Neural Networks Based Recognition of 3D Freeform Surface from 2D Sketch

Guangmin Sun, S. F. Qin, and D. K. Wright

Abstract — In this paper, the Back Propagation (BP) network and Radial Basis Function (RBF) neural network are employed to recognize and reconstruct 3D freeform surface from 2D freehand sketch. Some tests and comparison experiments have been made to evaluate the performance for the reconstruction of freeform surfaces of both networks using simulation data. The experimental results show that both BP and RBF based freeform surface reconstruction methods are feasible; and the RBF network performed better. The RBF average point error between the reconstructed 3D surface data and the desired 3D surface data is less than 0.05 over all our 75 test sample data.

Keywords — Artificial intelligence, Freeform surface recognition, Neural networks, Sketch design.

I. INTRODUCTION

IN recent years, Artificial Neural networks (ANN) have been widely applied to the function approximation, nonlinear mapping, prediction and pattern recognition problem. ANNs are effective in these applications because of their learning capabilities. For instance, they can be used for solving the surface and vertex corresponding problems in multiple-view-based 3D object recognition systems [1], for classifying 3D objects from 2D images [2] and for freeform surface reconstruction from range images in reverse engineering [3][4]. However, interpretation of 3D freeform surfaces from 2D sketches using ANN has little report.

Freeform surface interpretation from 2D sketches is more difficult than recovering 3D polyhedra. The fundamental difficulty in recognising unknown 3D objects from 2D freeform sketches (or images) is that any 2D sketch is freeform and its appearance varies with different viewpoints. Typically, 2D sketches are drawn from an unknown viewpoint. Current solutions for polyhedra such as line-labelling scheme [5][6], the gradient space approach [7], the linear System approach [8], the optimization method [9], and the 3D-2D geometric correlation method [10] are not suitable for freeform surface interpretation problems. Therefore, gesture-based systems [11][12] have been developed to interactively create 3D freeform surface models. But the gesture-based method there seems some problem for general open surfaces.

In this paper, we describe the development of neural networks-based freeform surface recognition and reconstruction method from 2D freehand sketches. The BP network and RBF network are employed to recognize and restructure the 3D freeform surface from 2D freehand sketch. The scheme of 3D freeform surface recognition and reconstruction is that a neural network is training by using some initial normalised 3D surface data and their corresponding 2D projection data, and then the unknown 2D sketches are normalised and sent into the well-trained neural network for automatically generating a corresponding 3D freeform surface. The method has been tested with a range of data and it gives satisfactory result.

The rest of this paper is organised as follows. In Section 2, the structure and learning algorithm of BP and RBF neural networks are introduced. Then the simulation data preparation and sketch feature extraction is presented in Sections 3. The experimental results and performance comparison are given in Section 4. Finally, conclusions are drawn in Section 5.

II. THE BP AND RBF NETWORK STRUCTURE AND LEARNING ALGORITHM

The BP and RBF neural networks are a special class of the general multi-layer feed-forward neural network model. The kind of neural network is the static system. The neurons of a network are arranged into layers. Information propagates in a forward direction from the network input to the network outputs. The classical BP and RBF network is simply composed of three layers - one input layer, one hidden layer and one output layer - as shown in Fig. 1.

The nodes in the input layer pass the input data directly to the nodes in the hidden layer. The hidden layer is fully connected to the input layer and produces localized responses to the inputs. These hidden nodes perform significant nonlinear data transformation for output nodes in order to produce arbitrary output functions. For the BP network, the neurons in the hidden layer have a sigmoid activity function and their input output relationship is:

$$Z = f(X) = \frac{1}{1 + \exp(-W_1 X)} \quad (1)$$

where $X = \{x_1, x_2, \ldots, x_N\}$ is the input pattern vector, W_1 is the weight vector between the input layer and the hidden layer. The value of Z is in the range of 0~1, and depends on the comparability between X and W_1.

This work has been supported by the EPSRC research grant of UK (GR/S01712/01) and the State Scholarship Fund of China.

Guangmin Sun is with the Department of Electronic Engineering, Beijing University of Technology, China. He is now working at Brunel University as a visiting scholar. (10 Hows Road, Uxbridge, Middlesex, UB8 2AR, UK, Tel: 44-1895-266811, Fax: 44-1895-269763, email: gmsun@bjut.edu.cn)

S. F. Qin is with the School of Engineering & Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. (email: Sheng.Feng.Qin@brunel.ac.uk)

D. K. Wright is with the School of Engineering & Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. (email: David.Wright@brunel.ac.uk).

1-4244-0049-X/05/$20.00 ©2005 IEEE

But for the RBF network, the neurons in the hidden layer have a Gaussian activity function and their input output relationship is:

$$Z = f(X) = \exp\left(-\frac{\|X-C\|^2}{2\sigma^2}\right) \quad (2)$$

where $X=\{x_1, x_2, \ldots, x_N\}$ is the input pattern vector. C is the kernel vector of the Gaussian function, it can also be seen as the weight vector between the input layer and the hidden layer. σ is the spread parameter of the Gaussian function, through which we can control the receptive field of that neuron. The value of Z is in the range of 0~1, and depends on the distance between X and C. The closer the distance is, the larger the output of the basis function is.

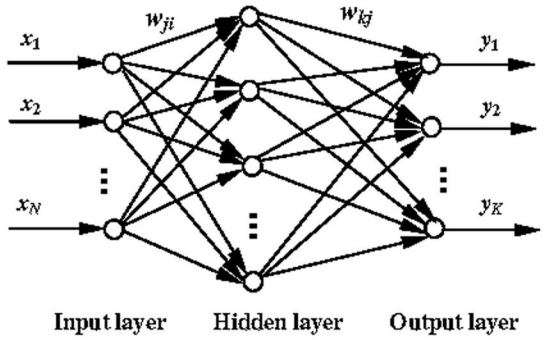

Fig. 1. The Structure of the BP or RBF neural network

In both kinds of networks, the output layer is fully connected to the hidden layer. The nodes in the output layer summarise the hidden layer output values with weights. After the output values of the hidden nodes have been computed, the values for the output layer nodes can be calculated by:

$$Y = W_2 Z \quad (3)$$

where W_2 is the weight vector between the hidden layer and the output layer. The outstanding issue associated with the development of a neural network is the network structure determination and the parameters selection. In our research, the network structure is fixed in advance, and then the parameters, including the connection weights and the parameters of kernel function, are adjusted through training.

The numbers of neurons in input and output layer are easy to determine and they are dependent on the task. The neuron number in the input layer is consistent with the dimension of the input feature vector and the neuron number in the output layer is consistent with the dimension of the desired output feature. The remaining unknown parameter about the network structure is the number of neuron in the hidden layer. It is important to determine the neuron number of the hidden layer because it determines the nonlinear behaviour of the network. In general, it can be set according to some heuristic considerations or experience. The more hidden nodes are used, the more accurate the approximation is. If the number of hidden nodes is too small, the network cannot approximate the underlying function accurately. On the other hand, if too many hidden nodes are used, the network will overfit the training samples and results in poor generalization. In our research, the optimum number of hidden units is determined by experiments.

The parameter values of the kernel function and the connection weight vectors of the networks are determined through the back-propagation learning algorithm. The learning process consists of two phases, feed-forward and back-propagation. During training, an input vector X is fed to the network and propagated to the final layer, then the output is compared with the desired output and the error is back-propagated, so that the parameter values and weights can be adjusted.

The back-propagation learning algorithm is shown as:

(1) Initialization: C, σ and W are initially set by some random values in the range [0, 1].

(2) Forward pass: Arbitrarily choose the input feature vector $X=[x_1, x_2, \ldots, x_N]^T$ and the desired output $D=[d_1, d_2, \ldots, d_k]^T$ from the training sample set and feed it to the network, compute the network outputs Y by proceeding forward through the network, layer by layer.

(3) Backward pass: Calculate the sum-squared error E and error gradients versus the parameters $\frac{\partial E}{\partial W_1}$, $\frac{\partial E}{\partial W_2}$ (for BP), or $\frac{\partial E}{\partial C}$, $\frac{\partial E}{\partial \sigma}$, $\frac{\partial E}{\partial W_2}$ (for RBF), layer by layer, starting from the output layer and proceeding backwards:

$$E = \frac{1}{2}(Y-D)^2 \quad (4)$$

(4) Update parameters through an iterative process:

For BP network:

$$W_2(n+1) = W_2(n) - \eta_1(n)\frac{\partial E}{\partial W_2} \quad (5)$$

$$W_1(n+1) = W_1(n) - \eta_2(n)\frac{\partial E}{\partial W_1} \quad (6)$$

For RBF network:

$$W_2(n+1) = W_2(n) - \eta_3(n)\frac{\partial E}{\partial W_2} \quad (7)$$

$$C(n+1) = C(n) - \eta_4(n)\frac{\partial E}{\partial C} \quad (8)$$

$$\sigma(n+1) = \sigma(n) - \eta_5(n)\frac{\partial E}{\partial \sigma} \quad (9)$$

Where n is the current number of epoch. $\eta_1(n)$, $\eta_2(n)$, $\eta_3(n)$, $\eta_4(n)$, and $\eta_5(n)$, are rates of learning with respect to corresponding parameters. They are adjusted with the learning and their initial values are 0.8.

(5) Repeat the algorithm for all training samples, if one epoch of training is finished, repeat the training for another epoch, until the precision or the training times reach their predetermined values.

After training, the well-trained neural networks can be used for recognising unknown 3D freeform surface from 2D sketching data.

III. THE GENERATION OF SIMULATION DATA FOR 3D FREEFORM SURFACE RECOGNITION

In order to use neural networks for 3D freeform surface recognition, we generate a set of simulation data between

known 3D freeform surfaces and their 2D isometric projections. The 2D and 3D coordinates are directly used as feature data. Currently, in our system, every 3D freeform surface consists of 4 three-dimension curves. For each 3D curve, we specify it by four 3D edit points on the curve and then generate a standard cubic Coons curve with 11 parameter points. All these 11 parameter points are isometric projected on the 2D projection plan. In order to mimic unknown view points in the real world, we randomly vary the positions and distance of 3D edit points and allow the varied 3D edit points have rigid rotations between –30 to +30 degrees about X, Y, Z axes respectively. All the resultant pairs of the curve's 3D coordinates data (x_i, y_i, z_i | i=1, 2, … ,11) and corresponding 2D coordinate data (x_i, y_i | i=1, 2, … ,11) have been normalized and formed the simulation data set to use for our network training and performance test.

In the simulation data set, the dimension of 3D freeform surfaces data is 132 and the dimension of their corresponding 2D projection data is 88. We divide randomly all the simulation data into two groups. The first group, as a training sample set, is composed of 300 pairs of input and output data. The 2D projection data is used as input and corresponding 3D data as desired output. The second group is used for network testing. It is composed of 75 pairs of input and output data.

IV. EXPERIMENTAL RESULTS AND PERFORMANCE COMPARISON

In our research, because the dimensions of the 2D input data and the 3D output data are 88 and 132 respectively, the neuron numbers of the input layer and output layer are N=88 and K=132 correspondingly. In order to compare the performance of BP and RBF networks, the hidden neuron numbers of the both networks are selected the same as 100. In fact, the experimental results have proved that, for the RBF network, the more hidden neurons are used, the more accurate the approximation is, even it can result in the exact approximation with zero error for the training samples. But for the BP network, with the increasing of hidden neuron number, it cannot obviously improve the precision of result, and if too many hidden nodes are used, it will over-fit the training samples and results in poor generalization and slower convergence.

After the structures of networks have been determined, the training sample set is utilized to train the both networks according to the Back-propagation learning algorithm mentioned in section 2. Then the recognition and reconstruction performance of the BP and RBF network are tested and compared with the test sample set. Finally, some experimental results are given.

Fig 2 and Fig. 3 show the change curves of the sum-squared error of both networks during training. Given the same error threshold, the BP and RBF networks performed differently. The BP network cannot converge to the threshold until 10,000 epochs, which is pre-set for stopping iteration. In contrast, the RBF network only ran through 2,741 epochs to reach the threshold.

Fig.4 shows the recognition result of the BP network for an un-training surface. Fig.5 illustrates the recognition

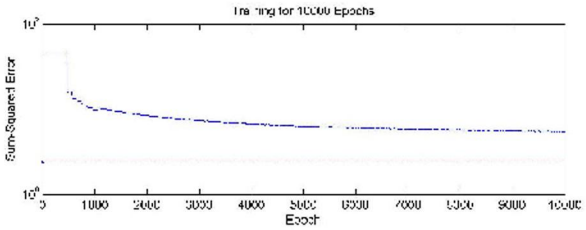

Fig. 2. The changes of the sum-squared error and learning rate from the BP network

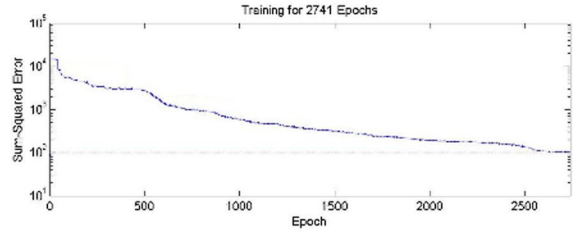

Fig. 3. The changes of the sum-squared error and learning rate from the RBF network

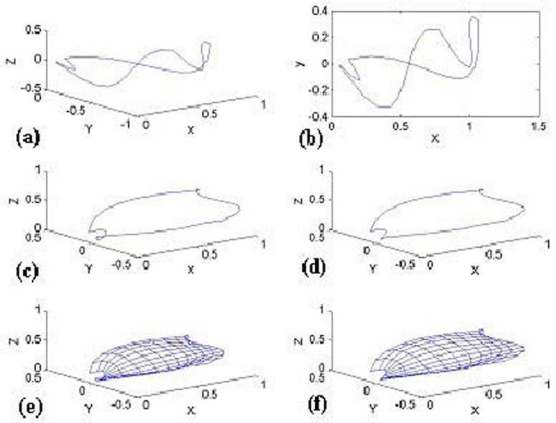

Fig.4. The recognition results of the BP network for an un-training surface: (a) The isometric view of 3D freeform surface boundaries; (b) The 2D projection of the 3D freeform surface boundaries; (c) The 3D freeform boundaries in the world coordinate system; (d) The 3D freeform boundaries reconstructed by the BP network; (e) The original 3D freeform surface; (f) The 3D freeform surface reconstructed by the BP network.

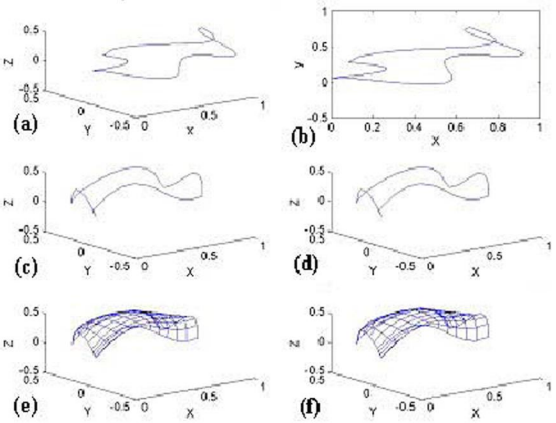

Fig.5. The recognition results of the RBF network for an un-training surface: (a) The isometric view of 3D freeform surface boundaries; (b) The 2D projection of the 3D freeform surface boundaries; (c) The 3D freeform boundaries in the world coordinate system; (d) The 3D freeform boundaries reconstructed by the RBF network; (e) The original 3D freeform surface; (f) The 3D freeform surface reconstructed by the RBF network.

result of the RBF network for an un-training surface. From the experiments above and the comparison, we can see that the proposed recognition and reconstruction methods of 3D freeform surface based on BP or RBF neural networks are feasible. And the RBF network performs better than the BP network. The training speed and reconstruction precision of the RBF network are higher than that of the BP network. For the RBF network, the average point error between the reconstructed 3D surface data and the desired 3D surface data is less than 0.05 over all our 75 test sample data.

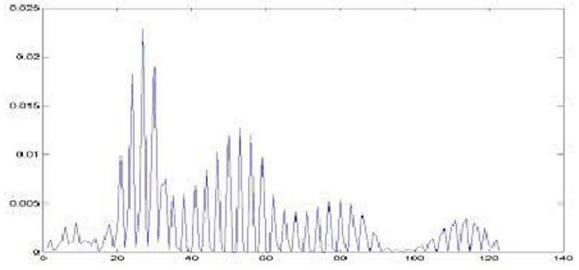

Fig.6. The curve of point errors between the reconstructed 3D surface data and the desired 3D surface data

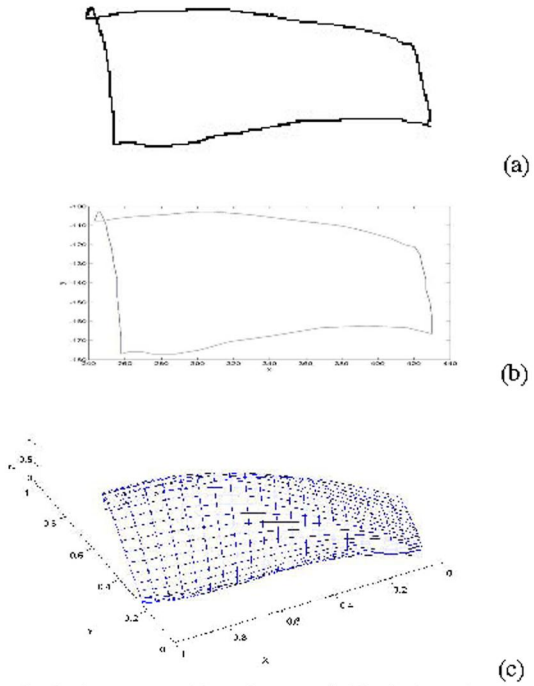

Fig.7. Surface recognition from real 2D design sketches: (a) Original freehand 2D sketch; (b) Extracted 44 point data from the four sides of original freehand 2D sketch; (c) The reconstructed 3D freeform surface by RBF NN.

Fig.6 shows the point error between the reconstructed 3D surface data and the desired 3D surface data.

Fig.7 shows a 3D reconstruction example of on line 2D sketch input by using RBF network. Original freehand 2D sketch (Fig. 7a) has been extracted with 44 points on its four sides (Fig. 7b). Finally, the RBF network gives the corresponding 3D freeform surface output (Fig. 7c). The result is very satisfactory.

V. CONCLUSION

The neural networks based 3D freeform surface recognition and reconstruction method from 2D sketches (or 2D drawings) has been presented in this paper. The BP and RBF networks are employed to recognize and restructure the 3D freeform surface from 2D freehand sketch. The testing and comparison results show that both BP and RBF based freeform surface reconstruction methods are feasible; and the RBF network performed better. The RBF average point error between the reconstructed 3D surface data and the desired 3D surface data is less than 0.05 over all our 75 test sample data. Based on the comparison result, the RBF network is finally chosen to use in a surface recognition example and the result is very satisfactory. From all the promising results, it is believed that the RBF network based freeform surface reconstruction method is applicable.

REFERENCES

[1] DESCHENES S., SHENG Y., and Chevrette P.C.: Three-dimensional object recognition from two dimensinal images using wavelet transforms and neural networks. Opt. Eng. 37(3) (1998), 763-770.

[2] GU P., YAN X.: Neural network approach to the reconstruction of freeform surfaces for reverse engineering. Computer Aided Design 27 (1) (1995), 59-64.

[3] BARHAK J., FISCHER A.: Adaptive reconstruction of freeform objects with 3D SOM neural network grids. Computers & Graphics 26 (2002), 745-752.

[4] POGGIO T., GIROSO F.: Regularization algorithms for learning that are equivalent to multi-layer networks. Sci. 247 (1990), 978-982.

[5] HUFFMAN D.A.: Impossible objects as nonsense sentences. In: Meltzer, Michie D., editors. Machine Intelligence, Edinburgh University Press. (1971), 295-323.

[6] CLOWES M.B.: On seeing things. Artificial Intelligence 2(1) (1971), 79-112.

[7] MACKWORTH A.K.: Interpreting Pictures of Polyhedral Scenes. Artificial Intelligence Vol. 4 (1973), pp. 121-137.

[8] VARLEY P.A.C., SUSUKI H., MITANI J., MARTIN R.R.: Interpretation of Single Sketch Input for Mesh& Solid Models, International Journal of Shape Modeling 6(2) (2000), 207-240.

[9] LIPSON H., SHPITALNI M.: Optimisation-based reconstruction of a 3D object from a single freehand line drawing. Computer-Aided Design 28(8) (1996), 651-663.

[10] LIPSON H., SHPITALNI M.: Correlation-based reconstruction of a 3D object from a single freehand sketch. AAAI Spring Symposium on Sketch Understanding, AAAI Press, Melno Park, CA. (2002), pp. 99-104.

[11] ZELEZNIK R.C., HERNDON K.P., and Hughes J.F.: SKETCH: an interface for sketching 3D scenes. SIGGRAPH Computer Graphics Proceedings (1996),163-170.

[12] EDELMAN S.: On learning to recognize 3D objects from examples. IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (8) (1993), 833-837.