NUMERICAL CONTROL OF KOHONEN NEURAL NETWORK FOR SCATTERED DATA APPROXIMATION

MIKLÓS HOFFMANN

Abstract. Surface reconstruction from scattered data using Kohonen neural network is presented in this paper. The network produces a topologically predefined grid from the unordered data which can be applied as a rough approximation of the input set or as a base surface for further process. The quality and computing time of the approximation can be controlled by numerical parameters. As a further application, ruled surface is produced from a set of unordered lines by the network.

1. Introduction

Surface reconstruction from a set of unorganized spatial points is one of the central problems in computer aided design. In many applications, like ship and car design creating a surface from scattered data is a frequently applied technique. One can find numerous methods for approximation or interpolation of scattered data or updating existing surfaces by scattered points using space warping, NURBS, subdivision or algebraic surfaces (see [18] and references therein for a general overview of the problem).

Different problems require different types of surfaces. Throughout this paper we will apply B-spline surface, which is a de facto standard description method in computer aided geometric design. For the sake of simplicity generally third order surfaces are used.

Definition 1. *The recursive function N_j^k given by the equations*

$$N_j^1(u) = \begin{cases} 1 & \text{if } u \in [u_j, u_{j+1}), \\ 0 & \text{otherwise} \end{cases}$$
$$N_j^k(u) = \frac{u-u_j}{u_{j+k-1}-u_j} N_j^{k-1}(u) + \frac{u_{j+k}-u}{u_{j+k}-u_{j+1}} N_{j+1}^{k-1}(u)$$

is called normalized B-spline basis function of order k. The numbers $u_j \le u_{j+1} \in \mathbb{R}$ are called knot values or simply knots, and $0/0 \doteq 0$ by definition.

Definition 2. *The surface $\mathbf{s}(u,v)$ defined by*

$$\mathbf{s}(u,v) = \sum_{q=0}^{n} \sum_{r=0}^{m} N_q^k(u) N_r^l(v) \mathbf{D}_{q,r}, \quad u \in [u_{k-1}, u_{n+1}], \quad v \in [v_{k-1}, v_{m+1}]$$

is called B-spline surface of order (k,l), where $N_q^k(u)$ and $N_r^l(u)$ are normalized B-spline basis functions, for the evaluation of which the knots $u_0, u_1, \ldots, u_{n+k}$ and $v_0, v_1, \ldots, v_{m+l}$ are necessary, respectively. The points $\mathbf{D}_{i,j}$ are

2000 *Mathematics Subject Classification.* 68U07, 65D17, 68T20.

Key words and phrases. surface reconstruction, Kohonen neural network, self-organizing map, B-spline surface.

called control points, while the mesh formed by these points is called control mesh.

As one can see, B-spline surface is uniquely given by its order, knot values and control points, which latter ones form a topologically quadrilateral mesh. In surface reconstruction problems the input is a set of unorganized points, thus the order, the knots and the control points are all unknowns. The overall aim of reconstruction methods is to determine these values where the basic strategy is the following (see [20] for overview):

(1) Fix the order (k, l), the number of control points (n, m) and the knot values u_i, v_j.
(2) Assign a pair of parameters (u_r, v_r) to each scattered point $\mathbf{P}_r$.
(3) Solve the system $\mathbf{s}(u_r, v_r) = \mathbf{P}_r$ or minimize $\sum_r \|\mathbf{s}(u_r, v_r) - \mathbf{P}_r\|^2$.

In terms of B-spline surfaces the crucial point of this strategy is step 2, which is frequently referred as the parametrization of the given data. Parametrization is the way how to assign parameter values to each point, where normally several restrictions and assumptions are introduced. One can try to consider the assigned values as unknown parameters in an optimization problem, but for large amount of data this approach leads to a complex non-linear system with several unknowns (see also [18]). At the recently developed base surface method data points are projected onto a predefined parametric surface to find the corresponding parameter value (c.f. [15], [16]). This technique can work well for certain type of data, but there are several conditions in terms of creating the base surface and the projection has to be a function, i.e. no overlapping allowed. Sometimes it is quite difficult to find a base surface which satisfies all the conditions.

This paper is devoted to the neural network approach of scattered data fitting. The earliest approaches of surface reconstruction by Kohonen self-organizing neural network can be found in the author's previous works ([7], [8], [19]) and Yu's paper [21]. Later on similar methods in different contexts have been developed in the recent papers of Barhak *et. al.* ([2], [3], [14]), Echevarría *et. al.* ([4]), Ivrissimtzis *et. al.* ([9], [10], [11]), and Knopf and Sangole ([13]). In this method we create different types of B-spline surfaces by Kohonen neural network in an iterative way. The quality of approximation can be controlled by the number of iteration steps and by other numerical parameters. The obtained surface can be used as a coarse approximant of the scattered data set or as a base surface for further reconstruction process. This way one can create base surfaces for more general types of data sets than by earlier methods. In this paper the method is also applied for creating 3D ruled surfaces which are of great importance in computer aided manufactory.

In the next section we give a brief introduction to Kohonen neural network and the network will be applied to create base surface for scattered points. In section 3 the numerical control of the network is discussed. In section 4 similar technique will be introduced for creating ruled surface from a set of unordered rulers. Conclusion and directions of further research close the paper.

2. The Kohonen neural network and its application

Artificial neural networks are widely used in computer science and its applications thus there are several types of networks for specific problems. A neural network consists of numerous computational elements (neurons or nodes), highly interconnected to each other. A weight is associated to every connection. Normally nodes are arranged into layers. During a training procedure input vectors are presented to the input layer with or without specifying the desired output. According to this difference neural networks can be classified as supervised or unsupervised (self-organizing) neural nets. Networks can also be classified according to the input values (binary or continuous). The learning procedure itself contains three main steps, the presentation of the input sample, the calculation of the output and the modification of the weights by specified training rules. These steps are repeated several times, until the network is said to be trained. For details and survey of artificial neural networks see e.g. [1],[17].

The Kohonen network, also known as self-organizing map, a two-layer unsupervised continuous valued neural network [12]. This network is an excellent tool for ordering any kind of scattered data. The network has a strong self-organizing ability, which practically used for dragging a predefined structure - a polygon for curve modelling and a quadrilateral grid for surface modelling - towards the given points. After the so-called training procedure this predefined grid will follow the structure and distribution of the given points.

The outline of the training steps of a Kohonen network containing n input and m output node are the following (see Fig.1.):

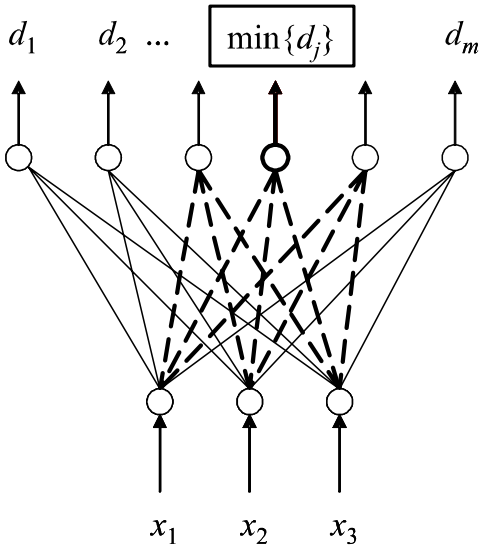

Figure 1. The training procedure of the Kohonen neural network

(1) Present new input values $x_i, i = 1, ..., n$.
(2) Input nodes send them to each output node.

(3) Output nodes compute the output values d_j by

$$d_j = \sum_{i=1}^{n} (x_i - w_{ij})^2 \qquad j = 1, ..., m$$

where w_{ij} is the weight associated to the connection from the i^{th} input node to the j^{th} output node.

(4) The node with the minimum output $d_{\min} = \min\{d_j\}$ is the winning node.

(5) The weights of the connections to the winning node as well as their neighbors are updated in a well-defined way.

(6) Go to step 1 until the network is trained.

How can we apply this tool for surface fitting? Let the number of input nodes $n = 3$, thus each output node has three connections. The three weights associated to these connections can be considered as spatial coordinates of a point $\mathbf{Q}_j(w_{1j}, w_{2j}, w_{3j})$ in 3D (see Fig.2.).One can fix the neighborhood

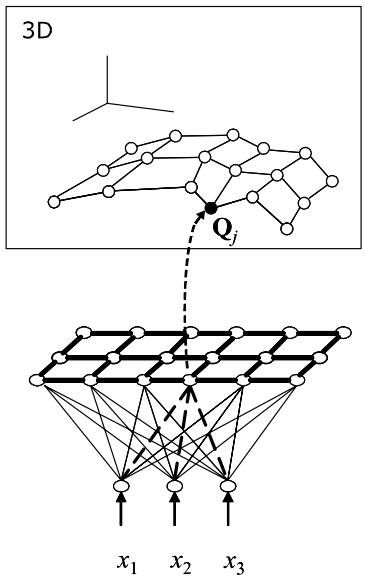

FIGURE 2. The application of the Kohonen network for surface fitting

relations of the network in advance, which yields a topologically fixed grid in 3D. If the weights of the network are changed during the training steps, some points of the spatial grid will change their spatial positions but the topology remains the same as before.

Now if the input values are spatial coordinates of one of the scattered points, then the training procedure will move this grid slightly towards the input point. This is because the winning node is associated to that point of the grid which is closest to the input point. The next iteration comes with another scattered point and the grid will move a little towards that point etc., thus after several iteration the grid will spread out and follow the overall shape of the scattered points. The movement can be controlled by numerical parameters as we will discuss it in the next section.

The precise algorithm of the training procedure is as follows:

Input: scattered points $\mathbf{P}_r$ (the number of the points are irrelevant)

Output: a grid with predefined topology which follows the overall shape of the scattered point set

(1) Fix the topology of the grid and the number of output nodes m. Let the number of input nodes $n = 3$. Let $t = 1$.
(2) Initialize the weights w_{ij} $(i = 1, 2, 3; j = 1, ..., m)$ of the network as small random numbers around the centroid of the point set or according to additional data (e.g. border of the point cloud).
(3) Present an input – three coordinates of a randomly selected spatial point $\mathbf{P}_i(x_1, x_2, x_3)$.
(4) Compute the output values and find the winning node by

$$d_{\min} = \min \left\{ d_j = \sum_{i=1}^{3} (x_i - w_{ij})^2, \qquad j = 1, ..., m \right\}$$

i.e. the node which is associated to the closest point $\mathbf{Q}_{\min}$ of the grid to the output point in 3D.
(5) Find the neighbors of the winning node by the neighborhood function $N(t)$ and update the weights of these nodes by

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i - w_{ij}(t)) \tag{2.1}$$

where $\eta(t)$ is a real-valued function called gain term.
(6) Let $t = t + 1$ and decrease $\eta(t)$ and $N(t)$.
(7) Go to step (3) and start next iteration until the network is trained. The network is said to be trained if the movement of the grid (i.e. the value of the gain term) falls under a predefined limit (normally $\eta(t) = 0.001$).

Some steps of the algorithm may need some further explanation. The topology of the grid in Step 1 means the structure of connections and the overall topology of the grid as well. Here the points of the grid are connected in quadrilateral way since the B-spline surface originally defined on this kind of grid. Other types of connectivity, however, can also be defined: triangular topology may be better suited for other applications (c.f. [21], [10]). The overall topology of the grid is a much harder problem, since normally we have no information of the topological properties of the input data. It has to be defined in advance and here we assume that the desired surface is of genus 0. The change of the overall topology of the grid during the training session would be the solution of the problem. Recent results in [10] and [14] show the problems of this direction of research which is, generally speaking, one of the main problems of surface reconstruction techniques today.

The number of output nodes m has to be fixed in Step1 of the algorithm. This work well in several cases but sometimes the refinement of the grid would yield a better surface. This problem can be solved by the dynamic version of the Kohonen neural network also known as growing cell structure [5]. The basic idea of the refinement is that further nodes are incorporated to the grid around the most frequent winners. In this case each node has a counter which is increased by 1 if this node is the winner. If one of the counters will be equal to a predefined limit a row or a column will be inserted

next to that neuron. For the detailed description of this modified algorithm see [19]. If we want to preserve the quadrilateral topology of connectivity we have to insert rows or columns anyway. In the case of triangular connectivity vertex split and edge collapse can be executed to refine the grid (c.f. [10]). This dynamic version of neural network certainly has some advantages: the number of training iterations decreases dramatically, the algorithm is generally faster even with the additional counters. The main drawback of this version is that the convergence of the dynamic version has not been established theoretically as yet.

Step 2 of the algorithm includes the initialization of the weights. This process determines the initial position of the grid. If we have no additional information about the point cloud, this initial situation can be a shranked grid somewhere "close" to the cloud. That's why we applied the following technique: some points are randomly chosen from the set and the centroid of them is computed. The grid is positioned initially around this centre adding small random values to the coordinates of the centre to obtain the vertices meanwhile the quadrilateral connections are fixed. This yields the shrinked grid which will spread out during the iterations. Actually this initial position is not extremely important in terms of the final surface - the first few iterations drag the grid towards the centre of the point cloud anyway. Most of the techniques not even mention the initial position ([9]) or using a very similar technique ([13]). The only important thing is that if the desired surface has a sphere-like shape and topology, like in our example (Fig. 3), the initial position should be somewhere inside the point cloud.

If we have additional information of the data, we can use it to determine an even better initial position. If for example we know the plane where the surface has a boundary, we can apply the "SOM boundary first" technique (c.f. [2]) to determine a planar grid with correct boundary which does not move during the training (see our example Fig. 3).

Fig. 3. shows some iterations of the training procedure as the grid is spreading out towards the input points. Finally the output of the algorithm is a topologically predefined grid which can be used as a control mesh for a B-spline surface. This B-spline surface is a rough approximant of the point cloud but can also be used for further process as a base surface for surface reconstruction. The reconstructed surface can be seen in Fig.3.d).

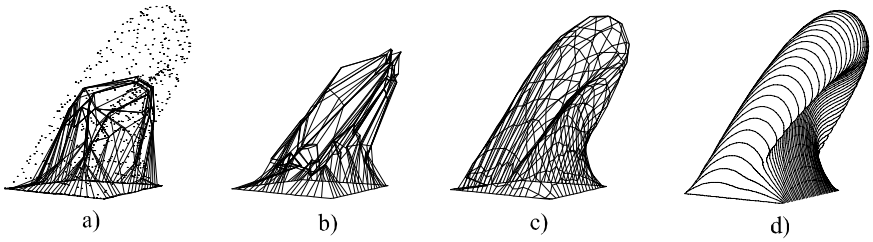

FIGURE 3. Movement of the grid during the training session: a) the scattered data and one of the first iterations b)-c) further iteration steps of the training d) the surface computed from the trained network

3. Numerical control of the training: the gain term and the neighborhood function

The algorithm given above contains two "hidden" functions: the gain term and the neighborhood function. These functions are for numerical control of the speed and accuracy of the approximation, i.e. the quality of the final surface.

The gain term, as one can see from equation (2.1) at step (5) of the algorithm, is for control of the measure of movement of the grid. The less the value of the gain term the smaller the movement of the point $\mathbf{Q}_{\min}$ and its neighbors towards the input point $\mathbf{P}_i$ (see Fig.4.). The value of the gain term has to be in $[0, 1)$. If it would be equal to 1, the point $\mathbf{Q}_{\min}$ would reach $\mathbf{P}_i$.The movement of the grid is desired to be large at

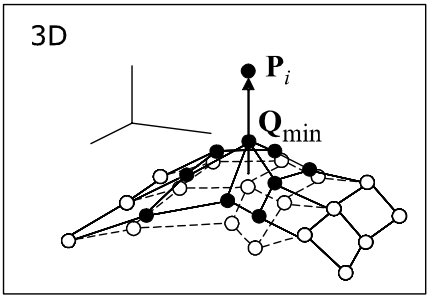

Figure 4. The gain term and the neighborhood function are for control of the movement of the grid: the gain term influences the measure of the movement of $\mathbf{Q}_{\min}$, the neighborhood function influences the moving part (filled circles)

the first iterations of the training session, when the overall shape of the scattered data should be found. The final iterations however require smaller movements when the final tuning of the grid is executed. Thus the gain term should be a Gaussian function. Notice that the gain term tends to 0 which guarantees the converhence of the algorithm as it has already been declared in [12]. Using the original suggestion of Kohonen here we applied the following function

$$\eta(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t}{q}\right)^2},$$

where t is the actual number of iterations and q is a parameter which has to be fixed in advance. It is very useful to incorporate the parameter q to the function by which one can control the steepness of the gain term : if the overall structure of the scattered data seems to be quite simple, then the decrease of the gain term can be very fast. If the shape is more complicated then slower change of the gain term is required to achieve good approximation (see Fig. 5). Naturally this latter case requires more iteration steps for the neural network to be trained.

The other function for the control of the training session is the neighborhood function $N(t)$. From step (5) of the algorithm it is clear that $N(t)$ is for the control of the moving part of the grid: the less the value of the neighborhood function the smaller the part of the grid which moves. During the

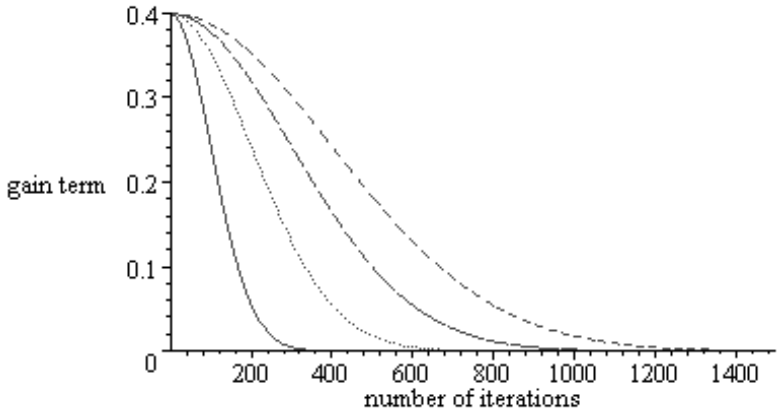

FIGURE 5. Graph of the gain term with parameter values q=100 (leftmost), 200, 300 and 400 (rightmost)

training session movement of large parts at the first steps and small parts at the final tune are desired. Thus similarly to the gain term the neighborhood function is also a Gaussian:

$$N(t) = \mathrm{INT}\left(\frac{m}{2}e^{-\frac{1}{2}\left(\frac{t}{s}\right)^2}\right),$$

where t is the actual number of iterations, m is the number of output neurons and s is a parameter which has to be fixed in advance. The role of s is similar to that of q. For simple shape the neighborhood function can be decreased faster than for more complicated structure to fasten the training session as well. If we have no information about the input data, q and s can be settled for 200.

Finally we have to mention that unfortunately these parameters cannot be settled for ever since different scattered data sets require different values. There is not much hope to determine these values in an automatic way as well, since in that case the overall structure of the input point set should have been described automatically *before* the modelisation. If we fix these parameters q and s as 200, the learning rate and accuracy will be a kind of average, i.e. probably better result can be achieved by modifiyng the parameters in advance. Other approaches, like [10], [11] also use some parameters to control the learning procedure and these also should be adjusted in a data dependent way.

4. Ruled surfaces from a set of rulings

Ruled surfaces are surfaces which contain a line through each point of the surface. They are of great importance in CAD/CAM. A further application of the described neural network technique is the construction of ruled surfaces from an unordered set of line segments, called rulings (c.f. Fig.6.).To apply the method described above first we shift the problem from $\mathbb{E}^3$ to the five dimensional projective space $\mathbb{P}^5$ with the help of Plücker-coordinates (c.f.[6]). Thus a point in $\mathbb{P}^5$ will be associated to each line in $\mathbb{E}^3$ and the

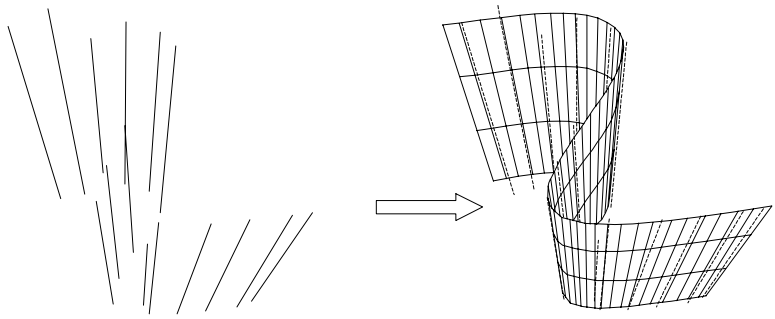

FIGURE 6. Construction of ruled surface from a set of unordered rulings.

original algorithm will be executed in $\mathbb{P}^5$. The output will be transformed back to $\mathbb{E}^3$ then.

Pick up two arbitrary points $\mathbf{A}(x_1, x_2, x_3)$ and $\mathbf{B}(y_1, y_2, y_3)$ of a line e and use the homogeneous coordinates $(x_1, x_2, x_3, x_4 = 1)$ and $(y_1, y_2, y_3, y_4 = 1)$, respectively. Compute the six Plücker-coordinates $(l_1, l_2, ..., l_6)$ of the line e by the following equations

$$\begin{aligned} l_{ij} &= x_i y_j - x_j y_i, \qquad i, j = 1, ..., 4 \\ (l_1, l_2, ..., l_6) &= (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}). \end{aligned}$$

The Plücker-coordinates are independent to the selected points and are unique, thus a point $\mathbf{L}_e(l_1, l_2, l_3, l_4, l_5, l_6) \in \mathbb{P}^5$ is assigned to the line $e \in \mathbb{E}^3$.

Now the algorithm to create ruled surface from a set of rulings is as follows:

Input: a set of unordered rulings e_r in $\mathbb{E}^3$.

Output: a topologically quadrilateral grid which contains only line segments in one direction

(1) A point $\mathbf{L}_r$ in $\mathbb{P}^5$ is assigned to each given ruling e_r by the Plücker-coordinates. Thus an unordered set of points is obtained.
(2) A Kohonen network is defined with 6 input nodes and the topology of the output nodes is fixed in a linear way. Let the points assigned to the nodes are $\mathbf{Q}_j \in \mathbb{P}^5$. These points form a polygon which moves during the training session.
(3) Applying the Kohonen network algorithm we obtain a polygon $\mathbf{Q}_j$ in $\mathbb{P}^5$ which follows the overall shape of the point set $\mathbf{L}_r$.
(4) Using the Plücker-coordinates backwards ordered lines $q_j \in \mathbb{E}^3$ are assigned to the vertices $\mathbf{Q}_j$ of the polygon in $\mathbb{P}^5$.
(5) Connecting the lines q_j and cut them by two planes a topologically quadrilateral grid is obtained with line segments in one direction.

The resulted grid can be applied as a control mesh for a B-spline surface which is a good approximant of the given rulings. An example can be seen in Fig.7. (the training steps are executed in $\mathbb{P}^5$ but shown in $\mathbb{E}^3$ for better visibility).

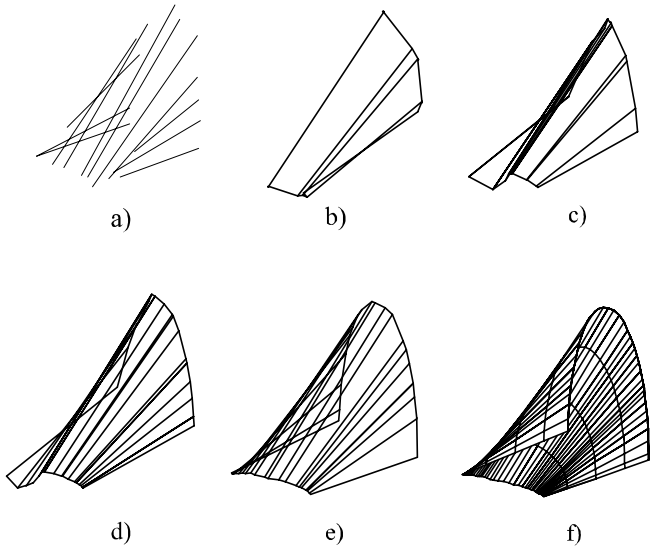

FIGURE 7. Ruled surface from a set of rulings: a) the input lines, b)-e) the training steps, f) the surface.

5. Conclusion

A scattered data approximation technique has been presented in this paper. The key step of the method is the ordering process where the Kohonen network produces a topologically predefined grid from the unordered set of points. The same technique was applied for 3D ruled surfaces. In that case the training process was shifted to the 5 dimensional projective space where points were associated to the rulings by Plücker-coordinates. Two functions and their influence for the numerical control of the training process were also presented.

In its present form this technique can be applied for topologically simple (disk or sphere-like) surfaces. Future directions of research must include the changing of the connectivity and overall topology of the grid during the training. The method for modelling ruled surfaces could be improved for developable surfaces and other special cases, like surfaces of rotation. A more automatic way of chosing learning parameters would also be a key step for real applications.

References

[1] J. Freeman, D. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley, 1991.
[2] J. Barhak, A. Fischer, Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques, IEEE-TVCG 7 (2001), 1-16.
[3] J. Barhak, A. Fischer, Adaptive reconstruction of freeform objects with 3D SOM neural network grids, Computers & Graphics 26 (2002), 745-751.
[4] G. Echevarría, A. Iglesias, A. Gálvez, Extending neural networks for B-spline surface reconstruction, Lecture Notes in Computer Science 2330 (2002), 305-331.
[5] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, Neural Networks 7 (1994), 1441-1460.
[6] V. Hlavaty, *Differential line geometry*, Nordhoff, 1953.

[7] M. Hoffmann, L. Várady, Free-form surfaces for scattered data by neural networks, Journal for Geometry and Graphics 2 (1998), 1–6.
[8] M. Hoffmann, Modified Kohonen Neural Network for Surface Reconstruction, Publ. Math. 54 (1999), 857-864.
[9] I.P. Ivrissimtzis, W-K. Jeong, H-P. Seidel, Using Growing Cell Structures for Surface Reconstruction, in: *Proceedings of the Shape Modeling International '03*, 2003, 78-86.
[10] I.P. Ivrissimtzis, W-K. Jeong, H-P. Seidel, Neural Meshes: Statistical Learning Methods in Surface Reconstrution, Tech. Report of Max-Planck Institut für Math., MPI-I-2003-4-007, 2003.
[11] W-K. Jeong, I. P. Ivrissimtzis, H-P. Seidel, Neural Meshes: Statistical Learning based on Normals, in: *Proc. of Pacific Graphics 03*, IEEE Computer Society Press, 2003 (to appear).
[12] T. Kohonen, *Self-organization and associative memory. 3^{rd} edition*, Springer-Verlag, 1996.
[13] G. K. Knopf, A. Sangole, Interpolating scattered data using 2D self-organizing feature maps, Graphical Models 66 (2004), 50-69.
[14] F. L. Krause, A. Fischer, N. Gross, J. Barhak, Reconstruction of freeform objects with arbitrary topology using neural networks and subdivision techniques, CIRP Annals - Manufacturing Technology 52 (2003), 125-128.
[15] W. Ma, P. He, B-spline surface local updating with unorganized points, Computer-Aided Design 30 (1998) 853–862.
[16] L. Piegl, W. Tiller, Parametrization for surface fitting in reverse engineering, Computer-Aided Design 33 (2001) 593–603.
[17] R. Rojas, *Neural Networks. A Systematic Introduction*, Springer-Verlag, 1996.
[18] T. Várady, R.R. Martin and J. Cox, Reverse engineering of geometric models - an introduction, Computer-Aided Design 29 (1997) 255–268.
[19] L. Várady, M. Hoffmann, E. Kovács, Improved free-form modelling of scattered data by dynamic neural networks, Journal for Geometry and Graphics 3 (1999), 177-183.
[20] V. Weiss, L. Andor, G. Renner and T. Várady, Advanced surface fitting techniques, CAGD 19 (2002), 19-42.
[21] Y. Yu, Surface reconstruction from unorganized points using self-organizing neural networks, in: *Proc. of IEEE Visualization 99*, 1999, 61-64.

Institute of Mathematics and Computer Science, Károly Eszterházy College, Leányka str. 4., Eger, Hungary
E-mail address: hofi@ektf.hu