Acta Acad. Paed. Agriensis, Sectio Mathematicae **27** *(2000) 63–68*

CONSTRUCTING RULED B-SPLINE SURFACES BY KOHONEN NEURAL NETWORK

Miklós Hoffmann, Emőd Kovács (Eger, Hungary)

Abstract. General and special ruled surfaces play a central role in several CAD applications. The purpose of this paper is to develop a method with the help of which one can construct free-form ruled surface for given lines as rulings. The Kohonen neural network and the Plücker coordinates of projective geometry help us to construct a standard free-form surface for these data.

AMS Classification Number: 68U05

1. Introduction

Ruled surfaces and their special subclass, developable surfaces are well-known in classical geometry, but also have extended applications in computer geometry and graphics as well as in computer aided manufacture. In these latter fields, however, the standard way of descripting surfaces is the different types of spline-surfaces, like Bezier, B-spline or NURB surface. Hence it is highly desired to describe and construct ruled and developable surfaces as spline surface. Several methods have been developed to approximate or interpolate by such kind of surfaces, see e.g. [1,2].

The standard definition of any type of spline surfaces uses a quadrilateral control grid as input data, which is a $(2, n)$ type in case of ruled surfaces. The main purpose of most of the known methods is to construct this control grid. In our method the Kohonen neural network has been applied in a preprocessing step to produce a control grid from the original input data. This artificial intelligence tool has already been applied succesfully by the authors in other approximation problems [3–5].

In our paper orginial input data structure consists of a set of given lines, called rulings. In this case the description of these lines and the neural network by homogeneous coordinates will be advantegous, which is a well-known technique in projective geometry. In another sense this computer geometrical approach has already been developed by Ravani et al. in [6]. The Plücker coordinates of the

This research was partially sponsored by the Hungarian Scientific Research Foundation (OTKA) No. F032679.

projective lines [7] will be also applied, which is investigated by Chen and Pottmann [9].

2. Ruled Surfaces

We begin our discussion with the definition of ruled surfaces.

Definition. A surface is ruled surface, if through every point of the surface there passes a line which lies entirely on the surface. These lines are called rulings of the surface.

This means, that a ruled surface is covered by a one parameter set of lines, the rulings. These lines has the following property: a ruling lies in the tangent plane of the surface at every point of the ruling. This property is important in terms of the special ruled surfaces, called developable surfaces. The original definition of these surfaces is the following:

Definition. A surface is developable surface, if it can be isometrically mapped (i.e. developed) into the plane.

Considering the property of the ruled surfaces mentioned above and this latter definition, one can easily see the following, well-known result:

Theorem. *A ruled surface is developable, if and only if the tangent planes at all points of an arbitrary ruling coincide.*

If these tangent planes vary along the ruling, the surface is called general (non developable) surface. In this paper we will consider only general ruled surfaces.

3. Constructing Ruled Surfaces with Given Rulings

Consider the following problem: an arbitrary set of lines are given, find a ruled surface passing these lines, that is a surface for which the given lines are rulings.

More precisely, we would like to interpolate these given lines with one of the standard spline surfaces of CAD, a Bézier, a B-spline or a NURB surface. In this paper the B-spline surface will be used, but all of these surfaces are defined by a control grid of points, called control points. This grid regularly has a quadrilateral topology, and one can chose to approximate or to interpolate these points.

We will use the following well-known property:

Theorem. *If the points of the control grid of a B-spline surface form straight lines in one direction, then the result surface is ruled surface.*

Thus we need to form a quadrilateral grid in which the verteces form straight lines in the direction of the given lines. These lines will be the rulings of the future surface, if it will interpolate the control points, and the theorem mentioned above yields it will be a ruled surface. To find this grid the Kohonen neural network will

be applied. This tool will produce the desired grid from the given lines, and then this grid will be the input control grid of the standard interpolation method of the B-spline surface.

In earlier papers the authors has already been used this neural network for approximation and interpolation problems, mainly for constructing curves and surfaces from scattered points [3-5]. The Kohonen neural network has a strong self-organizing property, that is during the training procedure a topologically invariant grid of points is moved towards the scattered data and tried to follow their structure and distribution.

This network is a continuous valued two-layered network. The first layer is the input layer, and there are three input neurons, since normally the input values are coordinates of spatial points. The output layer consists of several neurons forming a quadrilateral grid. The neurons of the two layers are interconnected and each of the connections has a weight, which is modified by a certain training algorithm during the training procedure. Since every output neuron has three connections and three weights, these weights can be considered as spatial coordinates of points, which form a grid with the same topology as of the output neurons. This latter grid moves towards the given points and after the training procedure the grid will pass all of the input points. For the more detailed description of this method and the Kohonen neural network see the earlier papers of the authors and [8].

Now, however, this method has to be modified, since the input structure consists of lines instead of points, and the grid has to be fullfilled the property mentioned above. Here we have to introduce a very effective tool for handling spatial lines: the Plücker-coordinates. Let us briefly overview this classical part of geometry.

Consider the three dimensional projective space P^3. Here every point has four homogeneous coordinates (x_1, x_2, x_3, x_4), which correlate the Cartesian coordinates (x, y, z) as $x = x_1/x_4, y = x_2/x_4, z = x_3/x_4$ if the point is not at infinity. Now consider a line l of P^3 passing two points (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4). The six Plücker coordinates of this line will be

$$(l_1, ..., l_6) := (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}), \quad l_{ij} = x_i y_j - x_j y_i.$$

These coordinates do not depend on the choice of the two points on the line l and fullfill the Plücker identity:

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0.$$

This 1-1 correspondence between the homogeneous 6-tupels of real numbers and the lines of P^3 yields another 1-1 correspondence between the lines of P^3 and the points of the five dimensional projective space P^5. With the help of this latter mapping we can embed the given rulings and the lines of the future grid of the surface. This way the Kohonen network will be trained by points from P^5, that is the input layer will contain 6 neurons, while the output layer will be a polygon

in P^5. When the neural network is trained, the result polygon passes through the given points of P^5. Transforming this situation back to P^3 the given lines and a quadrilateral grid will be received, which grid contains the given lines and consists of straight lines at that direction.

4. The Algorithm

Let a set of lines $l^i, (i = 1, ..., n)$ be given. Compute the Plücker coordinates of these lines:

$$l^i(l^i_j) \quad i = 1, ..., n \quad j = 1, ..., 6$$

Consider a Kohonen neural network with 6 input neuron and $m(= 4n)$ output neuron. Denote the weights of the connection from the j^{th} input neuron to the i^{th} output neuron by w_{ij}. Thus the weights $(w_{i_0 j}), (j = 1, ...6)$ of the connections to an output neuron can be considered as the coordinates of an output point V_i in P^5.

Now we train the Kohonen neural network. Initializing the weights w_{ij} as small random values around the average of the coordinates of the input points and setting the training time $t = 1$, the coordinates $(l^{i_0}_j), (j = 1, ...6)$ of a randomly chosen input point is presented. The Euclidean distance of all output points to this input point is computed:

$$d_m = \sum_{s=1}^{6} (l^{i_0}_s - w_{ms})^2$$

Finding the winning unit, that is the output point closest to the input one, the weights of the connections to this point and to its neighbors are updated by

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(l^{i_0}_j - w_{ij}(t))$$

where t denotes the training time. After updating the weights a new randomly selected input is presented until the network is trained. The network is said to be trained, if all the input lines are on the grid. If the input structure consists of hundreds of lines, then this requirement would yield long computing time, hence in this case the network is trained if the changes of the weights fall under a predefined limit.

When the training process finished, the result is a polygon in P^5 with the verteces $V_i(w_{ij}), (i = 1, ..., m), (j = 1, ..., 6)$. Transforming these verteces back to P^3 the projective lines v_i are received, with the Plücker coordinates (w_{ij}). Using the definition of these coordinates backwards, the points of these lines can be computed. Thus a set of lines is gained among which the all the original input lines can be found.

During the training session one can compute the whole grid at every training time t, using the points of the lines v_i as verteces connected also in the other

direction. Hence after the training procedure beside the rulings a quadrilateral grid containing these rulings is also received. The verteces of this grid can be applied as input control points to any of the standard free-form surface algorithm. Hence we can interpolate or approximate these points and the rulings.

5. Concluding Remarks

Obviously there are infinitely many ruled surface passing through a set of given spatial rulings, and probably each of the methods would produce a different one. Comparing our method with other algorithms the main advantage is the following: beside the self-organizing ability of the Kohonen network it also has a minimal energy property (see [8]), that is the network tries to minimize the strain energy during the training session. Hence in a sense the result grid is the smoothest grid containing all the given rulings, while other methods cannot quarantee this property.

The method described above could be developed also for developable surfaces, but it is a subject for further research.

References

[1] Hoschek, J., Schwanecke, U., Interpolation and approximation with ruled surfaces, in: *The Mathematics of Surfaces VII.* (ed.: Robert Cripps), Elsevier, (1988) 213–231.

[2] Pottmann, H., Farin, G., Developable rational Bezier and B-spline surfaces, *Computer Aided Geometric Design*, **12** (1995), 513–531.

[3] Hoffmann, M., Várady, L., Free-form curve design by neural networks, *Acta Acad. Paed. Agriensis*, **24** (1997), 99–104.

[4] Hoffmann, M., Várady, L., Free-form surface design by neural networks, *Journal for Geometry and Graphics*, **2** (1998), 1–6.

[5] Várady, L., Hoffmann, M., Kovács, E., Improved free-form modelling of scattered data by dynamic neural networks, *Journal for Geometry and Graphics*, **3** (1999), 177–183.

[6] Bodduluri, R., Ravani, B., Design of developable surfaces using duality between plane and point geometries, *Computer-Aided Design*, **10** (1993), 621–632.

[7] Hlavaty, V., Differential line geometry, Nordhoff Ltd, Groningen (1953)

[8] Kohonen, T., Self-organization and assoctaitive memory, 3rd editon, Springer Verlag (1996)

[9] Chen, H., Pottmann, H., Approximation by ruled surfaces, *Journal of Computational and Applied Mathematics*, **102** (1999), 143–156.

Emőd Kovács
Inst. of Mathematics and Informatics
Eszterházy Károly College
H-3300 Eger, Hungary
Leányka str. 4.
e-mail: emod@ektf.hu

Miklós Hoffmann
Inst. of Mathematics and Informatics
Eszterházy Károly College
H-3300 Eger, Hungary
Leányka str. 4.
e-mail: hofi@ektf.hu