Journal for Geometry and Graphics
Volume 3 (1999), No. 2, 177–181

Improved Free-form Modelling of Scattered Data by Dynamic Neural Networks

Lajos Várady[1], Miklós Hoffmann[2] and Emőd Kovács[2]

[1]*Institute of Mathematics and Informatics, Lajos Kossuth University*
P.O.Box 12. H-4010 Debrecen, Hungary
email: varadyl@math.klte.hu

[2]*Institute of Mathematics and Computer Science,*
Károly Eszterházy Teachers' Training College
P.O.Box 43. H-3300 Eger, Hungary
email: hofi@gemini.ektf.hu, emod@gemini.ektf.hu

Abstract. The aim of this paper is to improve the method of modelling scattered data by free-from surfaces presented in [5]. In that method a neural network was used for ordering the data and forming a quadrilateral control grid from the scattered points, hence the standard free-form methods like Bézier-surface or NURBS could be applied to approximate or interpolate the data. Instead of the original artificial neural network, which has been used for ordering the data, now a recent development, the dynamic version of this neural network is applied. Hence the preprocess of ordering the spatial scattered data is based on the neural network, the improvement of the network yields a much faster and more reliable algorithm.

Key Words: scattered data, free-form surface, neural network

MSC 1994: 68U05

1. Introduction

The handling of scattered data has several applications in CAD and there are numerous different methods to solve this problem, see [1, 4, 6]. In our paper [5] mentioned above we developed a new technique for modelling scattered data by free-form surfaces. The main advantage of this method is a preprocessing step which produces a topologically quadrilateral grid from the scattered data, hence the basic free-form methods can be applied without any kind of modification.

For this preprocessing step an artificial neural network has been used. The Kohonen network, developed by T. Kohonen [7] has a self-organizing ability, which allows a predefined grid to keep its topology during the training procedure, when this grid moves toward the input

ISSN 1433-8157/$ 2.50

scattered points and follows their spatial structure. This grid will be the input control grid of the arbitrarily chosen free-form method to construct the desired surface.

For this process, however, a good estimation of the number of vertices of the grid (i.e. the number of output neurons of the network) is required. If this number is too small, then some of the input points can be out of the result grid, while an unsufficiently large number of points can increase the processing time. In our method generally a number of $4n$ neurons was given for n input points, which is a reliable choice, but for large n the method can be rather slow. On the other hand, if the number of input points is infinite, then the best number of neurons also has some uncertainty.

To avoid this problem we will use a recent development of dynamic Kohonen network [2, 3] which allows to change the number of output neurons dynamically during the self-organizing process. This ability yields less neurons and hence a much faster process while on the other hand the method will be more reliable.

Since the original method has been described in [5], we now discuss only the development of the network, the modification of the algorithm and the results which prove the effectiveness of the improved algorithm.

2. The dynamic Kohonen network

The original Kohonen network consists of two layers of neurons. The input layer has three neurons, since the network will be trained by the coordinates of the 3D input points, while in the output layer the number m of neurons depends on the number n of input points, generally $m = 4n$, or an appropriate number, if n is large. The two layers are totally interconnected, and a weight is associated to every connection. These weights can be considered as the spatial coordinates of vertices of a grid, the predefined topology of which is quadrilateral. During the training process the weights will be changing, hence this grid will move slowly in the three dimensional space toward the input points, meanwhile the topology of the grid will remain the same.

As we can see, the number of output neurons is a crucial point of the network and the choice of m has some uncertainty. To avoid this problem, the dynamic version of the Kohonen network is applied, in which the number of the output neurons, and hence the number of the vertices of the grid is growing dynamically during the training session.

The basic idea of the dynamic change of the net is the following: in every iteration of the training an active neuron is chosen, which is the closest vertex of the grid to the input point. This vertex and its neighbors is moved toward the input point. Suppose, that a certain neuron, or the neurons of a neighborhood is active very frequently. This means that geometrically this part of the grid is not dense enough, there are several input points around this neuron or this neighborhood. So extra neurons will be inserted next to the most frequently chosen neuron.

Since the insertion of one neuron would deform the topology of the grid, we will insert a row or a column of neurons instead. This row or column will be inserted to that side of the neuron, on which the neighbor neuron is the least active. This way the grid will grow dynamically, and will reach the most appropriate size to the end of the training.

3. Insertion of new neurons

Now we give the formal description of this insertion algorithm: Let the number of output neurons be $k \times l$, ($k = 2, l = 2$ at the beginning of the procedure). A resource variable λ_{ij}, ($i = 1, ..., k\,;\, j = 1, ..., l$) is associated to every output neuron. The $(i_0, j_0)^{th}$ resource variable will be increased by 1 when the $(i_0, j_0)^{th}$ output neuron will be active. Moreover, a constant Λ is defined as an upper limit of the λ_{ij}. If one of the variables will be equal to this limit (i.e. this neuron has been chosen Λ times), a row or a column will be inserted next to the associated neuron.

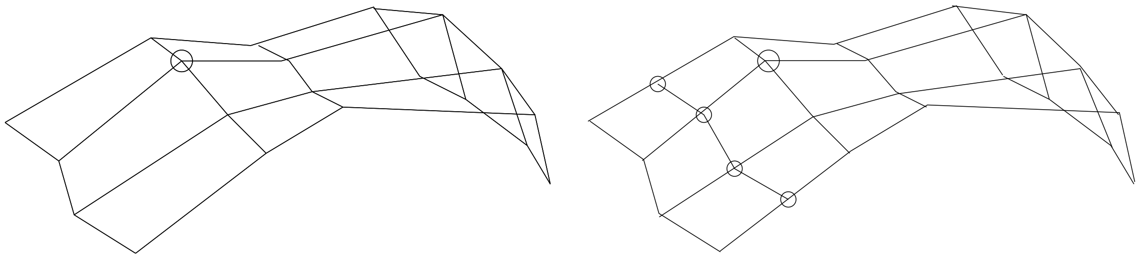

Figure 1: The insertion of new neurons beside the most active neuron

After the former training steps (discussed in [5]) the following steps must be executed:

- STEP 1: Compare the resource variables with the limit. If there is an index ij, for which

$$\lambda_{ij} = \Lambda,$$

  then
- STEP 2: Choose the least frequent neighbor of this neuron, that is find

$$min(\lambda_{i-1\,j}, \lambda_{i\,j-1}, \lambda_{i+1\,j}, \lambda_{i\,j+1})$$

  (some of them may not exist, if the neuron is on the border of the grid)
- STEP 3: Suppose, that the neighbor we found is the $(i, j-1)^{th}$ neuron. Then a column will be inserted between the $(j-1)^{th}$ and the j^{th} column. The weights of the neurons of this new column, as the coordinates of the new vertices of the grid, will be chosen as the average of the weights of their two neighbors, hence geometrically these new vertices will be the midpoints of the sections of their neighbors. k or l has to increase accordingly, in this example $l = l + 1$.
- STEP 4: Reset all the resource variables: $\lambda_{ij} = 0$, ($i = 1, ..., k\,;\, j = 1, ..., l$).

A further advantage of this insertion is that the ratio of k and l, i.e., the shape of the grid is changing automatically and will be the most advantegous after the training procedure.

The original method can also be modified in terms of stopping criteria. If the number of input points is relatively small, then the original network is said to be trained if all the input points are on the grid. In this case one can construct a surface (by one of the interpolation methods) which passes through all the input points. If we have hundreds of input points, or data given by a distribution, as in some of the scattered data problems, then the network is said to be trained if the change of the grid is under a certain predefined limit. In this latter case the dynamic network would be also stopped by exceeding a predefined limit for the number of neurons (hence the number of output neurons can even be smaller than the number of input points).

3.1. Efficiency of the new algorithm

The new algorithm improves the method in two ways. First of all, the training iterations, that is the number of presenting input values decreases dramatically. Since the insertion needs only very limited computing time, finally the new algorithm is much faster, than the original one. The table below shows the changes of the number of sufficient iterations (average after several runs). On the other hand, the number of vertices of the final grid also decreases, which generally yields a smoother surface.

Input points	*orginal*		*dynamic*	
	iterations	*neurons*	*iterations*	*neurons*
10	3200	40	2300	30
100	5500	400	3600	250

After the training session a grid will be obtained produced by the Kohonen net. This grid can be considered as the control mesh of the future surface. Either interpolation or approximation can be executed by the standard NURBS surface or Bézier-surface method and at this time of the procedure it is irrelevant, that the starting points were scattered so we can consider them as regular, ordered vertices of a control mesh.

4. Concluding remarks

A new algorithm of scattered data manipulation is presented in [5], where a neural network was applied to order the scattered points. This algorithm is modified and improved by using the dynamic version of the Kohonen algorithm, which allows to change the number and structure of the output neurons during the training session. This possibility yields less computing time and output neurons, while the shape of the grid also fits better to the input points. These advantages help to model the scattered points with a better free-form surface.

Acknowledgement

The authors wish to thank the Hungarian National Scientific Fund (OTKA No. F019395) for their financial support of this research project.

References

[1] W. Boehm, G. Farin, J. Kahmann: *A survey of curve and surface methods in CAGD*, Computer Aided Geometric Design **1**, 1–60 (1984).

[2] B. Fritzke: *Growing Cell Structures — a Self-organizing Network for Unsupervised and Supervised Learning*, Neural Networks **7**, no. 9, 1441–1460 (1994).

[3] B. Fritzke: *Growing Grid — a Self-organizing Network with Constant Neighborhood Range and Adaptation Strength*, Neural Processing Letters **2**, no. 5, 9–13 (1995).

[4] T. Foley, H. Hagen: *Advances in scattered data interpolation*, Surv. Math. Ind. **4**, 71–84 (1994).

[5] M. Hoffmann, L. Várady: *Free-form Surfaces for Scattered Data by Neural Networks*, J. Geometry Graphics **2**, no. 1, 1–6 (1998).

[6] J. Hoschek, D. Lasser: *Fundamentals of Computer Aided Geometric Design*, AK Peters, Wellesey, Massachusetts, 1993, pp. 388–437.

[7] T. Kohonen: *Self-organization and associative memory*, third edition, Springer Verlag, 1989.

Received March 26, 1999