The Calculation of Parametric NURBS Surface Interval Values Using Neural Networks

Erkan Ülker and Ahmet Arslan

Selçuk University, Computer Engineering Department,
42075 Konya, Turkey
{Eulker, Ahmetarslan}@selcuk.edu.tr
http://www.mmf.selcuk.edu.tr/bilgisayar/akademik/akademik1.html

Abstract. Three dimensional coordinate values of parametric NURBS (Non-Uniform Rational B-Splines) surfaces are obtained from two dimensional parameters *u* and *v*. An approach for generating surfaces produces a model by giving a fixed increase to *u* and *v* values. However, the ratio of three dimensional parameters increases and fixed increase of *u* and *v* values is not always the same. This difference of ratio costs unrequired sized breaks. In this study an artificial neural network method for simulation of a NURBS surface is proposed. Free shaped NURBS surfaces and various three dimensional object simulations with different patches can be produced using a method projected as network training with respect to coordinates which are found from interval scaled parameters. Experimental results show that this method in imaging modeled surface can be used as a simulator.

1 Introduction

Customer's preferences for aesthetic appearance may vary and this causes increase of using free surfaces in product design. Therefore using Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) became inevitable in modern industry. NURBS (Non-Uniform Rational B-Spline) surfaces are known as common parametric surface definition method. Moreover, coordinate calculating about NURBS surface is rather complex because of conversion to a basic type of surface. Since each surface has a different mathematical expression, complete surface conversion is not easy and conversation procedure always require casualty of surface data [1, 2].

Non-parametric methods require big amount of surface data for storage surfaces [3]. Three dimensional coordinate values of parametric NURBS surfaces are produced from *u* and *v* two dimensional parameters. The main advantage of the NURBS surfaces is their ability to reproduce second-degree basic surfaces, such as cylindrical, spherical, parabolic and hyperbolic surfaces. Furthermore, usage of NURBS in computer graphic and CAD increase rapidly [4-11]. Another type of parametric surface type is double cubic polynomial surface, which is out of the scope of this study.

A common approach to produce the surface is forming a model by fixed increments of *u* and *v* values. Despite the fixed increments in parameters, 3-D coordinates do not change in the same amount, and this causes unwanted fractures in the surface. As a

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 247 – 254, 2006.

result the surface will not be aesthetic enough. Another approach is to make calculations with the derivations of the parametric variables to determine the interval values of the surface. However, this approach increases cost and complexity of surface calculation [12].

Artificial Neural Networks (ANN) is a machine learning algorithm which simulates human brain cells. ANN is efficient method to solve complex and non-linear problems. They are successfully used in many areas especially in image processing and pattern recognition. Once the Neural network is trained, it can solve the problem efficiently [13-18]. ANN is formed by a number of connected neurons in layers. A neuron in ANN puts a linear combination of its input(s), either from the previous layer of neurons or from user input and its weight into an activation function. This function returns the value that the neuron will pass on to the next layer. The ANN learns by adjusting these weights by some pre-chosen algorithm. To solve a non-linear problem, a multi-layer ANN should be used which contains one or more hidden layer [19-23].

In this study we present an alternative method based on an ANN model, to produce three dimensional surfaces as an image. The ANN model is developed to calculate intermediate values of a NURBS surface which is produced by giving a fixed increase to *u* and *v* parameters.

2 Parametric NURBS Surfaces

NURBS is an industry standard for geometric design and visualization. Some advantages of using NURBS surfaces are listed below [24-28]: (i) It is a common form that is suggested for both Standard Analytical Form shapes and free form shapes; (ii) It provides flexibility for designing shapes in many different types; (iii) They can be processed efficiently by digital and stable algorithms; (iv) They have an unchangeable character under affine like perspective transformations. NURBS are generalization of rational and non-rational Bezier curves and surfaces. One of the disadvantages of the NURBS is, even defining a basic shape like a circle, requirement of additional memory. In addition to control points, some other parameters appear in the definition of NURBS. But after the operation it supplies wanted. NURBS shapes can not be defined only with control points; weight of each control point is also required. A NURBS curve, *C(u)*, is a vector valued and piecewise rational function which can be expressed as;

$$C(u) = \frac{\sum_{i=0}^{n} W_i * P_i * N_{i,k}(u)}{\sum_{i=0}^{n} W_i * N_{i,k}(u)} \tag{1}$$

where W_i represents *weights*, P_i represents *control points* (vector) and $N_{i,k}$ is the normal *k degree B-spline basis functions* and $N_{i,k}$ can be defined as a recursive function as follows:

$$N_{i,k}(u) = \frac{u - t_i}{t(i+k) - t_i} * N_{(i,k-1)}(u) + \frac{t_{(i+k+1)} - u}{t_{(i+k+1)} - t_{(i+1)}} * N_{(i+1,k-1)}(u)$$
$$N_{i,0} = \begin{cases} 1, & if \quad t_i \le u \le t_{i+1} \\ 0, & t_i \le u \le t_{i+1}, \quad otherwise \end{cases} \Bigg\} \tag{2}$$

where t_i represents knots that are shaped by a knot vector and $U = \{t_0, t_1, \ldots, t_m\}$.

A definition of a NURBS curve in equation (1) can be organized using rational basic functions as follows:

$$C(u) = \sum_{i=0}^{n} P_i * R_{i,k}(u), \quad and \; R_{i,k}(u) = \frac{W_i * N_{i,k}(u)}{\sum_{j=0}^{n} W_j * N_{j,k}(u)} \tag{3}$$

Any NURBS surface is defined similarly by:

$$S(u,v) = \sum\sum P_{i,j} * R_{i,k,j,l}(u,v), \quad and \; R_{i,k,j,l}(u,v) = \frac{W_{i,j} * N_{i,k}(u) * N_{j,l}(v)}{\sum_{r=0}^{n}\sum_{s=0}^{m} W_{r,s} * N_{r,k}(u) * N_{s,l}(v)} \tag{4}$$

3 Artificial Neural Network Model

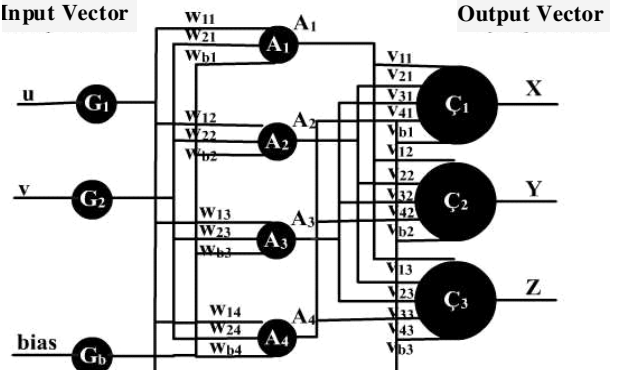


Fig. 1. Desired model of Artificial Neural Network

3.1 Major Construction of Network

ANN structure is shown in *Fig 1*. Network is constructed in two layers. First layer has two neurons *(G_1, G_2)* and second layer has three (C_1, C_2, C_3). Second layer works as a shape producer and function of the first layer is key point distributor. Network's input is a 3 dimensional *X= (u, v, bias)* vector. Bias input is 1, and network's output is 3 dimensional *Y=(x, y, z)* vector. Each neuron of first layer is one-to-one related to each input and first two layers are completely related to each other. The power of relation between m^{th} input and n^{th} neuron of first layer is represented by W_{mn}. *W* power matrix

is formed from *m X n* power matrix and the power of relation between first layer's n^{th} output and p^{th} nerve under second layer is represented by v_{np}. *V* power matrix has *nXp* size. In the network design, *m=3*, *n=4*, and *p=3*.

A learning period is needed by artificial neural network similar to the human brain. During the calculation process in designed ANN, *x, y* and *z* coordinate values are calculated and saved into a file according to the addition of *u* and *v*, then generalization is done by using weights which obtains relation between neurons according to the these data (NURBS surface coordinate *x, y* and *z* that match *u* and *v* values). In the learning process back-propagation algorithm is used to obtain desired output.

3.2 Training Algorithms

In general, training algorithms for neural networks can be classified into three basic groups: controlled learning, non-controlled learning, and reinforcement learning. Learning in the neural network determines the weight set that gives the desired output of the network which is tested for inputs and outputs. This weight vector is formed by increasing or decreasing the values of the weights in a loop until the differences between desired output and network output is minimal. Back-Propagation learning algorithm is one of the most common of them. It has been used in several areas. However its learning time is long because its approach ratio is very poor. Catching local minimum is another disadvantage of Back-Propagation algorithm. Another learning algorithm Levenberg is a variation of Newton Algorithm. Its learning time is shorter. On the other hand, more calculations and memory resources are required. Back-Propagation algorithm requires less memory but convergence to result is slower. For this reason Back-Propagation algorithm is preferred in ANN design.

3.3 Back-Propagation Learning Algorithm

With back-propagation, the input data is repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (back propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output.

In this study, particularly, delta rule is used which is commonly used in training of multi-layered networks. Modification of weight between j^{th} input and i^{th} neuron for *k* sample pair of input-output is expressed as follows:

$$\Delta w_{i,j}^{k} = \alpha (T_{i}^{k} - O_{i}^{k}) x_{j}^{k} \tag{5}$$

where T_i is desired target output, O_i is real output, α is ratio of learning, and x_i is j^{th} input of network. For a sample pair on k^{th} layer, total error and average error function is defined as:

$$E = \frac{1}{2}\sum_{k} (T_k - O_k)^2 = \frac{1}{2P}\sum_{p}\sum_{j} (T_j - O_j)^2 \tag{6}$$

where T_k is desired target output, O_j is calculated output, and p is the total number of input-output pair in the training set. In the case of not converging, the formula can be generalized as in (7) for setting weights as it includes β momentum term:

$$\Delta w_{i,j}(t+1) = \alpha \delta_i^k O_j^k + \beta \Delta w_{ij} \tag{7}$$

where β is momentum term, δ_i^k is i^{th} neuron's error value for k sample pair and contains derivative of used threshold function. The threshold function determines the output of process element. Usually, a function which can derivative is preferred. Although there are many types of threshold functions, sigmoid function is used in this study because it is used commonly in back-propagation.

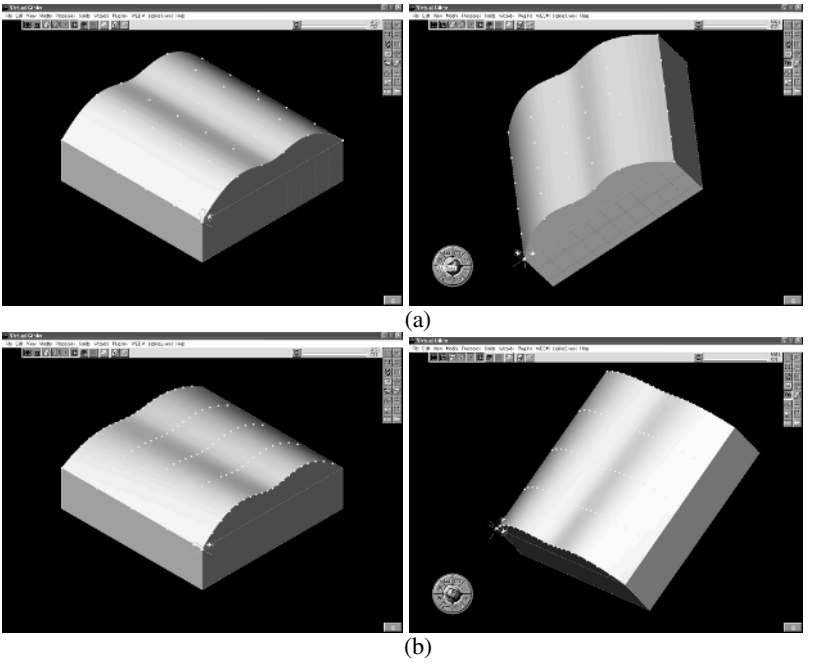


Fig. 2. Two different sample surface; (a) First two Surfaces (above) (b) Second Surfaces (below) (are produced by Solidworks and Neural Network)

4 Experimental Results

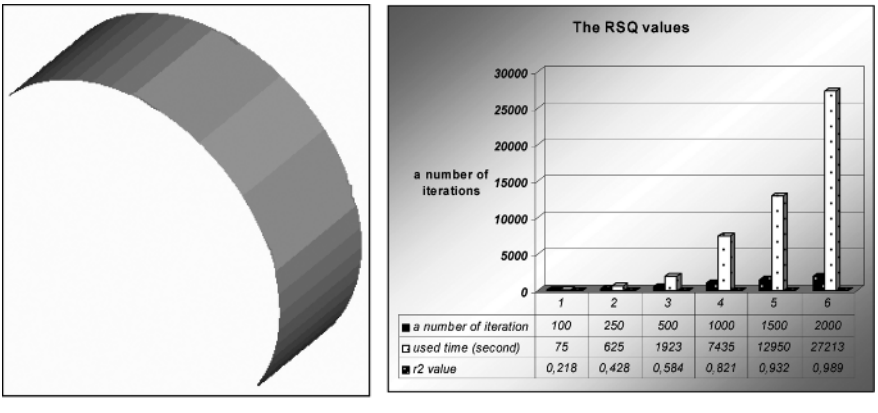
Two different surfaces produced by commercial software *Solid Works 2004* and displayed in *Fig 2* are considered in this study for experiments. Coordinates on the layer (*S(u,v)*) are generated by fixed increments of *0.1* on *u* and *v* values as the surface is created. After network is trained according to these values, weight values are calculated by *Back Propagation Learning Algorithm*. Then surface coordinates is produced by giving *u* and *v* the values that are multiples of *0.1*. Parametric equation of surface, real coordinates which is produced according to this addition, and coordinate

values which are produced by suggested method are compared and a part of comparison results for one of these surfaces is given in *Table 1*. Experiments show that our model is flexible enough and able to model any desired shape. A comparison of the computation time shows that our approach yields faster results than the other algorithms that have fixed *u* and *v* increment after the first training of network.

Table 1. Surface Coordinates Values that are Produced by Solid Works 2004 and Back Propagation Algorithm

u	v	Solid Works Outputs			ANN Outputs		
		X	Y	Z	X	Y	Z
0	0.2	0	40	0	2	42	11
0	0.8	0	160	0	10	167	9
0.2	0.8	49	160	30	34	125	24
0.4	0.2	84	40	27	77	30	20
0.6	0	115	0	27	110	9	27
0.6	0.2	115	40	27	111	20	27
0.8	0.8	163	141	30	150	160	30
1	0.6	200	120	0	199	170	0
1	0.9	200	190	0	200	197	0

The graphical relationship between the number of iterations in training the ANN, the computation time, and the r^2 values for the surface of *Fig 3.a* is displayed in *Fig 3.b*.



(a) (b)

Fig. 3. (a) The test surface saved as NURBS for simplicity. (b) The relation between a number of iterations and used time for training of ANN in terms of surface demonstrated in Fig. 3.a.

5 Conclusion and Future Work

An approach is defined by using ANN's for getting intermediate coordinates of three dimensional NURBS surfaces. Free form shaped NURBS surfaces which do not contain pockets and various three dimensional NURBS models which are formed from several patches can be produced by using this method. Experimental results show that this method can be used as an alternative simulation tool for visualization of the modeled surface.

As a future work, other Artificial Intelligence techniques can be employed to calculate intermediate values of these surfaces and results can be compared.

Acknowledgements

This study has been supported by the Scientific Research Projects of Selcuk University (in Turkey).

References

1. Zeid, İbrahim , CAD/CAM Theory and Practice, McGraw-Hill, (1991).
2. Hyunbo Shim and Euikwon Suh, Contact treatment algorithm for the trimmed NURBS surface Journal of Materials Processing Technology, Vol. 104, p. 200-206, (2000).
3. Hearn, Donald, Computer graphics, Prentice Hall, New Jersey, (1994).
4. Aziguli W., Goetting M. And Zeckzer D., Approximation of NURBS curves and surfaces using adaptive equidistant parameterizations, Tsinghua Science&Technology, Vol. 10, pp. 316-322, (2005).
5. Che X., Liang X., and Li Q., G1 continuity conditions of adjacent NURBS surfaces, Computer Aided Geometric Design, Vol. 22, pp. 285-298, (2005).
6. Yang J., and Abdel-Malek K., Approximate swept volumes of NURBS surfaces or solids, Computer Aided Geometric Design, Vol. 22, pp. 1-26, (2004).
7. Ma W., But W.-C., and He P., NURBS based adaptive slicing for efficient rapid prototyping, Computer Aided Geometric Design, Vol. 36, pp. 1309-1325, (2004).
8. Yau H.-T., Kuo M.-J., NURBS machining and feed rate adjustment for high-speed cutting of complex sculptured surfaces, International Journal of Production Research, Taylor & Francis, Vol. 39, (2001).
9. Kumar S., Manocha D., Lastra A., Interactive display of large NURBS models, Visual-ization and Computer Graphics, IEEE Transactions on Vol. 2, pp. 323-336, (1996).
10. Wang Q, Hua W., Li G., Bao H., Generalized NURBS curves and surfaces, Geometric Modeling and Processing, Proceedings p 365-368, (2004).
11. Hatna A., Grieve B., Cartesian machining versus parametric machining: a comparative study, International Journal of Production Research, Taylor & Francis, Vol 38, (2000).
12. L. Piegl, W. Tiller: Geometry-based triangulation of trimmed NURBS Surfaces. Computer Aided Design, Vol. 30(1), pp. 11-18, (1998).
13. E. Castillo, A. Iglesias, J.M. Gutiérrez, E. Alvarez, J.I., Functional Networks. An application to fitting surfaces, World Multiconference on Systemics, Cybernetics and Informatics, Proceedings of the ISAS-98 Fourth International Conference on Information Systems, Analysis and Synthesis Vol. 2, pp. 579-586, (1998).

14. A. Iglesias, A. Galvez, Applying Functional Networks to CAGD: the Tensor Product Surface Problem, Fourth International Conference on Computer Graphics and Artificial Intelligence, pp. 105-115, (2000).
15. A. Iglesias, A. Galvez, Fitting 3D data points by extending the neural networks paradigm, Computational Methods and Experimental Measurements, WIT Press /Computational Mechanics Publications, Southampton-Boston (Series: Computational Engineering) Vol. 3, 809-818., (2001).
16. A. Iglesias, A. Gálvez, Applying Functional Networks to Fit Data Points From B-spline Surfaces, Fitting 3D data points by extending the neural networks paradigm, Proceedings of the Computer Graphics International, CGI'2001, Hong-Kong (China), IEEE Computer Society Press, Los Alamitos, California 329-332, (2001).
17. G. Echevarría, A. Iglesias, A. Gálvez, Extending Neural Networks for B-spline Surface Reconstruction, Computational Science-ICCS'2002, Springer-Verlag (Series: Lectures Notes in Computer Science), Berlin Heidelberg, Vol. 2330, pp. 305-314 (2002).
18. A. Iglesias, G. Echevarria, A. Galvez, Functional Networks for B-spline Surface Reconstruction, Future Generation Computer Systems, Special Issue on "Computer Graphics and Geometric Modeling", Vol. 20, Issue 8, pp. 1337-1353, (2004).
19. Barhak, J.,Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Trans. on Visualization and Computer Graphics, 7(1), pp 1-16, (2001).
20. Mishkoff H. C. Understanding Artificial Intelligence, Radio Shack, (1986).
21. Gevarter W. B. Intelligence Machines: An Introductory, Prentice-Hall, (1985).
22. Hoffman, M., Varady, L.: Free-form surfaces for scattered data by neural networks. Journal for Geometry and Graphics, 2, pp. 1-6, (1998).
23. Iglesias, A., Galvez, A.: A new Artificial Intelligence paradigm for Computer-Aided Geometric Design. Lecture Notes in Artificial Intelligence, Vol. 1930, pp. 200-213, (2001).
24. Piegl Les, On NURBS: A Survey, IEEE Computer Graphics and Applications, Vol. 11, No. 1, pp. 55 – 71, (1991).
25. Rogers David F., Rae A. Earnshaw (editors), State of the Art in Computer Graphics - Visualization and Modeling, New York, Springer-Verlag, pp. 225 – 269, (1991).
26. C. deBoor, A Practical Guide to Splines, New York, Springer-Verlag, (1978).
27. Watt Alan, Watt Mark, Advanced Animation and Rendering Techniques, New York, AMC press Addision-Wesley, (1992).
28. Foley James D. et al, Introduction to Computer Graphics, Addision-Wesley, (1994).