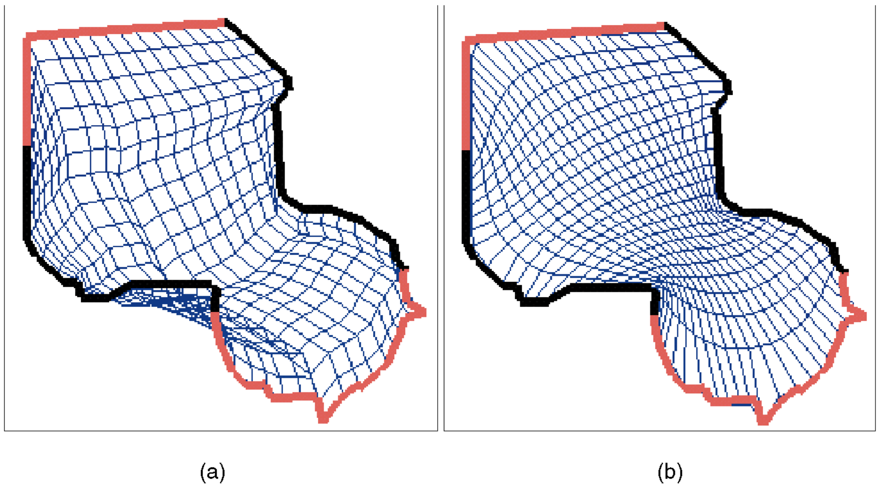


Fig. 3. 2D parameterization grid: (a) self-intersection grid; (b) PDE non-self-intersecting grid.

affect the parametric density of the grid and the smoothness and magnitude of surface oscillations (Figs. 10 and 11).

3. Area criteria: The sampled points are divided according to a uniform parametric density criterion created by using a neural network SOM. The method is defined in detail in Section 4. The cell area in such a grid is proportional to the point distribution, with small cells in high density regions and large cells in low density regions.

3.2 Calculating the Non-Self-Intersecting Parametric Grid

The Laplacian PDE was chosen because it produces non-self-intersecting iso-curves. Such a second order PDE formulation is frequently used as a model to formulate heat transfer problems and to compute iso-potential curves. Though a non-self-intersecting grid can also be computed by using conformal mapping [3], solving the second order Laplacian PDE is faster and more robust. Moreover, the angle preservation property of conformal mapping is not required since angles are distorted during the initial projection stage. Other possible solutions for the non-self-intersection problem are various smoothing operators described in [15]. These, however, can sometimes converge to a self-intersecting grid.

In the proposed method, the non-intersecting iso-parametric curves are computed by applying the boundary conditions on the four subboundaries and numerically solving the Laplacian PDE:

$$\frac{\partial^2 T}{\partial p^2} + \frac{\partial^2 T}{\partial q^2} = 0 \qquad p, q \in \Omega, \tag{2}$$

where Ω describes the area of the projected points and T is the potential function inside the domain Ω. A mapping from (p, q) to (u, v) is calculated by applying the nonlinear transformation from the domain Ω to the rectangular domain [(0, 1), (0, 1)]. The domain Ω is enclosed by the four boundary curves Γ_i i = 1, ..4 on which initial boundary conditions need to be imposed. The PDE equation is solved once for the u iso-curves and once in the v iso-curves with the following boundary conditions:

For the u iso-curves:

$$\begin{aligned} &T(p,q) = 0 && p,q \in \Gamma_1 \\ &T(p,q) = f_1(L_2(p,q)) && p,q \in \Gamma_2 \\ &T(p,q) = 1 && p,q \in \Gamma_3 \\ &T(p,q) = f_2(L_4(p,q)) && p,q \in \Gamma_4 \end{aligned} \tag{3}$$

and for the v iso-curves:

$$\begin{aligned} &T(p,q) = f_3(L_1(p,q)) && p,q \in \Gamma_1 \\ &T(p,q) = 0 && p,q \in \Gamma_2 \\ &T(p,q) = f_4(L_3(p,q)) && p,q \in \Gamma_3 \\ &T(p,q) = 1 && p,q \in \Gamma_4, \end{aligned} \tag{4}$$

where f_i are monotonically ascending/descending functions in the domain [0, 1] that are initially picked as linear functions. L_i are the arc length functions of the subboundary curves in the domain Γ_i, respectively.

In order to numerically solve the Laplacian PDE, an intermediate grid is created in the domain Ω with node spacing $(\Delta p, \Delta q)$ in the p and q directions, respectively. The solution is based on the following discrete finite difference scheme:

$$\begin{aligned} &T_{p+\Delta p,q} + T_{p-\Delta p,q} + T_{p,q+\Delta q} + T_{p,q-\Delta q} - 4T_{p,q} = 0 \\ &\forall \quad p,q \in \Omega. \end{aligned} \tag{5}$$

Writing the scheme for each node leads to a set of equations. The boundary conditions are obtained by substituting (5) at the boundary positions. In our method, the Gauss-Seidel iterative method was used to solve the equation set, where each node inside the boundary is assigned the average value of its adjacent nodes. The process converges to an equilibrium state. The method is described in detail in [10]. As a result, each node in the intermediate grid has a solution value. Then, the u and v iso-curves of the solution are extracted (Fig. 2) by applying a 2D version of the Marching Cube Algorithm [19]. Other methods based on neighborhood scanning can be applied.

The main advantage of using the Laplacian PDE is that the parametric grid is non-self-intersecting. The iso-parametric curves do not split; rather, they start from one side of the boundary and end at the opposite side, without crossing each other or the boundary (Fig. 2).

Next, the 2D grid is created by intersecting the u and v iso-parametric curves, producing a grid of $\mathrm{Nu} \times \mathrm{Nv}$ cells (Fig. 3). The time complexity of the entire algorithm is O(Np*Nq) per Gauss Seidel iteration. The number of iterations depends on the convergence tolerance.

The PDE parametric grid does not fulfill the fourth criterion of uniform parametric density. That is, the number of points in a cell is not uniformly distributed because boundary sample points, not internal points, affect the creation of the parametric grid. An adaptive grid correction method can be added, which: 1) changes the f_i functions shapes from linear to nonlinear monotonic functions that take into account the number of points per row/column; 2) resolves the PDE with the new boundary conditions. The process is repeated until convergence. Such a correction process was implemented and significantly improved parametric density in most cases. Since the orientation of the grid was not changed, however, the process was limited. On the other hand, the neural network SOM was able to produce uniform parametric density since it determines grid orientation. This approach is described in the following section.

4 Neural Network SOM Implementation

SOM grids are based on the self-organizing map proposed by Kohonen in [16]. The advantage of a SOM grid is that all the sampled points, not only the boundary points, participate in grid creation, producing a grid that is sensitive to sampling density. In this paper, the concept of Self-Organizing Maps (SOM) is proposed as the basis for the neural network SOM parameterization method.

SOM consist of grid nodes represented by neurons. Each neuron contains geometrical information about the node coordinates and neighborhood relations. The entire neural map is trained by the sampled data in random order and is modified into the shape of the projected sampled points. The connections between the neurons do not change during training; instead, the neurons get different geometrical values. Implementation of the basic SOM algorithm, which can be found in [16], is simple and quick. For the parameterization process, however, some modifications were required:

- Each neuron can be either mobile or static.
- Each neuron can be either active or inactive.
- The training function is both bubble and Gaussian in nature (see definition below).

4.1 SOM Algorithm

The SOM algorithm is described as follows:

1. Initialize the position of all the neurons {**N**(*i,j*) / *i* = *0..Nu, j* = *0..Nv*}. Neuron coordinates are usually set to random numbers within the bounding box of the sampled points.
2. Pick a sample $\mathbf{P}_s$ in random order.
3. Find (*i**, *j**) parametric coordinates of the winner neuron **N**(*i**, *j**) among all the *active* neurons. The winner neuron is the closest active neuron to the sample point $\mathbf{P}_s$.
4. Update the position of each *active* and *mobile* neuron in the neighborhood of the winner neuron toward the sample point (see update rules below).
5. Increment *s*; if *s* is smaller than the *run length*, go to Step 2, otherwise stop.

The neuron update rules are as follows:

- Projected sampled points $\mathbf{P}_s$ are represented in (*p*, *q*) coordinates.
- The weight of each sample point is W_s, where 1 is the default.
- Initial neighborhood radius is *R*.
- Initial learning rate is α_0.
- *Run length* of the algorithm is *L*.
- The bubble neighborhood of the winner neuron is defined as follows: All the **N**(*i*, *j*) that satisfies: $|i - i^*| + |j - j^*| < \frac{R \bullet (L-s)}{L}$.
- The neuron displacement is defined as: $\mathbf{N}(i,j) = \mathbf{N}(i,j) + hc_s \bullet (\mathbf{P_S} - \mathbf{N}(i,j))$, where hc_s is the Gaussian training function: $hc_s = \alpha_s \cdot e^{-\frac{(i-i^*)^2+(j-j^*)^2}{2}}$ and α_s is the learning rate for point *s*: $\alpha_s = 1 - (1 - \frac{\alpha_o \bullet (L-s)}{L})^{W_s}$.

The time complexity of the algorithm for practical cases is $\mathrm{O(L^*R^2 + L^*Nu^*Nv)}$.

The SOM algorithm creates a 2D SOM grid based on the projected points. This grid reflects the distribution of the projected sampled points. There is no known proof for preserving uniform 2D parametric density. A proof does exist for the 1D case and can be found in [16].

In the 2D case, the SOM suffers from boundary problems: The boundary of the neural network does not coincide with the boundary of the projected points.

4.2 Boundary Correction of the SOM Grid

The "boundary last" and "boundary first" algorithms are proposed in order to cope with the boundary problem.

4.2.1 *"Boundary Last" Method*

With the "boundary last" method, the SOM boundary neurons are drawn toward the projected points outside the grid by increasing the weight of the sampled points. Thus, the SOM is inflated iteratively like a balloon until very few instances fall outside its boundary. As a result, the boundary points converge to the SOM boundary (Fig. 4), thus overcoming the boundary problem in the original SOM algorithm. After the "boundary last" algorithm has been implemented, a small number of sample points still lie outside the grid; this, however, is later corrected in the parametric optimization stage.

This algorithm has the advantage of detecting the orientation of the grid without user intervention. Moreover, the "boundary last" method is suitable for samples with convex boundaries. It is easy to implement, but remains problematic for concave boundaries (Fig. 4).

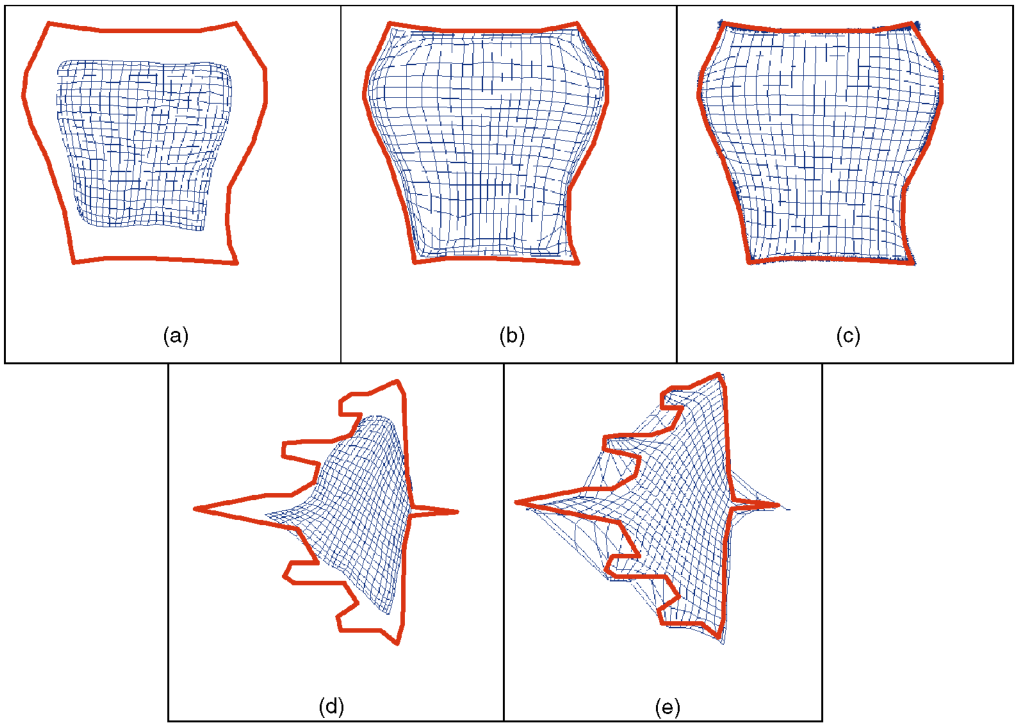


Fig. 4. Different phases in the SOM "Boundary last" method for a mask and an airplane: (a)-(c) mask model, (d)-(e) airplane model.

4.2.2 *"Boundary First" Method*

With the "boundary first" method, the 3D sampled points lying close to the boundary are detected via an edge detection algorithm. The SOM boundary neurons undergo training, while the SOM interior neurons do not participate in this training. The SOM internal neurons are then subsequently trained, while the SOM boundary neurons remain immobile. Fig. 5 shows the progress of the algorithm.

With the "boundary first" method, the boundary is trained separately from the whole grid. Thus, the boundary, with an intrinsic dimension of one, is trained by data having the same intrinsic dimension and is not affected by the data of the inner points. This sort of training yields a better boundary than the "boundary last" algorithm, especially for concave curves. As with the "boundary last" method, this method independently determines grid orientation and the four grid corner positions.

5 Initial 2D Parameterization

5.1 PDE and Neural Network SOM Comparison

Selecting the parameterization method depends on the range image as well as on the advantages and disadvantages of each particular method. The advantages of the PDE method over the neural networks SOM methods are as follows:

- The PDE method fulfills the non-self-intersection criterion. Neural network SOM methods do not always fulfill this criterion and are especially problematic for highly concave boundaries.
- Unlike the neural network SOM method, the PDE method is robust and not sensitive to training parameters.
- The PDE method is not order-dependent. It does, however, depend on the four boundary curves of the sampled points. The neural network SOM method may yield different solutions for the same data due to random ordering of the sampled points.

The advantages of the neural network SOM method over the PDE methods are as follows:

- The SOM method detects the orientation of the grid and the position of the four subboundaries.
- The SOM method takes into account all the sampled points, internal and boundary, and therefore strives toward uniform parametric distribution of the sampled points.

Thus, the neural network SOM method is suitable for convex boundary samples, while the PDE method is more appropriate for concave boundary samples. Even though the neural network SOM parameterization is characterized by randomness, it has proven reliable in all cases we tested.

5.2 2D Point Parameterization Using a Parametric Grid

After the parameterization grid has been constructed using either the PDE or the neural network SOM method, the sampled points can be parameterized by assigning (u, v) parametric values to each sampled point (x, y, z) as follows:

1. Each point $\mathbf{P}_s = (x, y, z)$ from the N sampled points is projected on the parametric grid plane to (X, Y) and its cell (i, j) is detected.

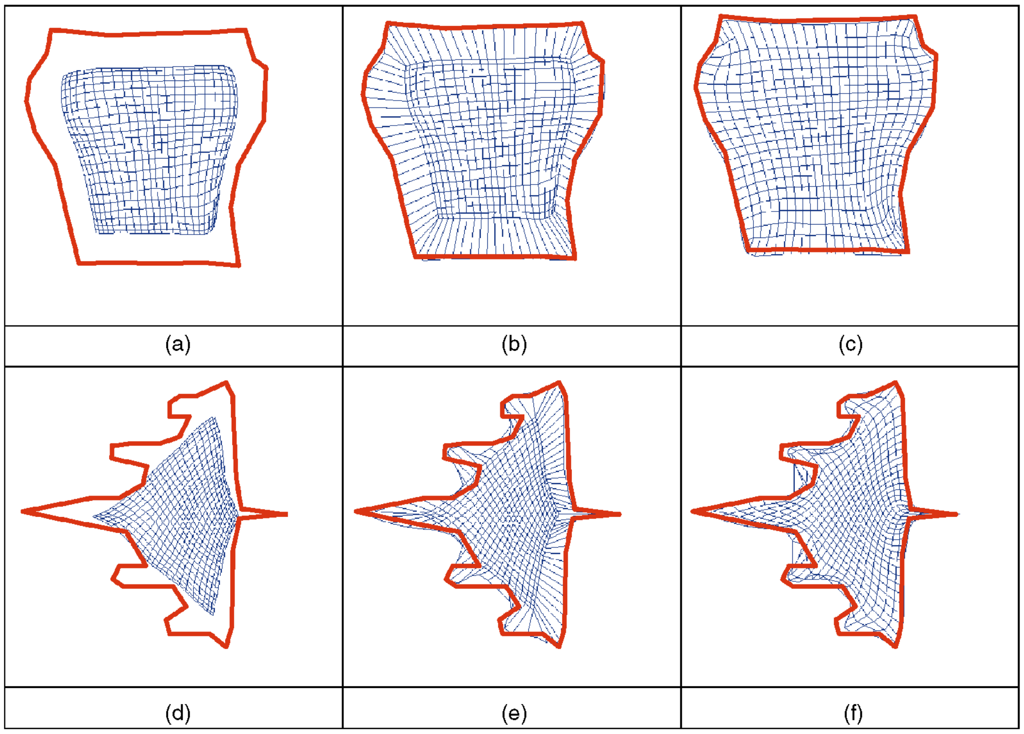


Fig. 5. Different phases in the SOM "boundary first" method applied on a mask sample and on an airplane: (a)-(c) mask model, (d)-(f) airplane model.

2. Each cell (i, j) is represented as a 2D Coons parametric surface. The value of (u, v) is determined by the relative location of (X, Y) on the grid cell (i, j) using the equations given in the Appendix.

The time complexity of the entire parameterization stage after grid creation is: O(N*Nu*Nv).

6 3D Base Surface Fitting

Kruth and Ma [17] have proposed an adaptive parameterization method using a base surface. The main advantage of this method in contrast to 2D methods is that the base surface is not necessarily a plane. It is, in fact, a first approximation of the final fitted surface. This initial base surface is problematic for concave shapes. The fitted surface can be noisy and self-intersecting or have improper orientation. In order to overcome the above problems, our proposed PDE and neural network SOM methods create a parameterization where the resulting base surface satisfies the following criteria:

1. The 3D boundary points are approximated by the surface boundary.
2. The base surface is non-self-intersecting (for PDE parameterization) or preserves details in densely sampled areas (for neural network SOM parameterization).
3. The approximation surface is smooth and does not suffer from oscillating behavior.

When data points are obtained empirically by measurements, approximation is more desirable than interpolation for overcoming noise errors and reducing computational complexity. For Cartesian and parametric surfaces, fitting methods are based on either:

1. Noniterative least square fitting (LSQ) accomplished by the singular value decomposition (SVD) technique.
2. Iterative fitting with a good initial approximation. In this paper, a good initial approximation is created by the control mesh approximation method (CMA). Correction is applied either by gradient descent algorithm (GDA) or by random surface error correction (RSEC).

6.1 LSQ Fitting of B-Spline Surfaces

The LSQ error function is defined as follows for a B-Spline surface:

$$\min \mathbf{E} = \sum_{s=1}^{N}\left[\mathbf{P_s} - \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u_s)\cdot B_j(v_s)\cdot \mathbf{V_{ij}}\right]^2, \qquad (6)$$

where $\mathrm{P_s}$ are the 3D scattered sampled points.

The noniterative LSQ method for approximating the 3D base surface was implemented using the SVD algorithm [22], which was chosen due to its robustness. This algorithm, however, failed to produce good results for some of our models (Fig. 7) due to a high condition number of the over-constrained equation set. Iterative techniques like GDA are less sensitive to such numerical problems, but have a trade-off in accuracy. In our environment, the algorithm was limited to small meshes of 15×15 control points. The time complexity was on the order of minutes and was quite slow relative to the CMA, GDA, and RSEC, as described in the next section. The lack of speed can be attributed to the fact that the SVD algorithm does not take advantage of matrix sparsity caused by local support of B-Spline surfaces.

6.2 Control Mesh Approximation (CMA) Method

In our proposed method, once the scattered measured points have been parameterized (Sections 3-5), the surface is fitted to the data points by iterative methods. These methods are faster than noniterative LSQ because they utilize the B-Spline local support characteristic and they are less prone to numerical problems. An initial approximation of the control polygon created by the CMA algorithm was implemented in order to shorten the fitting time. This approximation is not as accurate as was desired, but provides good initial conditions for the subsequent correction algorithms (GDA or RSEC).

The control polygon of the approximated B-Spline surface is calculated as follows:

1. A 2D $\mathrm{n} \times \mathrm{m}$ grid is created in the parametric domain *u, v* (not to be confused with the parametric grid).
2. For each 3D point $\mathbf{P_s}$ (s = 1..N) with normalized parameterization (*u, v*), a cell (i, j) is determined according to the following mapping: $\frac{i-1}{n} < u < \frac{i}{n}$ and $\frac{j-1}{m} < v < \frac{j}{m}$.
3. $\mathbf{V_{ij}}$ is calculated as the average of all the points $\mathbf{P_s}$ that belong to the same cell (i, j).
4. Given the control polygon and a knot vector, the surface is determined (1).

The complexity of the CMA algorithm is O(N*n*m).

Note that the surface does not interpolate the control polygon or the sampled points. However, the convex hull property guarantees convergence of the control mesh to the surface as control mesh density increases. This comes from the B-Spline characteristic in which the control-polygon is converged to the surface points during refinement. The maximum density of the control mesh is limited to the density of the sampled points. In addition, the approximation was improved by modifying the position of the control points. In this paper, two iterative control polygon correction methods were implemented:

1. Gradient Descent Algorithm (GDA), also known as steepest descent algorithm,
2. Random Surface Error Correction Algorithm (RSEC).

6.3 Control Mesh Correction Using the Gradient Descent Algorithm (GDA)

The GDA is a correction algorithm that was developed in order to overcome the inaccuracies in the CMA algorithm. It also corrects the boundary of the surface where the error can grow. This algorithm is based on an iterative LSQ technique. The surface error is defined in (7). The error for each control point $\mathbf{V_{ij}}$ is reduced by changing $\mathbf{V_{ij}}$ in small steps in the direction of the error gradient:

1. For a point $\mathbf{P_s}$ with parameters $(u_s.v_s)$, s = 1..N: Evaluate the distance of the point Ps from the surface $\mathrm{S}(u_s.v_s)$:

$$\mathbf{Err_s} = \mathbf{P_s} - \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u_s)\cdot B_j(v_s)\cdot \mathbf{V_{ij}}. \qquad (7)$$

2. Correct the surface control points as follows:

$$\mathbf{V_{ij}} = \mathbf{V_{ij}} + \eta \sum_{s=1}^{N} B_i(u_s)\cdot B_j(v_s)\mathbf{Err_s}, \qquad (8)$$

   where η determines the step size in the error gradient direction. The step size η can be controlled and usually determines the accuracy of the generated surface. In our examples, it was chosen empirically as $O(10^{-3} - 10^{-5})$, about five times greater than the convergence tolerance ε.
3. Step 2 is iterated until the change in all $\mathbf{V_{ij}}$ is smaller than a convergence tolerance ε.
4. The control points are reparameterized by projecting them onto a base surface (Section 7). The error is measured and stage 2 is repeated until the global error converges to a given convergence tolerance σ.

For computation simplification, each component ($x(u, v)$, $y(u, v)$ and $z(u, v)$) of the B-Spline is fitted separately. The value of η should be sufficiently small in order to converge to a local minimum and sufficiently large to converge fast. Other gradient methods that use the Hessian method (see [18]) can predict the size of η for optimal convergence, but this is beyond the scope of this work.

The GDA can be efficiently implemented while using the local support property of B-Splines with a complexity of $\mathrm{O}(N * k^2)$ per iteration, where *k* is the order of the B-Spline surface and N is the number of sampled points. The disadvantage of this algorithm is that the convergence is not guaranteed if η is not sufficiently small.

6.4 Random Surface Error Correction (RSEC)

In order to improve the control approximation, another algorithm (RSEC) is proposed. In this algorithm, correction is according to a local error rather than a global one. The random concept is again based on the SOM [16]. In this algorithm, the B-Spline surface is considered as a neural network and the control points are the neurons, but, instead of a winner neuron, we use a winner parameter and, instead of a Gaussian neighborhood training function, we use the B-Spline basis functions for control point position update.

The algorithm is based on the following stages:

1. Reorder the sample points randomly s = 1..N. Without reordering the sample points randomly, the algorithm will not converge to the desired surface.
2. For a point $\mathbf{P_s}$ with parameters $(u_s.v_s)$, s = 1..N:

   a. Evaluate the distance of the point $\mathbf{P_s}$ from the surface $\mathbf{S}(u_s.v_s)$:

$$\mathbf{Err_s} = \mathbf{P_s} - \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u_s)\cdot B_j(v_s)\cdot \mathbf{V_{ij}}. \qquad (9)$$

TABLE 1
Comparison of Approximation Methods for Mask

Method	Control Polygon size	Total Time	Repara-meterization Time	# of Iterations	Boundary Error	Surface Error
PDE Parameterization						
CMA	52x52	8 sec	-	1	1.202326	0.486495
GDA	52x52	62 min	~42 min	3	0.622919	0.226782
RSEC	52x52	65 min	~42 min	3	0.889290	0.168178
Neural network SOM “Boundary first” Parameterization						
CMA	52x52	8 sec	-	1	1.220696	0.441713
GDA	52x52	42 min	~28 min	2	0.831381	0.254475
RSEC	52x52	48 min	~28 min	2	1.121670	0.298538

# of sampled points = 32,626; bounding box size of the model = $15 \times 15 \times 13\ \mathrm{cm}^3$.

TABLE 2
Comparison of Approximation Methods for an Airplane

Method	Control Polygon size	Total Time	Repara-meterization Time	# of Iterations	Boundary Error	Surface Error
CMA	52x52	8 sec	-	1	1.038951	0.431203
GDA	52x52	340 min	~300 min	5	0.1222228	0.085556
RSEC	52x52	200 min	~160 min	3	0.3345834	0.077747

# of sampled points = 37,295; bounding box size of the model = $11 \times 7 \times 3\ \mathrm{cm}^3$.

b. Correct the surface control points as follows:

$$\mathbf{V_{ij}} = \mathbf{V_{ij}} + \eta B_i(u_s) \cdot B_j(v_s)\mathbf{Err}_s. \quad (10)$$

3. The control points are reparameterized by projecting them onto a base surface (Section 7). The error is measured and stage 2 is repeated until the global error converges to a given convergence tolerance σ.

The complexity of using the RSEC algorithm is O(N*n + N*m) if the degree of the B-Spline surface is considered as a constant. In this paper, all the surfaces were cubic B-Splines and the calculation time for a single point on the surface is considered to be a constant due to the local support property of B-Splines.

This RSEC yielded results that resembled those of the GDA in accuracy and time, as can be seen in Tables 1, 2, and 3. Despite the discrepancies between the two methods, there is no significant advantage to using one over the other in the models we used. As a rule of thumb, GDA should be used for correction due to its stability. If its results are found to be inadequate, however, RSEC should be considered as an alternative due to its local greediness and random nature.

7 Adaptive Base-Surface Parameterization

Once an approximated surface has been computed (Section 6), it can be used as the new parametric base surface for another iteration. This is accomplished by projecting the 3D sample points onto the base surface, as follows:

1. The initial point on the base surface closest to the sampled point is found by discretely searching the base surface parametric domain in $\Delta u, \Delta v$ intervals. The (u, v) parameters of the chosen point are assigned to the sampled point and retained for the next step.
2. These parameters are corrected iteratively. Equation (11) is used to calculate the change in the parameter values up to a given tolerance or until the number of iterations exceeds a given limit, implying diversion or very slow convergence. The development of the equation can be found in [20].

$$\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} |\mathbf{S_u}|^2 + \mathbf{r} \bullet \mathbf{S_{uu}} & \mathbf{S_u}\mathbf{S_v} + \mathbf{r} \bullet \mathbf{S_{uv}} \\ \mathbf{S_u} \bullet \mathbf{S_v} + \mathbf{r} \bullet \mathbf{S_{vu}} & |\mathbf{S_v}|^2 + \mathbf{r} \bullet \mathbf{S_{vv}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r} \bullet \mathbf{S_u} \\ \mathbf{r} \bullet \mathbf{S_v} \end{bmatrix}, \quad (11)$$

where

- $\mathbf{S_u}, \mathbf{S_v}, \mathbf{S_{uu}}, \mathbf{S_{vv}}, \mathbf{S_{uv}}$—the derivatives of the B-spline surface at the parameters (*u*, *v*).

TABLE 3
Comparison of Approximation Methods for a Real Knife

Method	Control Polygon size	Total Time	Reparameterization Time	# of Iterations	Boundary Error	Surface Error
PDE Parameterization						
CMA	52x52	22 sec	-	1	0.3553214	0.196429
GDA	52x52	340 min	~300 min	5	0.1222228	0.085556
RSEC	52x52	200 min	~160 min	3	0.3345834	0.077747
Neural network SOM "Boundary last" Parameterization						
CMA	52x52	22 sec	-	1	0.4404472	0.1332435
GDA	52x52	235 min	~160 min	3	0.1419249	0.0651551
RSEC	52x52	195 min	~160 min	3	0.3996531	0.0610476

# of sampled points = 106,714; bounding box size of the model = $17 \times 6 \times 3\ \mathrm{cm}^3$.

- **r**—The distance of the sample point from the point on the surface, that is, $\mathbf{r} = \mathbf{S} - \mathbf{P}_s$ at parameters (u, v). Step 1 may be omitted and the original parametric values of the point can be used as an initialization for Step 2. In this work, however, we found that Step 1 improves results, even though the process is very slow.

After the parametric correction has been applied, a new surface is fitted to the reparameterized points. The process is then repeated, usually converging adaptively to the desired surface. The final surface satisfies convergence tolerance while capturing the original shape of the sampled surface [17]. The complexity of the reparameterization stage is given by O(N*n*m) if the degree of the B-Spline surface and the boundary on number of iterations are considered a constant.

8 Examples

The development environment consisted of a CyberWare 3D laser scanner and a 350 Mhz Pentium II PC with 128M RAM memory and Windows NT 4.0 operating system. The software was written in the C Programming Language, using the Irit software package [5]. The feasibility of the proposed reconstruction process is demonstrated on several freeform objects: a knife, a mask, and an airplane. First, the objects were sampled at high density, $O(10^4)$ points. Next, the proposed PDE or neural network SOM parameterization methods were applied. Finally, several approximation surface methods were tested and compared:

1. SVD LSQ fitting method.
2. CMA initial surface fitting.
3. GDA method with adaptive reparameterization.
4. RSEC method with adaptive reparameterization.

The general SVD LSQ algorithm was limited to a 15×15 control polygon due to its inability to handle sparse matrices and a high number of conditions in the equation set. This small mesh size led to poor quality surfaces, and a comparison with other methods seems inappropriate due to different mesh sizes. However, this is the best result we were able to obtain using this method (see Fig. 7).

The CMA method was not limited in size and was faster than the SVD LSQ method for a given error. There were, however, two classes of errors: surface errors and boundary errors. The surface quality was high, while the boundary had convergence problems. In order to improve the error both on the surface and on the boundary, two correction methods were developed and tested: GDA and RSEC (Sections 6.3-6.4). The adaptive base surface parameterization was applied at each iteration. The GDA and RSEC improved the boundary results, but slowed down the process. Reparameterization of the base surface proved to be the time bottleneck; otherwise, the RSEC and GDA had the same time complexity. Parameterization time using PDE and neural network SOM was negligible compared to surface fitting time and did not take more than 2-3 minutes in all the examples.

Following is a comparison between the approximation methods, according to performance parameters: size, global time, reparameterization time, surface error, boundary error. This comparison was carried out on mask, knife, and airplane objects (see Figs. 6, 8, 9, 10 and Tables 1, 2, 3 for results). The mask object was sampled from a real object and demonstrates the ability of the methods to deal with complicated surfaces. The knife example was also sampled from a real object and illustrates the ability of the algorithms to deal accurately with a large number of sample points. The airplane example was synthetically calculated from an object and demonstrates the ability of the algorithms to deal with highly concave surfaces. In Tables 1, 2, 3, iteration numbers refer to adaptive reparameterization iterations (Section 7); all size measurements are in centimeters; errors presented are maximum errors.

The mask example shows two results: neural network SOM parameterization was fitted by GDA and the PDE parameterization was fitted using RSEC. Although the error

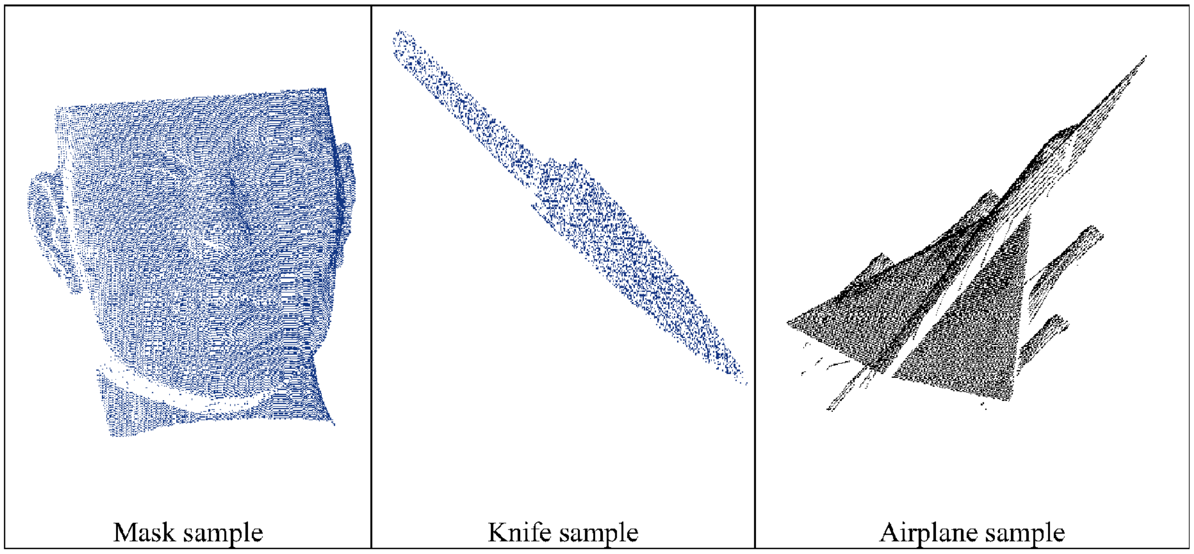


Fig. 6. Range images of a mask, knife, and plane model.

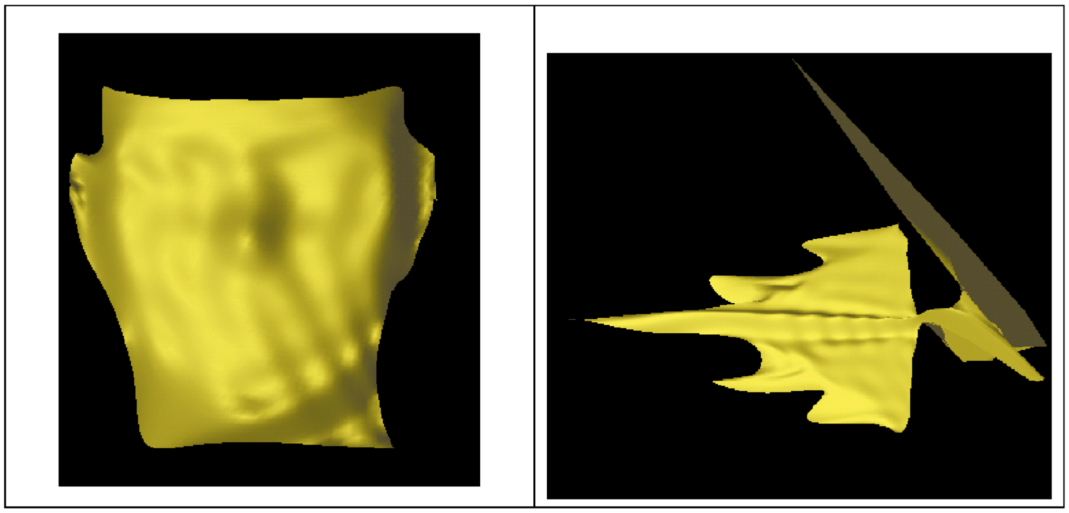


Fig. 7. Surface fitting by SVD LSQ of airplane and mask model.

is smaller in the PDE-RSEC example, the mask looks worse because of the local nature of RSEC. The RSEC is greedier than the GDA and works per sampled point at a time. The error is smaller numerically in this case; however, the surface seems less smooth.

Fig. 9 and Table 2 do not contain results from neural network SOM parameterization because SOM parameterization fails with concave boundaries, as can be seen in Figs. 4 and 5.

The knife sample demonstrates how the orientation of the grid can change qualitatively. Fig. 11 shows the parameterization grids and the point distribution of both grids. In the PDE parameterization, the subboundaries were picked arbitrarily (as in [14]). This causes a highly nonuniform point distribution in the parametric domain during initial parameterization. However, with the neural network SOM parameterization method, the points are distributed more fairly in the parametric domain so that the resulting surface is much smoother (see the shaded handle area), as can be seen in Fig. 10. This result suggests that using another criterion for boundary curve subdivision, as discussed in Section 3.1, would have produced better results for the PDE method. Moreover, there is a qualitative explanation. If the sampled points are distributed uniformly in the parametric domain, then each control point of the B-spline surface is influenced by a constant number of sampled points. As a result, areas with dense information will have many control points and will produce a surface with higher resolution in these areas. Areas with lower density will have fewer control points to support. The neural network SOM parameterization technique is sensitive to scanning density, as can be seen in Fig. 11.

The error convergence was illustrated on the different fitting methods relative to each object by error maps for PDE parameterization only (Fig. 12). The purple areas reflect regions with an error above a given threshold, i.e., either areas with very large errors or areas with no sampled points projected on them.

Although the total execution time may seem long, most of the time is spent in the optimization stage, which is the

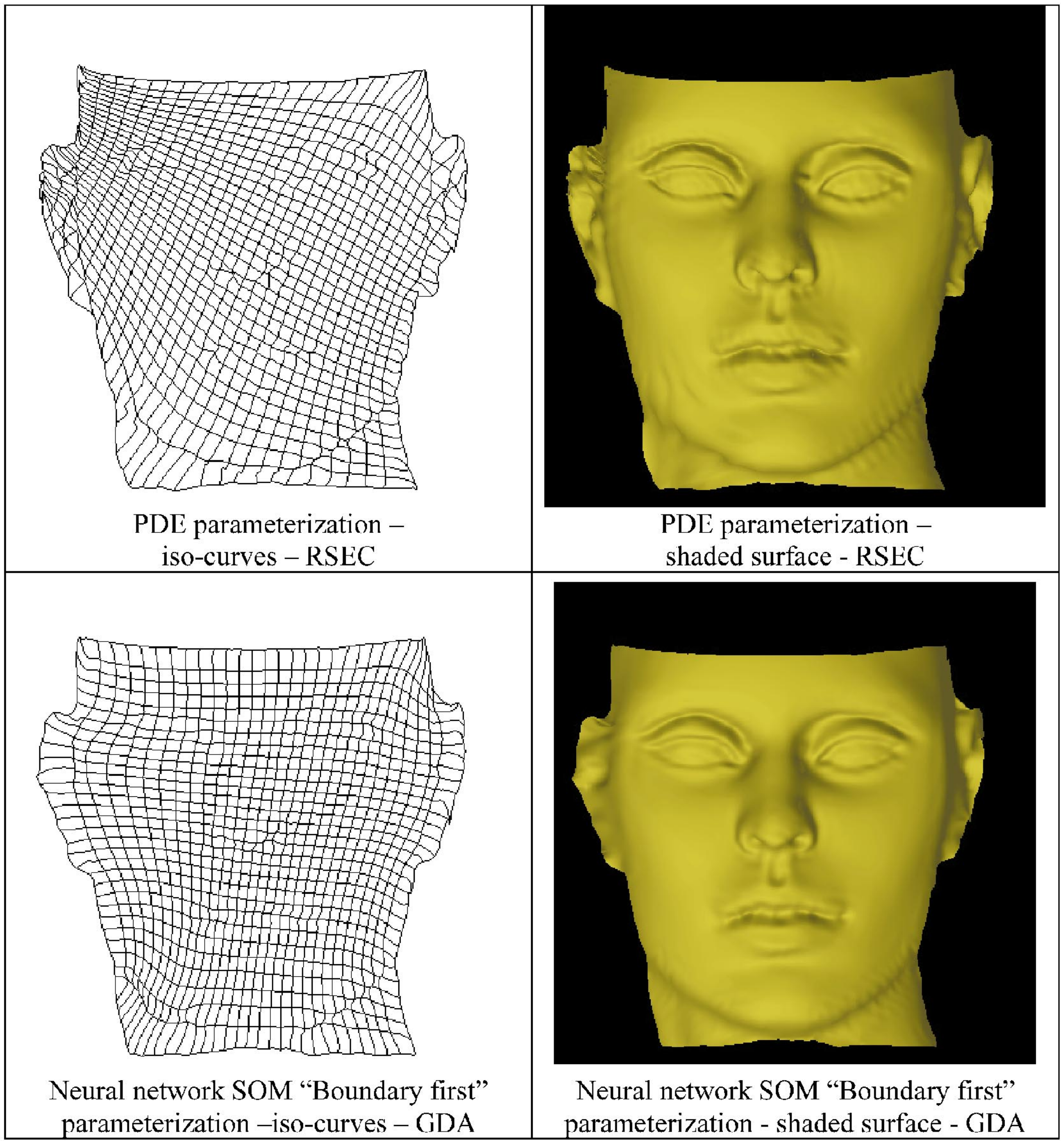

PDE parameterization – iso-curves – RSEC
PDE parameterization – shaded surface - RSEC
Neural network SOM "Boundary first" parameterization –iso-curves – GDA
Neural network SOM "Boundary first" parameterization - shaded surface - GDA

Fig. 8. Mask example.

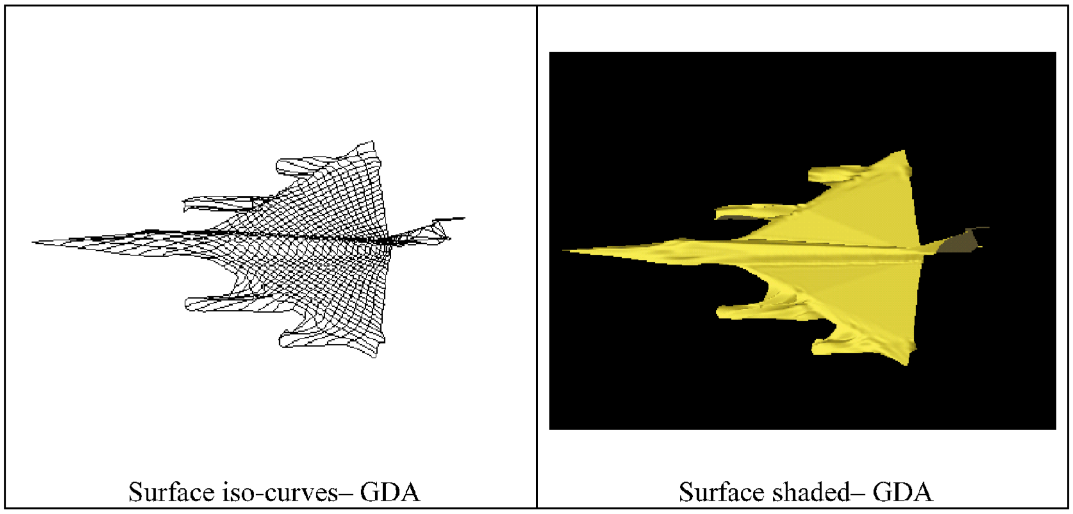


Fig. 9. Airplane example.

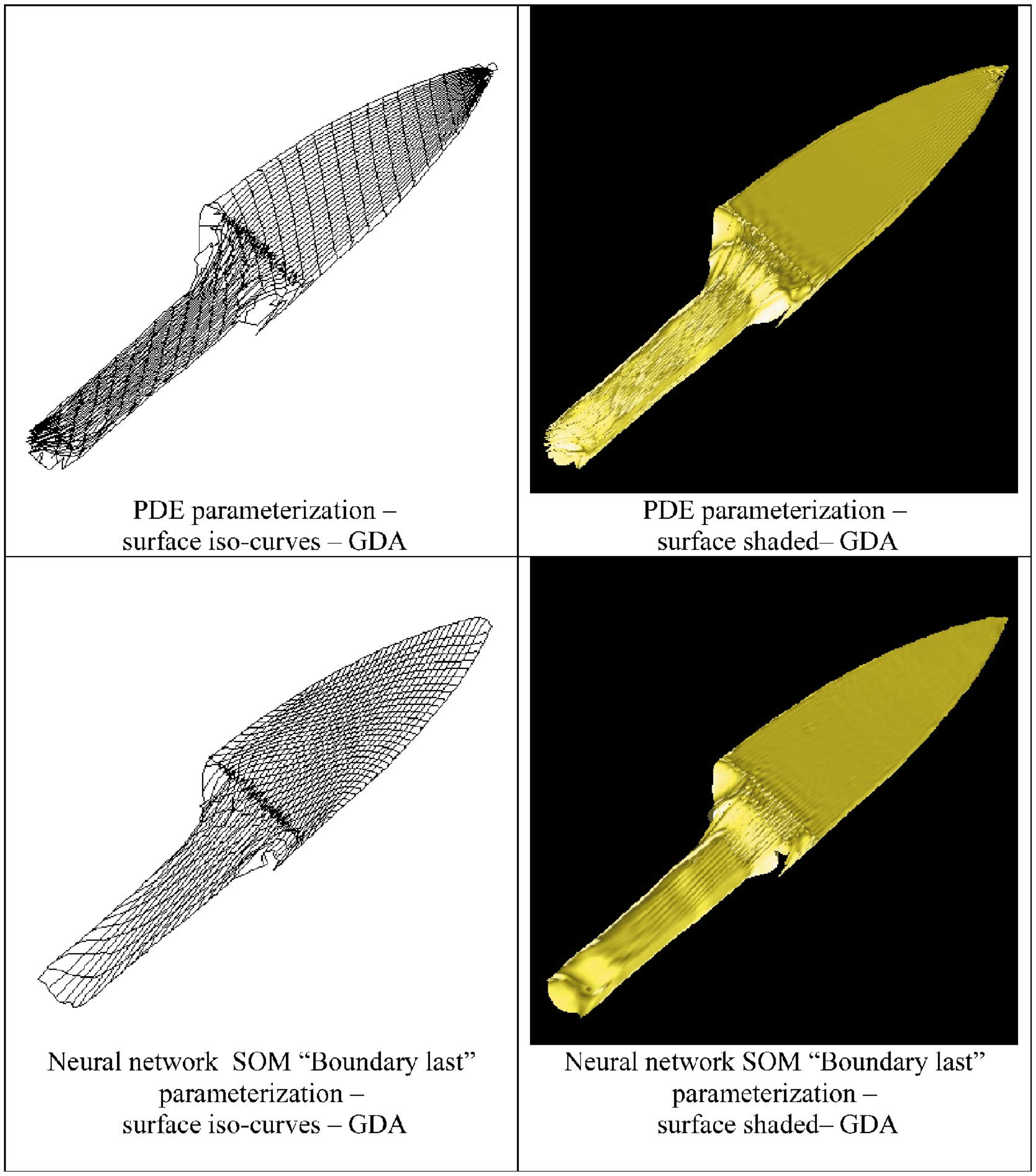


Fig. 10. Knife example.

time bottleneck. Until this stage, the fitted surface is a good approximation of the original surface and it would take no more than 10 minutes. However, it is not optimal. The execution time can be compared to other publications as follows: In [4], objects that were fitted from tens of thousands of points took hundreds of minutes on a 105Mhz HP 735 machine. In [13], objects that were fitted from tens of thousands of points took hours on a Dec-Alpha workstation. In [2], objects that were fitted from thousands of points took around 10 minutes on an SGI Indigo machine. The latter, however, has no parametric optimization stage.

9 Summary, Conclusions, and Future Work

In this paper, we proposed two new parameterization methods, neural network SOM and PDE. The PDE method produces a parametric grid without self-intersections, thus improving the topology of the fitted surface. The neural network SOM parameterization method leads to a uniform parametric density grid.

The approach outlined in this paper is based on the following stages: 1) reconstructing a parametric base surface based on neural network SOM or PDE methods; 2) adaptively calculating the parameterization of the 3D sampled points; and 3) fitting the surface by applying the CMA method and then correcting by GDA or RSEC. The feasibility of the parameterization algorithm is demonstrated on several examples using sculptured freeform objects.

As a result of applying the proposed method, the reconstruction process was improved. 1) Using the neural network SOM and the PDE methods, topology was improved, leading to smoother surfaces and better shape preservation without the self-intersecting anomaly. 2) The initial base surface was calculated very rapidly and preserved the shape characteristics of the reconstructed

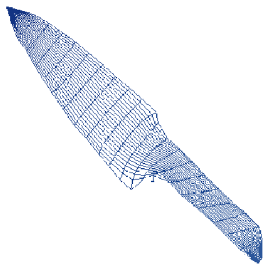
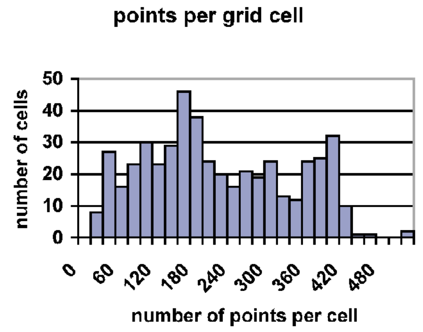
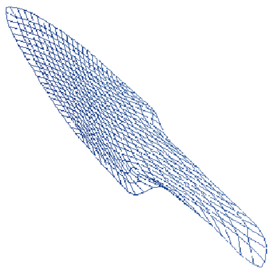
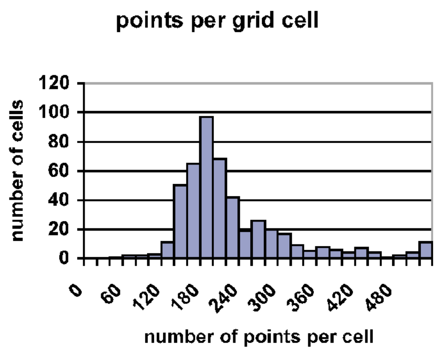
Parameterization method	2D parametric grid	Point density in grid histogram
PDE		
Neural Network SOM		

Fig. 11. Parameterization grid of the knife model using PDE and neural network SOM parameterization and a point distribution diagram.

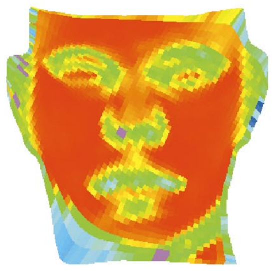
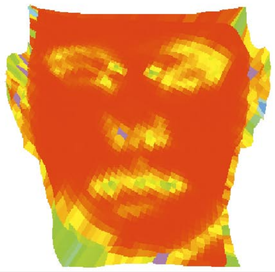
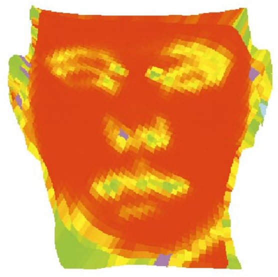
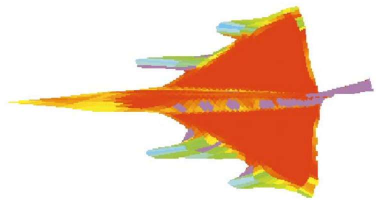
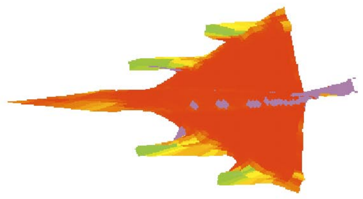
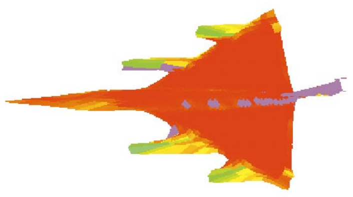
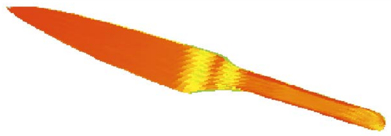
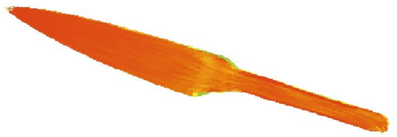
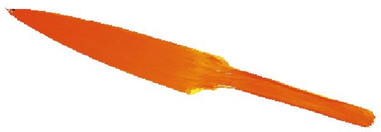
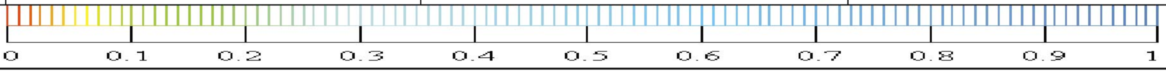
	Control mesh approximation	Gradient descent	Random surface error correction
Mask			
Air-plane			
Knife			
			

Fig. 12. Error analysis of the models by the four fitting methods (error is in centimeters). Note that purple areas are either areas with very large errors or areas with no sample points projected onto them.

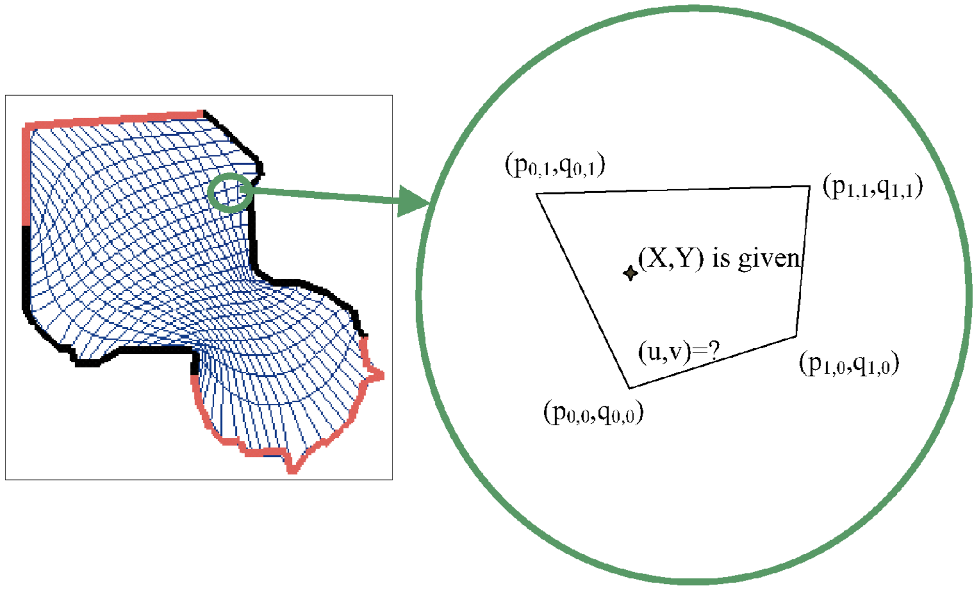


Fig. 13. A single parameterization grid cell and the parameterization points.

surface. 3) The CMA method enables fast approximation of the final surface. The method, however, suffers from accuracy problems on the boundary that were improved by the proposed correction methods (GDA, RSEC), which slowed down the process.

The feasibility of the method was demonstrated on several examples, mainly complex sculptured surfaces that were scanned at high density. This work related to only one sampling view.

In the future, surfaces from several views will be merged and blended in order to create a complete envelope of a volumetric object. Preliminary results already achieved in this direction are beyond the scope of this paper. Another improvement in future work will be using a conjugate gradient algorithm instead of GDA and training SOM neural networks directly on the 3D data. Using such an approach, the initial projection stage will be skipped.

Appendix

This appendix presents the formula (12-14) for deriving the parameterization of a point (X, Y) that falls in a cell (i, j) on the parameterization grid (Fig. 13). More details are given in [6].

$$\begin{aligned} u &= \frac{i+\Delta u}{Nu} \qquad i = 0\ldots Nu-1 \\ v &= \frac{j+\Delta v}{Nv} \qquad j = 0\ldots Nv-1 \end{aligned} \tag{12}$$

$$\Delta u = -\frac{p_{0,0} - \Delta v p_{0,0} + \Delta v p_{0,1} - X}{-p_{0,0} + p_{1,0} + \Delta v p_{0,0} - \Delta v p_{1,0} - \Delta v p_{0,1} + \Delta v p_{1,1}} \tag{13}$$

$$A\Delta v^2 + B\Delta v + C = 0 \qquad where$$
$$\begin{cases} A = q_{0,0}p_{1,0} - q_{0,0}p_{1,1} - q_{1,0}p_{0,0} + q_{1,0}p_{0,1} - q_{0,1}p_{1,0} + q_{0,1}p_{1,1} \\ \qquad +q_{1,1}p_{0,0} - q_{1,1}p_{0,1} \\ B = q_{0,0}p_{1,1} + 2q_{1,0}p_{0,0} - 2q_{0,0}p_{1,0} - q_{1,1}p_{0,0} + q_{1,1}X - Yp_{0,0} \\ \qquad +Yp_{1,0} + Yx_{0,1} + q_{0,0}X - y_{1,0}X - q_{1,0}p_{0,1} + q_{0,1}p_{1,0} \\ \qquad -q_{0.1}X - Yp_{1,1} \\ C = q_{0,0}p_{1,0} - q_{0,0}X - q_{1,0}p_{0,0} + q_{1,0}X - Yp_{1,0} + Yp_{0,0}. \end{cases} \tag{14}$$

Acknowledgments

This research has been supported in part by the Israeli Ministry of Science and the Korean Ministry of Science and Technology as part of the Bi-National Israel-Korea Project No. 6121-1-99. Additional funding was provided by the Anna & Leo Rosner Fellowship.

References

[1] M. Alhanaty and M. Bercovier, "Curve and Surface Fitting and Design by Optimal Control Methods," Technical Report No. 99-29 Leibniz Center, Hebrew Univ., 1999.

[2] C. Bajaj, F. Bernardini, and G. Xu, "Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans," *Proc. SIGGRAPH*, pp. 109-118, 1995.

[3] I.O. Dellahy, B. Shariat, and D. Vandorpe, "Conformal Mapping for the Parameterization of Surfaces to fit Range Data," *Product Modeling for Computer Integrated Design and Manufacture*, pp. 263-272, Chapman & Hall, 1997.

[4] M. Eck and H. Hoppe, "Automatic Reconstruction of B-Spine Surfaces of Arbitrary Topological Type," *Proc. SIGGRAPH*, pp. 325-334, 1996.

[5] G. Elber, "The Irit Solid Modeler Version 7.0," http://www.cs.technion.ac.il, 1996.

[6] A. Fischer and J. Barhak, "B-Spline Surface Reconstruction from a Single Range Image Using Parameterization Based on PDE Solution," TME Report-461, Technion Israel Inst. of Technology, 1999.

[7] A. Fischer, A. Manor, and J. Barhak, "Adaptive Parameterization for Reconstruction of 3D Freeform Objects from Laser-Scanned Data," *Proc. IEEE Pacific Graphics '99*, Oct. 1999.

[8] A. Fischer and S. Park, "Remote Sensing and LOD Modeling for Manufacturing Products," *Int'l J. Advanced Manufacturing Technology,* 1998.

[9] P.L. George, *Automatic Mesh Generation—Application to Finite Element Methods.* Wiley, 1991.

[10] C.F. Gerald and P.O. Wheatley, *Applied Numerical Analysis.* Addison-Wesley, 1989.

[11] P. Gu and X. Yuan, "Neural Network Approach to the Reconstruction of Freeform Surfaces for Reverse Engineering," *Computer-Aided Design,* vol. 27, no. 1, pp. 59-69, 1995.

[12] B. Guo, "Surface Reconstruction from Points to Splines," *Computer-Aided Design,* vol. 29, no. 4, pp. 269-277, 1997.

[13] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, J. Hubert, J. McDonald, J. Schweitzer, and S. Werner, "Piecewise Smooth Surface Reconstruction," *Proc. SIGGRAPH,* pp. 295-301, 1994.

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

[15] R.E. Jones, "QMESH: Self-Organizing Mesh Generator Program," Report SLA-73-1088, Sandia Labs, 1974.

[16] T. Kohonen, *Self-Organizing Maps.* Springer, 1997.

[17] J.P. Kruth and W. Ma, "Parameterization of Randomly Measured Points for Least Squares Fitting of B-Spline Curves and Surfaces," *Computer-Aided Design,* vol. 27, no. 9, pp. 663-675, 1995.

[18] P. Laurent-Gengoux and M. Mekhilef, "Optimization of a NURBS Representation," *Computer-Aided Design,* vol. 25, no. 11, pp. 699-710, 1993.

[19] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM Computer Graphics,* vol. 21, pp. 163-169, 1987.

[20] A. Manor and A. Fischer, "Reverse Engineering of 3D Models Based on Image Processing and 3D Scanning Techniques," *Proc. Sixth IFIP WG5.2 Workshop Geometric Modeling: Fundamentals and Applications,* Dec. 1998.

[21] L. Piegl and W. Tiller, *The Nurbs Book.* Springer, 1997.

[22] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes.* Cambridge Univ. Press, 1989.

[23] T. Speer, M. Kuppe, and J. Hoschek, "Global Reparameterization for Curve Approximation," *Computer-Aided Geometric Design,* vol. 15, pp. 869-877, 1998.

[24] T. Varady, R.R. Martin, and J. Cox, "Reverse Engineering of Geometric Models—An Introduction," *Computer-Aided Design,* vol. 29, no. 4, pp. 255-268, 1997.

Jacob Barhak received a BSc degree in mechanical engineering and a BA degree in computer science from the Technion, Israel Institute of Technology (1993). From 1993 to 1998, he worked in the software industry in the field of information technology. He is currently pursuing a PhD degree in mechanical engineering at the Technion. His main interests include reverse engineering, neural networks, and geometrical modeling.

Anath Fischer received a BSc degree (1985) in computer science and MSc (1987) and PhD degrees (1992) in mechanical engineering from the Technion. She is currently (since 1994) a faculty member in the Department of Mechanical Engineering, Technion-Israel Institute of Technology. From 1992 to 1994, she was a postdoctoral associate in the Injection Molding Program, Department of Mechanical Engineering, Cornell University, Ithaca, New York. Her main interests include geometric modeling for CAD/CAM, physical-based systems, reverse engineering, and rapid prototyping. She is a member of the IEEE.