Freeform Surface Reconstruction Using Basis Functions Neural Networks

اعادة بناء الاسطح الحرة باستخدام الشبكات العصبية ذات الدوال الاساسية

Tawfik T. El-Midany, Ahmed Galal, Mohammed Gomaa Elkhateeb*
Prod. Eng. & Mech. Design Dept. Mansoura University, Mansoura, Egypt
*Email: elkhateeb@mans.edu.eg

الملخص

يعتبر اعادة بناء السطح خطوة رئيسة فى الهندسة العكسية. الهدف من اعادة بناء السطح هو عمل نموذج سطحى من سحابة من النقاط. هذه السحابة تم جلبها من خلال عمل مسح لنقاط سطح الجزء المراد عمل هندسة عكسية له. هذا النموذج السطحى يمكن استخدامه لاحقا فى من خلال تطبيقات استخدام الكمبيوتر فى التصميم و الانتاج من اجل انتاج هذا الجزء. فى هذا البحث تم استخدام الشبكات العصبية ذات الدوال الاساسية من اجل اعادة بناء اسطح NURBS من سحابة النقاط باقل خطأ ممكن. هذه الشبكات لها امكانية عالية و سريعة لعمل تقريب للنقاط الى اسطح NURBS. ايضا يزود هذا النوع من الشبكات بعد تدريب الشبكة نقاط التحكم و اوزانها لسطح NURBS و هو ما لايتوافر مع الشبكات العادية لتقريب الدوال. تدريب الشبكات يتم باستخدام اسلوب الانتشار للخلف. اوضحت النتائج ان هذا النوع من الشبكات يمكن تطبيقه على سحابات النقط ذات الاحجام الكبيرة و الصغيرة. اوضحت ايضا ان التقريب يمكن ان يصل الى خطأ مقبول من خلال الاختيار الصحيح لمعدل التدريب و عدد نقاط التحكم.

Abstract

Surface reconstruction is a main step in Reverse Engineering. The aim of surface reconstruction is to create a surface model from point cloud. Point cloud is acquired by digitizing the surface of the part to be reverse engineered. The obtained surface model can be further used by CAD/CAM applications for manufacturing the part. In this paper, Rational B-spline Neural Networks (RBNN) are used to reconstruct NURBS surface patches from point clouds with acceptable error. RBNN are basis function neural networks. They have higher and faster approximation ability to NURBS surface patches. Also, they provide, after training, the control points and weights of the NURBS surface which cannot be done by regular function approximation networks. Training of the network is done using the Back-Propagation algorithm. Results showed that the RBNN can be applied for small and large volume point clouds. Also, acceptable error can be reached by proper selection of the training rate and number of control points.

Key Words:
Freeform Surface, Surface Reconstruction, NURBS, Neural Networks

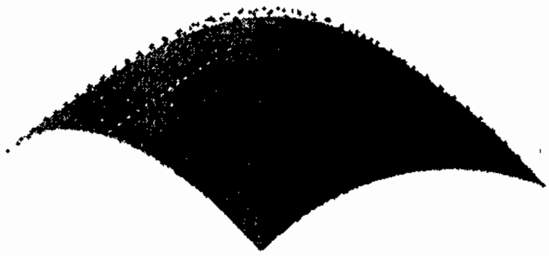
Accepted July 12, 2011.

1. Introduction

Reverse Engineering (RE) is the process of producing design details in the form of CAD model from the physical part. This model can be used in CAD/CAM applications for modification and manufacturing of the part. Applications of RE include the production of a copy of an existing part when the original drawings are not available. In other cases, one may need to re-engineer an existing part, when more analysis and/or modifications are required to construct a new improved product. RE is realized in two steps: surface digitizing and surface reconstruction. Surface digitizing measures the part surface using contact or noncontact techniques. It provides point cloud describing the surface geometry [1,2,3,4].

Surface reconstruction can be defined as the process of recovering the 3D shape of the part. It can be done by: polygonal mesh, or surface fitting. In polygonal mesh, the mesh is created by connecting adjacent points, called vertices, to form triangles or quadrilateral meshes. The produced models are not suitable for CAD/CAM applications as they are not in an engineering drawing format. They are used for rapid prototyping, laser milling, 3-D graphics, and animations [1, 5].

Interpolation



Approximation

Fig. 1. Difference between Interpolation and Approximation.

Surface fitting is the process of constructing a surface that has the best fit to a series of data points which is useful for later manipulations and applications. Surface fitting can be classified into two types: interpolation and approximation as shown in fig.1. In interpolation, the resulting surface interpolates the measured data at each data point. This is useful when the function values at the measured points are known to a high precision. In approximation, the fitted surface passes as closely as possible to the data points. Approximation is more suitable for point clouds as it smoothes residual noise that may be present in the points. Also, it reduces the great computational effort required to obtain surfaces by interpolation.

The commonly used curves and surface for surface fitting are the parametric curves and surfaces including Bezier B-Spline and NURBS. NURBS curves and surfaces are preferred in engineering design and manufacturing applications because they permit simple object-shape modification by changing only a small number of parameters, such as control points, knot vectors, or weights [6,7,13].

2. NURBS approximation

Supposing that there are point cloud containing (r x s) $\{Q_{l,k}\}$ points. It is required to approximate these points to a NURBS surface patch $S(u_l,v_k)$. Assume that $S(u_l,v_k)$ has (p x q) degrees with (m x n) control points in u and v directions. $S(u_l,v_k)$ can be defined as:

$$S(u_l,v_k)=\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} W_{i,j} P_{i,j} N_{i,p}(u_l,v_k) N_{j,q}(u_l,v_k)}{\sum_{i=1}^{m}\sum_{j=1}^{n} W_{i,j} N_{i,p}(u_l,v_k) N_{j,q}(u_l,v_k)}$$

where $\{P_{i,j}\}$ are control points forming the control grid, $\{w_{i,j}\}$ are the weights, $\{N_{i,p}(u)\}$ and $\{N_{j,q}(v)\}$are the non-rational B-spline basis functions defined over the open non uniform knot vectors:

$$\bar{U} = \{\underbrace{0,\dots,0}_{p+1}, \bar{u}_{p+2},\dots,\bar{u}_{r-p-1}, \underbrace{1,\dots,1}_{p+1}\}$$

$$\bar{V} = \{\underbrace{0,\dots,0}_{q+1}, \bar{v}_{q+2},\dots,\bar{v}_{s-q-1}, \underbrace{1,\dots,1}_{q+1}\}$$

where r = n+p+1 and s = m+q+1.

NURBS approximation means determining the control points and weights of $S(u_l,v_k)$ that minimizes the following least square expression:

$$E=\sum_{l=1}^{r}\sum_{k=1}^{s}(Q_{l,k}-S(u_l,v_k))^2$$

Before approximation, a parameterization technique should be used to assign parameter values (u_l,v_k) to each point. Chord length method is the most common method. It can be stated as following:

$$u_{1,j}=0;\quad u_{l,j}=\frac{\sum_{i=2}^{l}|D_{i,j}-D_{i-1,j}|}{\sum_{i=2}^{l}|D_{l,j}-D_{i-1,j}|}$$

$$v_{1,j}=0;\quad v_{l,j}=\frac{\sum_{i=2}^{l}|D_{i,j}-D_{i,j-1}|}{\sum_{i=2}^{l}|D_{i,j}-D_{l,j-1}|}$$

Following parameterization, knot vectors are to be determined. Knot vectors are used with parameters to calculate the B-spline basis function. To reflect the distribution of parameters, the averaging method is employed to determine the knot vectors. The internal knots can be calculated by:

$$\bar{u}_{j+p+1}=\frac{1}{p}\sum_{i=j+1}^{i+p}u_i$$

Once parameters and knot vectors are calculated, an approximation method is applied to compute the control points and weights of the $S(u_l,v_k)$ which minimizes the following expression:

$$E=\sum_{l=1}^{r}\sum_{k=1}^{s}(Q_{l,k}-S(u_l,v_k))^2$$

Approximation methods for NURBS surface are divided into two categories: direct methods, and surface skinning. Direct methods consist of simultaneously estimating the unknown control points of the surface and weights. Surface skinning, also known as lofting, is a process of passing a smooth surface through a set of so called cross-sectional curves. These curves approximate the measured points. They must have the same degree and knot vectors. This leads to high number of control points making the process more complex and time consuming than direct methods [9,10].

Direct methods include: least square methods (LSM), optimization techniques, and neural networks. The LSM still the most commonly used method. Weiyin Ma and J P Kruth [11] used the LSM to approximate measured points to a B-Spline surface. Ma and Kruth [12] used two LSM algorithms for NURBS curves and surfaces fitting which automatically identify the control points and their respective weights. The weights of control points are first identified through singular value decomposition, the control points are then determined by least squares minimization. It was found that the surface error produced by LSM was higher. Djordje Brujic et al. [13] used LSM modified by sparsity structures of the relevant matrices to the problem and regularization terms. This improves the computational complexity and saves time producing lower errors. As parameterization and knot determination are approximated process, these are reflected in the surface error. To improve the performance of LSM, several algorithms were developed to improve parameters. Aziguli Wulamu et al. [14] developed a parameterization called "Adaptive Equidistant Parameterization". The surface errors obtained from this algorithm was lower than those of chord length and equidistant methods. Kunal Soni, Daniel Chen, and Terence Lerch [15] used feature based parameterization.

Jiing-Yih Lai, Chiou-Yuan Lu [5] used the cumulative chord-length method to assign the initial parameters to the points. Then the Powel's method is applied to optimize the parameters at each iteration until the desired error is reached. Mohammed Riyazuddin [16] used LSM to find the control points of the NURBS surface. Then simulated annealing was applied to optimize the values of weights then the knot vectors.

Meerja Huma [17] used the Simulated Evolution algorithm to optimize the values of knots and weights. Akemi Gálvez et al. [18] used genetic algorithm to find the parameters of the points. Akemi Gálveza et al. [19] used two step genetic algorithm. The first is for parameterization and the second for knot calculations. Nallig Leal et al. [20] used evolutionary strategy to obtain the weights of the NURBS surface so that the distance between the point cloud and the NURBS is minimal. Akemi Gálvez and Andrés Iglesias [21] used particle swarm optimization with LU decomposition to compute parameters, knot vectors, control points and weights.

Neural networks are used for surface approximation due to their ability to function approximation. Regular function approximation neural networks include: Multi-Layer Feed-Forward (MLF), and Radial Basis Function (RBF). P Gu and X Yan [22] used Multi-Layer Feed-Forward neural network to approximate parameterized points generated from NURBS surface. Ju H. et al [4] used RBF neural network to approximate parameterized points acquired by laser scanning freeform surface. These networks can be used only, after training, to calculate points on the approximated surface. Although MLF and RBF neural networks have higher approximation ability, they do not provide a solution capable of integration with CAD/CAM applications. As they do not provide the control points and weights of the surface which define it in CAD/CAM applications to allow manipulation and applications. Also, using the generated points for data transfer between packages is difficult and slower than using the parameters of the surface.

To overcome the stated problems, parametric basis function neural networks were developed. They are similar in architecture to MLF neural networks. They have fast approximation ability to parametric curves or surfaces due to using parametric basis functions as activation functions. Also, they provide control points and weights, after training, as they are represented in the network by the connecting weights. George K. Knopfa and Jonathan Kofman [7] first used this type of neural networks to approximate points to Bezier surface. Xiaogang G. [8] used B-Spline basis function neural network to approximate points to B-Spline curves and surfaces.

3. Rational B-spline Neural Network (RBNN)

RBNN are modified B-Spline basis function neural networks which can be used to approximate points to NURBS curves or surfaces faster. Also, the NURBS control points and weights can be provided after training to reconstruct the surface in CAD/CAM packages. It is noted that the special architectures enable the network to have a relationship between the inputs and outputs of the network to the relation between the

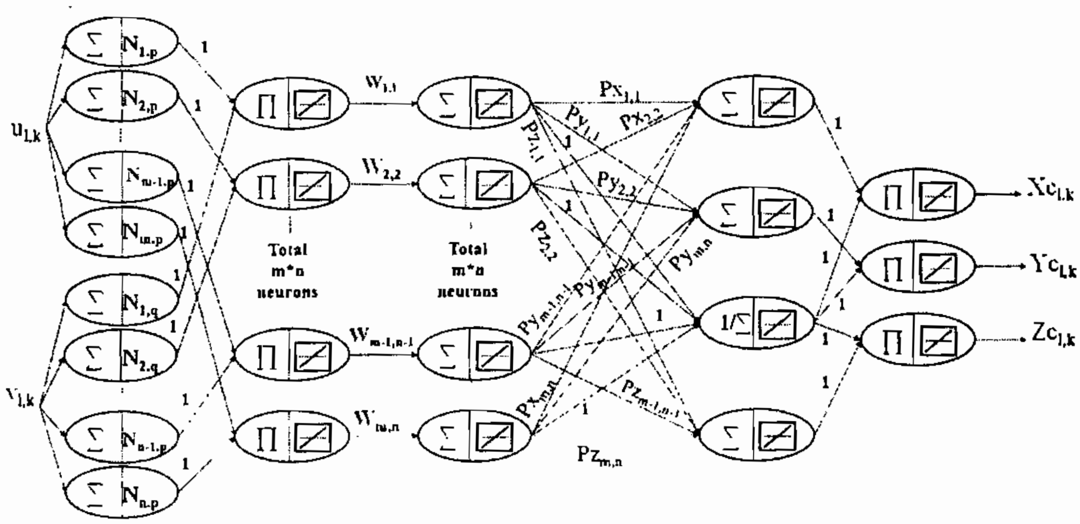


Fig. 2. Architecture of surface approximation RBNN

NURBS parameters and coordinates.

3.1 Surface RBNN Architecture

The architecture of surface approximation RBNN is shown in Fig. 2. This architecture has two inputs, parameters u and v of a point, and three outputs which are approximated $Xc_{l,k}$, $Yc_{l,k}$, and $Zc_{l,k}$ coordinates of the point. The network has five layers. The number of neurons in the first layer equals the sum of the number of neurons (m + n) in both directions u and v. The second and third layer have (m x n) neurons, the forth layer has four neurons, and the fifth layer has 3 neurons. The net input functions are linear sum in the first, third, and forth layers which the second and fifth have delta function. The activation functions linear for all layers except the first has B-Spline basis function. According to fig. 2, the connecting weights between the first and the second layers are unity and are fixed during training. The connecting weights between the second and third layer represent the weights of the NURBS weights, $W_{i,j}$, and they are updated at every training cycle (epoch). The connecting weights between neurons in the third layer and the first, second, and forth neurons in the forth layer represent the X, Y, and Z coordinates of the control points ($Px_{i,j}$ – $Py_{i,j}$ – $Pz_{i,j}$) respectively. These connections are also updated at every epoch. The rest of connections have unity and are fixed during training. By starting the networks, it performs linear sum of the terms of the networks from the inputs to the outputs. The outputs are calculated by:

$$Xc_{l,k} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} W_{i,j} Px_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}{\sum_{i=1}^{n}\sum_{j=1}^{m} W_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}$$

$$Yc_{l,k} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} W_{i,j} Py_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}{\sum_{i=1}^{n}\sum_{j=1}^{m} W_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}$$

$$Zc_{l,k} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j} Pz_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}{\sum_{i=1}^{n}\sum_{j=1}^{m} w_{i,j} N_{i,p}(u_{l,k}) N_{j,q}(v_{l,k})}$$

3.2 Training

During training, the connecting weights of the network are iteratively adjusted to minimize the network performance function. Here, the performance function for the network is the mean square error between the network outputs and the target outputs. The outputs are the approximated $Xc_{l,k}$, $Yc_{l,k}$, and $Zc_{l,k}$ coordinates of the points, while the targets are the measured $Xt_{l,k}$, $Yt_{l,k}$, and $Zt_{l,k}$ coordinates of the points.

For (r x s) data points, the performance function can be calculated by:

$$E = \sum_{l=1}^{r}\sum_{k=1}^{s}\left(\left(Xt_{l,k} - Xc_{l,k}\right)^2 + \left(Yt_{l,k} - Yc_{l,k}\right)^2 + \left(Zt_{l,k} - Zc_{l,k}\right)^2\right)$$

By applying the back-propagation training algorithm, the learning rules for epoch k+1 can be expressed as:

$$W_{i,j}(k+1) = W_{i,j}(k) - \eta\left(\frac{\partial E}{\partial W_{i,j}}\right)$$

$$Px_{i,j}(k+1) = Px_{i,j}(k) - \eta\left(\frac{\partial E}{\partial Px_{i,j}}\right)$$

$$Py_{i,j}(k+1) = Py_{i,j}(k) - \eta\left(\frac{\partial E}{\partial Py_{i,j}}\right)$$

$$Pz_{i,j}(k+1) = Pz_{i,j}(k) - \eta\left(\frac{\partial E}{\partial Pz_{i,j}}\right)$$

where η represents the training rate. The learning rate is held constant throughout the training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable and, if the learning rate is too small, the algorithm will take too long to converge. Fig. 3. shows the flow chart of the algorithm followed for training the network to reach the desired error.

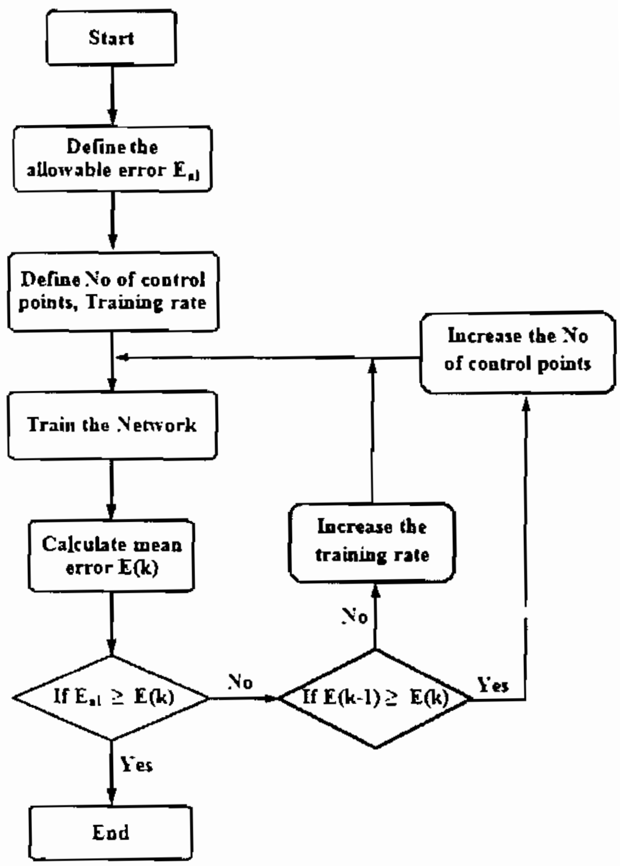
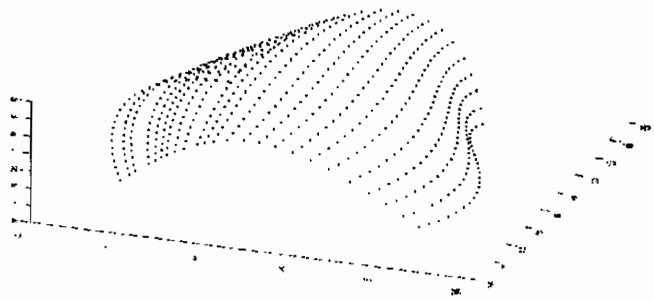


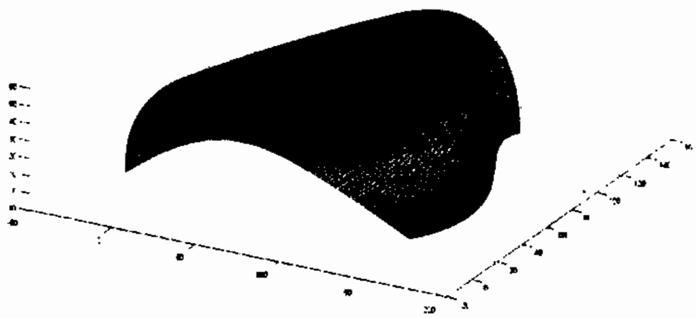
Fig. 3: Flow chart of the algorithm followed for training RBNN to reach the desired error

4. Case Study

To validate the developed algorithm, it is applied 600 data points obtained from bicubic B-spline surface as shown in Fig. 5-a. These points are to be fitted to a bicubic NURBS surface. It is found that the mean square error depends largely on the number of control points of the fitted surface and the training rate. From experiments, the points are fitted with mean error of 15.6 $\times 10^{-6}$. The fitted surface is shown in Fig.5-b. To get this error, the number of control points in both u and v directions were (11 x 11) and the training rate was 0.1169. This is due to the large area of the surface and its variable curvature. The time consumed for the process depends on the volume of data points, and the number of control points. The code was written using MATLAB. Training curve is shown in Fig. 4. The time taken was 18 seconds in 100 iterations using 2.5/4 core 2 Quad processor.



(a)



(b)

Fig. 4: (a) 600 regular data point (b) Approximated NURBS surface.

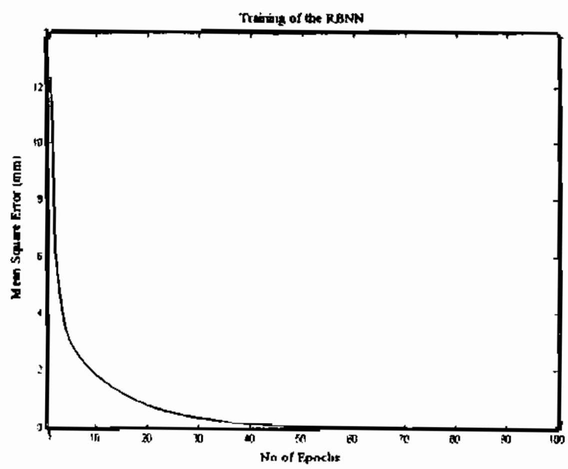


Fig. 5: Training curve of the RBNN

5. Conclusion

Freeform surface reconstruction is a major challenge in the Reverse Engineering. The goal of surface reconstruction is to create CAD model of the part from point cloud. NURBS surfaces are widely used in the reconstruction of freeform surfaces. This is due to easy shape manipulation by control points, weights, and knot vectors. In this paper, RBNN are used for NURBS surface approximation due to their high approximation ability to NURBS surfaces. After training, RBNN provide the control points and weights of the NURBS surface which is not available in other networks. The high approximation ability of RBNN eliminates the need for parameters or knot optimization. The performance of the network depends on two factors: the number of control points and the training rate. By increasing the number of control error the mean square error. It is found that the optimum value of training rate depends on the volume of point clouds, and the number of control points. Large volume point clouds require higher training rate while smaller require smaller training rate. Also, increasing the number of control points should be accompanied with an increase in the training rate. Therefore, the values of the number of control points and training rate should be balanced together to achieve optimum performance for the network.

6. References

[1] Vinesh Raja and Kiran J. Fernandes, "Reverse Engineering An Industrial Perspective", Springer Series in Advanced Manufacturing. 2008.

[2] E. Savio, L. De Chiffre, R. Schmitt," Metrology of freeform shaped parts", CIRP Annals, Vol. 56 No.2. PP810-835, 2007.

[3] J.-P. Pernot, B. Falcidieno, F. Giannini, J.-C. Leon," Incorporating free-form features in aesthetic and engineering product design: State-of-the-art report", Computers in Industry, Vol. 59 No.7, PP626-637, 2008.

[4] Ju Hua, Wang Wen, Xie Jin, Chen Zi-chen, "Neural Network Approach for modification and fitting of digitized data in Reverse Engineering", Journal of Zhejiag University Science, Vol. 5 No. 1, PP75-80, 2004.

[5] Jiing-Yih Lai, Chiou-Yuan Lu, "Reverse Engineering of Composite Sculptured Surfaces", Int J Adv Manuf Technol, Vol. 12, PP180-189, 1996.

[6] Pramod N Chivate, Andrei G Jablokow, "Review of surface representations and fitting for reverse engineering", Computer Integrated Manufacturing Systems, Vol. 8 No. 3, PP193-204, 1998.

[7] George K. Knopfa, Jonathan Kofman, "Adaptive reconstruction of free-form surfaces using Bernstein basis function networks", Engineering Applications of Artificial Intelligence, Vol. 14 No. 5, PP577–588, 2001.

[8] Xiaogang Guo, "Reconstruction of parametric curves and surfaces using an adaptive basis function network", M.Sc Thesis, Faculty of Graduate Studies, The University of Western Ontario, 1998.

[9] K. R. Koch, "Identity of simultaneous estimates of control points and of their estimates by the lofting method for NURBS surface fitting", Int J Adv Manuf Technol, Vol. 44 No. 11-12. PP1175-1180, 2009

[10] S.M.Shamsuddin, M.A.Ahmed, Y.Samian, "NURBS skinning surface for ship hull design based on new parameterization method", Int J Adv Manuf Technol, Vol. 28 No. 9-10, PP 936-941, 2006.

[11] Weiyin Ma, J P Kruth, "Parameterization of randomly measured points for least squares fitting of B-Spline curves and surfaces", Computer-Aided Design, Vol. 27 No.9 ,PP663-675, 1995.

[12] Ma W, Kruth JP, "NURBS curve and surface fitting for reverse engineering", International Advanced Manufacturing Technology, Vol 14 No. 12, PP918–27, 1998.

[13] Djordje Brujic, Iain Ainsworth, Mihailo Ristic, "Fast and accurate NURBS fitting for reverse engineering, Int J Adv Manuf Technol, Vol. 41 No. 9-10, PP 948-959, 2009.

[14] Aziguli Wulamu, Goetting Marc, Zeckzer Dirk, "Approximation of NURBS Curves and Surfaces Using Adaptive Equidistant Parameterizations", Tsinghua Science And Technology, Vol. 10 No. 3, PP 316-322, 2005.

[15] Kunal Soni, Daniel Chen, Terence Lerch, "Parameterization of prismatic shapes and reconstruction of free-form shapes in reverse engineering", Int J Adv Manuf Technol, Vol. 41 No. 9-10, PP 948-959, 2009.

[16] Mohammed Riyazuddin, "Visualization with NURBS using simulated annealing optimization technique", M.Sc Thesis, King Fahd University Of Petroleum and Minerals, 2004.

[17] Meerja Huma, "Evolutionary heuristic optimization for digital curves and surfaces", M.Sc Thesis, King Fahd University Of Petroleum and Minerals, 2005.

[18] Akemi Gálvez, Andrés Iglesias, Angel Cobo, Jaime Puig-Pey, "Jesús Espinola. Bézier Curve and Surface Fitting of 3D Point Clouds Through Genetic Algorithms, Functional Networks and Least-Squares Approximation", Lecture Notes in Computer Science, Vol. 4706, PP680-693, 2007.

[19] Akemi Gálveza, Andrés Iglesias, Jaime Puig-Peya, "Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction", Information Sciences, In Press, Available online 8 October 2010.

[20] Nallig Leal, Esmeide Leal, and John William Branch, "Simple Method for Constructing NURBS Surfaces from Unorganized Points", Proceedings of the 19th International Meshing Roundtable 2010, Part 3, PP 161-175, 2010.

[21] Akemi Gálvez, Andrés Iglesias, "Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points", Information Sciences, In Press, Available online 13 November 2010.

[22] P Gu, X Yan, "Neural network approach to the reconstruction of freeform surfaces for reverse engineering", Computer-Aided Design, Vol. 27 No.1, PP 59-64, 1995.