EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

2D Sketch Based Recognition of 3D freeform Shapes by Using the RBF Neural Network

S.F. Qin, Guangmin Sun*, D.K. Wright, S. Lim, U. Khan, C. Mao

School of Engineering & Design, Brunel University, Uxbridge, Middlesex UB8 3PH, UK
*Department of Electronic Engineering, Beijing University of Technology, 100022, Beijing, PR China

Abstract

This paper presents a novel free-form surface recognition method from 2D freehand sketching. The approach is based on the Radial basis function (RBF), an artificial intelligence technique. A simple three-layered network has been designed and constructed. After training and testing with two types of surfaces (four sided boundary surfaces and four close section surfaces), it has been shown that the method is useful in freeform surface recognition. The testing results are very satisfactory.

Categories and Subject Descriptors (according to ACM CSS): H.5.2 [Information Interfaces and Presentation]: Graphical user interfaces (GUI); I.2.10 [Artificial Intelligence]: surface modelling.

1. Introduction

Shape design plays an important role in a product's commercial competitiveness. Currently, there exists little computational support for the early stages of form design as in conceptual design [QWJ01]. The early shape design process has the characteristics of fuzzy problems, tolerating high degrees of uncertainty and ambiguity. Various CAD systems have been developed to support 2D drafting and 3D modelling of products, but they usually require complete, concrete and precise definitions on the geometry, which are only available at the end of the design process [GD00]. Therefore, at the early design stage, there is a strong need of intuitive and efficient geometric modelling tools for designers to effectively express, communicate and record their new ideas.

Freeform surface interpretation from 2D sketches is more difficult than recovering 3D polyhedra. Current solutions for polyhedra such as line-labelling schemes [Huf71][Clo71], The gradient space approach [Mac73], The linear System approach [VSMM00], and optimization method [LS96] are not suitable for freeform surface interpretation problems. Interpretation of drawings containing curved objects presents several unsolved problems because that many of the simplifying assumptions made in interpreting polyhedra do not hold for curved objects [VTMH04]. For example, for a curved object, a curved line may change from convex to occluding. It cannot have a consistent line label. Therefore, gesture-based systems [ZHH96][IMT99] have been developed to interactively create 3D freeform surface models. However, in order to deal with general engineering surfaces, the gesture-based method seems problematic.

In the vision community, there has been extensive research on object detection and classification based on learning [Ede93][RYA02]. The learning process must start with the selection of a class from which the concepts to be learned will be drawn. This selection poses the difficult problem of achieving a compromise between the conflicting requirements of description and generalization. On one hand, the concepts are required to be sufficiently expressive to describe faithfully the target patterns and to capture any fine distinctions that may be present among them. On the other hand, concepts whose descriptions must be learned from examples and, at the same time, support generalization to novel situations should be kept as simple as possible.

In recent years, Artificial Neural Networks (ANN) have been widely applied to the function approximation, non-linear mapping, prediction and pattern recognition problems. Neural networks are effective in these applications because of their learning capabilities. Artificial neural networks (ANN) techniques are biologically

inspired models analogous to the basic functions of biological neurons. They have been widely utilized to implement learning processes or solving the surface and vertex corresponding problems in multiple-view-based 3D object recognition systems [LLTL91], for classifying 3D objects from 2D images [DSC98], and for freeform surface reconstruction from range images in reverse engineering [GY95][BF02][PG90]. However, there are little reports of interpretation of freeform surfaces from 2D sketches using ANN.

This paper describes the development of the artificial neural network (ANN)-based freeform surface recognition method from on-line 2D sketches. In this study, the RBF network was selected as the recogniser because of its renowned function approximation, non-linear mapping and identification ability. The fundamental difficulty in recognising unknown 3D objects from 2D sketches (or images) is that one set of 2D sketches (a projection) may correspond to infinite numbers of 3D objects. Moreover, the sketches are typically drawn from an unknown viewpoint. Their appearance varies with different users and drawing skills. Therefore, the recognition algorithms need to be very robust.

The goal of the proposed freeform shape recognition and reconstruction method is to perform robust interpretation in 3D to the 2D sketches regardless of the object orientation. The target applications are shape/form designs in product/industrial design. The shape recogniser is a RBF neural network, which is trained by using some initial normalised 3D shape data and their corresponding 2D projection data, and then the unknown 2D sketches are normalised and sent into the 3D shape recogniser—the trained RBF neural network for automatically generating a corresponding 3D freeform shape. The method has been tested with a range of data and it gives satisfactory results.

The rest of this paper is organised as follows. The RBF network is introduced in Section 2. In Sections 3 and 4, the training data preparation and experimental results and performance tests are presented respectively. Finally, conclusions are drawn in Section 5.

2. The RBF neural network structure and learning algorithm

Radial basis function neural networks are special classes of the general feedforward neural network models. A 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. Generally, the neurons in the hidden layer have a Gaussian activity function and their input – output relationship is:

$$z_j = f(X_p) = \exp\left(-b_j \left\| X_p - C_j \right\|^2\right) \qquad j = 1, 2, ..., M \quad (1)$$

where z_j is the output of the jth node in the hidden layer, $X_p = \{x_{1p}, x_{2p}, \ldots, x_{Np}\}$ is the input pattern vector, C_j is the center vector of the Gaussian function for node j, b_j is the inverse of the width associated with the kernel function of node j, through which we can control the receptive field of that neuron, and M is the number of nodes in the hidden layer. The value of z_j is in the range from 0~1, and depends on the distance between X_p and C_j. The closer the distance is, the larger the output of the basis function is. 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:

$$y_k = \sum_{j=1}^{M} W_{kj} z_j \qquad k = 1, 2, ..., L \quad (2)$$

where y_k is the output of the kth node in the output layer, z_j is the output of the jth node in the hidden layer, W_{kj} is the weight between the node j of hidden layer and the node k of output layer, and L is the number of nodes in the output layer.

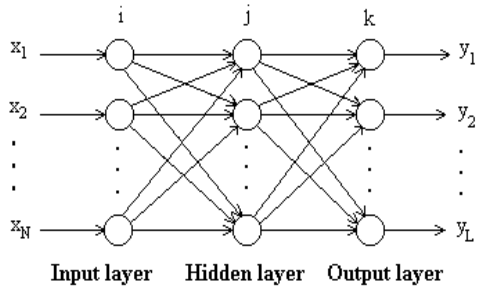

Figure 1: *The Structure of the RBF neural network.*

The outstanding issue associated with the development of a RBF network is the network structure determination and the parameters selection. They can be performed respectively or at the same time. In our research, the network structure determination and the parameters selection are performed respectively. The network structure is fixed in advance, and then the hidden layer parameters and the connection weights 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 critical because it determines the non-linear behaviour of the network. In general, it can be set according to some heuristical considerations. 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 over-fit the training samples and results in poor generalization. In our research, the optimum number of hidden units was determined by experimentation.

The appropriate parameter values of the center $\boldsymbol{C}_j$ and the width b_j of the kernel function in the hidden layer and the connection weights $\boldsymbol{W}_{kj}$ of the output layer 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 $\boldsymbol{X}_p$ 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 below:

(1) Initialization: C_j , b_j and W_{kj} are initially set by some random values in the range [0, 1].

(2) Forward pass: Arbitrarily choose the input feature vector $\boldsymbol{X}_p = [x_{1p}, x_{2p}, \ldots, x_{Np}]^T$ and the desired output $D_p=[d_{1p}, d_{2p},\ldots,d_{kp}]^T$ from the training sample set and feed it to the network, compute the network outputs by proceeding forward through the network, layer by layer.

(3) Backward pass: Calculate the sum-squared error E_p and error gradients versus the parameters $\frac{\partial E_p}{\partial W_{kj}}$, $\frac{\partial E_p}{\partial C_j}$, $\frac{\partial E_p}{\partial b_j}$, layer by layer, starting from the output layer and proceeding backwards:

$$E_p = \frac{1}{2}\sum_{k=1}^{L}\left(y_{kp} - d_{kp}\right)^2 \qquad (3)$$

where y_{kp} is the output of the kth node in output layer, d_{kp} is the desired output of the kth node.

(4) Update parameters through an iterative process:

$$W_{kj}(n+1)=W_{kj}(n)-\eta_1 \frac{\partial E_p}{\partial W_{kj}} \qquad (4)$$

$$C_j(n+1)=C_j(n)-\eta_2 \frac{\partial E_p}{\partial C_j} \qquad (5)$$

$$b_j(n+1)=b_j(n)-\eta_3 \frac{\partial E_p}{\partial b_j} \qquad (6)$$

Where η_1, η_2, and η_3, are rates of learning with respect to parameters W_{kj}, C_j, b_j.

(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 RBF NN can be used for recognising unknown 3D freeform surface from 2D sketching data.

3. Training data acquisition and normalization for 3D freeform surface recognition

In order to train our RBF network, we generate a set of training data between known 3D freeform surfaces and their 2D isometric projections. The 2D and 3D coordinates are directly used as feature data.

Given a 3D shape surface represented by a set of points (N data points on or control points of the surface curves), $S = \{P_i(x, y, z) | i = 1,2,\ldots, N\}$, Its variation can be obtained by either randomly disturbing its original data point positions or rotating it rigidly between –30 to +30 degrees about X, Y, Z axes respectively so that unknown view points in the real world and ambiguity in sketch strokes can be mimicked. Let the original shape has
P-1 variations, in total, there are P similar shapes for training. The resulting 3D training data set T is represented as a one-dimensional vector with the K=3*N*P coordinate component elements.

$$T = \{S_p | p = 1,2,\ldots,P\} = \{t_k | k = 1,2,\cdots,K\}$$

In order to deal with general surfaces, the 3D coordinate data set is firstly normalized in a unit volume as shown in Fig.2. The normalization is performed in two steps:

(1) Search for the minimum element within the training data vector and check if it is negative. If so, the vector will be translated by adding the absolute the value of the minimum element to all elements in the vector.

(2) Search for the maximum element within the vector, and normalize the training vector by dividing all elements by the maximum element (a scaling transformation), so that the training shapes will be within a unite volume.

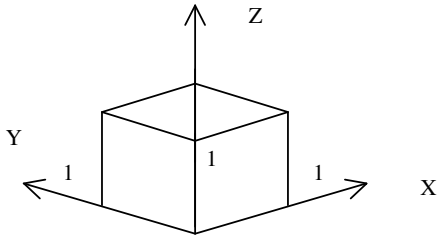

Figure 2: *3D data normalisation*

After normalizing the 3D training data, their 2D isometric projections can be obtained easily. All resultant 3D coordinates and corresponding 2D data sets are then used for training our RBF neural network.

For the learning purpose, 2D sketched input has to be normalized in accordance with the 3D normalization. 2D sketches can be described in a 2D window system X_w-Y_w (Fig. 3). Its origin is at the lower left corner. Similar to screen display resolutions, an actual window may has different heights and widths. However, we can still regard the window as a square without losing information if using the bigger value (H) between its height and the width to represent it. With reference to the isometric projection, from the middle point on the bottom line we draw two lines paralleling to two isometric axes respectively. Similarly, two lines are drawn from the central point on the top line. These four lines combining with the left and the right edges of the window form a projection region, which responds to the biggest area within the window to describe an isometric projection of a cube positioned like the unit volume in Fig 2. The 2D sketch normalization process has four steps as follows.

(1) Scale 2D sketches (four curves in blue color in Fig. 3) to the centre point C (H/2, H/2) of the window to make sure the sketches are within the project region. Draw ray casting lines from the centre to 2D sketched points (small circle on sketched curves) and check if a sketched point is out of the projection region. If so, calculate a scale factor to get the point in. After checking all sketched points, the smallest factor will be applied to all points to achieve our goal.
(2) Scale 2D sketches again to the centre point by a fixed scaling factor: 2*0.82/H (0.82 is the foreshortening factor of an isometric projection).
(3) Translate the origin of the window system to the centre C.
(4) Translate all points along y-axis by –0.82.

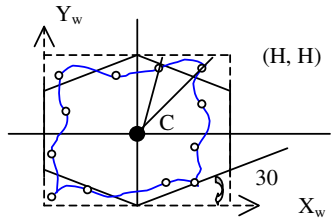

Figure 3: *2D sketch normalisation*

Currently, in our system, every 3D freeform surface consists of 4 three-dimensional curves. For each 3D curve, we specified it by four 3D edit points on the curve and then generate a standard cubic Coons curve with 11 parameter points. All these 44 parameter points will then be used for representing a shape. In order to mimic unknown view points in the real world, we randomly vary the positions 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 these variations will have their corresponding new shapes (each shape has 44 points). After receiving the variations, all points associated with the shapes will be used to normalize the 3D training data.

Each normalized shape with 44 normalized 3D points will be then isometric projected on the 2D projection plan. Note that the projection is automatically consistent with the 2D sketch normalization. The pair of normalized 3D coordinates data (x_i, y_i, z_i | i=1, 2, … , 44) and corresponding 2D coordinate data (sx_i, sy_i | i=1, 2, … ,44) then be used for training data.

By the method mentioned above, we have obtained the required training data set of a 3D freeform surface and their corresponding 2D projection data set. The dimension of a 2D training vector is 88 (44x2) and the dimension of a 3D training vector is 132 (44x3). In total we obtained 370 shape variations. Among them, we randomly picked up 300 shapes as a training sample set and left 75 for network testing. For training the recogniser, the 2D training vector of a shape is inputted into the input layer of the neural network, while the corresponding 3D training vector is uploaded in the output layer for supervised network learning. After the training, we first used a group of 2D training vectors from the 75 left shapes as inputs to the recogniser and the corresponding 3D outputs are regarded as learning outcomes. By comparing the 3D outputs to their corresponding 3D original shape data, the recogniser can be evaluated if it is ready for recognising shapes from sketches. If the recogniser is well trained, it can take sketches as input and output 3D desired shapes. If it is not good enough, the further training may be needed.

4. Experimental results and performance tests

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 L=132 correspondingly. The number of hidden neuron is 100, determined by experiment.

After the structure of network has been determined, the training sample set is utilized to train the network according to the Back-propagation learning algorithm mentioned in section 2.

The Fig 4 gives the changes of the sum-squared error and leaning rate. After 2741 epochs, the neural network converged to a predefined threshold (T). From the learning rate graph, it can be seen that the learning rate decreased sharply and with stability towards a stable state. Then the recognition performance of the RBF network was tested with the test sample set. To do so, two groups of experiments were made.

We trained the RBF network by two types of surfaces: cross-sectional surfaces and four-sided surfaces. Each surface has four curves but their topology is different. This means we could use any four curves to represent any type of surface. Here we only show two types of surface in terms of open topology surfaces (cross section curves) and full-connected surfaces (four sided patches). On each curve, 11 parametric points with equal parametric intervals are used to represent a smooth curve. Some experimental results are given in Figures 5 and 6.

Fig.5 shows the recognition result of a four-sided surface. The Fig.5 (a) shows the original 3D surface. This surface was isometric-projected on the XY plane. The corresponding 2D data was then fed to the neural network. The output of the neural network is shown in the Fig. 5(b) as a reconstructed 3D freeform surface.

Fig.6 illustrates the recognition result of a desired cross-sectional curved surface (Fig. 6a). This surface was represented by four cross sectional curves (Fig 6b). Those curves were then projected and fed into the neural network and reconstructed to a surface with rendering (Fig 6c).

Fig.7 shows a design example. The Original freehand 2D sketch (Fig. 7a) has been extracted with 44 points on its four sides (Fig 7b). Those 2D points were normalized before being applied to the neural network. The corresponding data resulted in a 3D freeform surface (Fig 7c) from the neural network. The result is very satisfactory. From our experiments, the average error between the desired 3D surface and recognized 3D surface over all our test sample data (75) is less than 0.1 (in normalized space). From the results above, we observe that the RBF network performs very well in 3D freeform surface recognition.

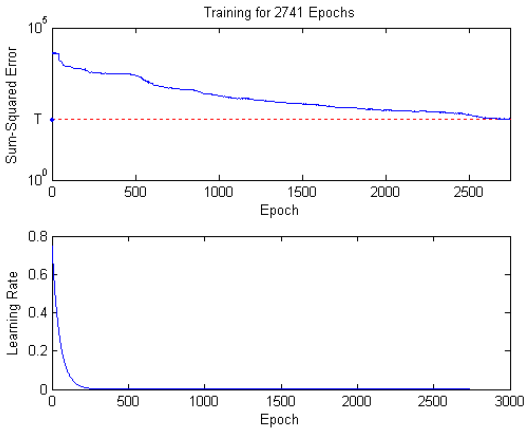

Figure 4: *The changes of the sum-squared error and learning rate.*

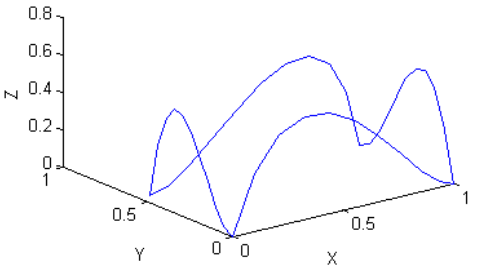

Figure 5(a) *An original four-sided 3D surface in world coordinate system*

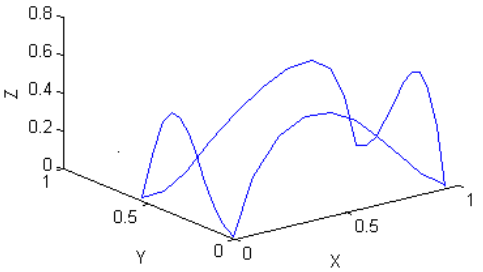

Figure 5(b): *The 3D freeform surface reconstructed by NN*

Figure 5: *Recognition result of a four-sided surface*

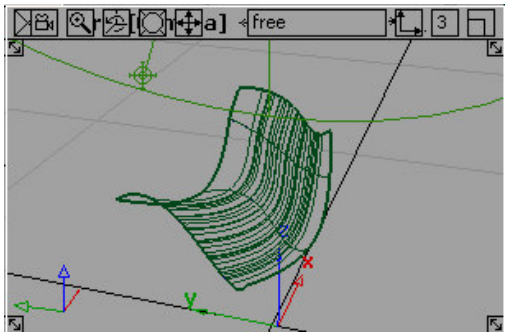

Figure 6(a): *The desired surface.*

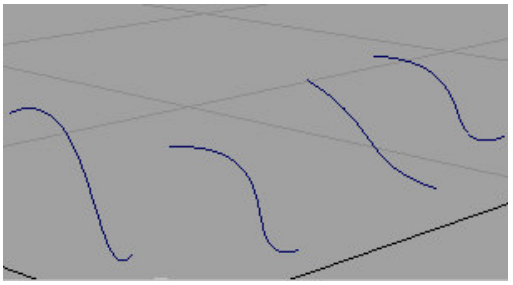

Figure 6(b): *Cross sections for recognition.*

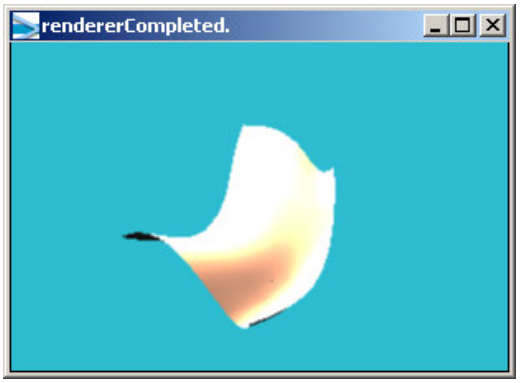

Figure 6(c): *Recognized surface with rendering.*

Figure 6: *Recognition of a cross-sectional surface*

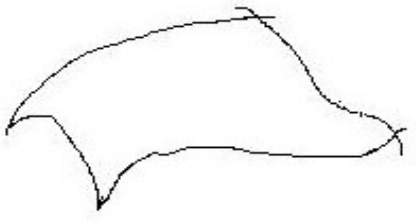

Figure 7 (a): *The original 2D freehand sketch.*

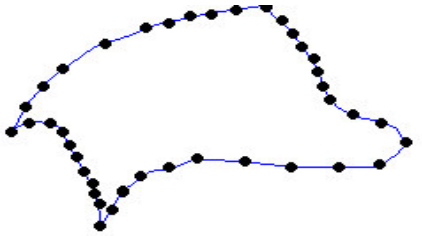

Figure 7 (b): *Extracted with 44 points on its four sides*

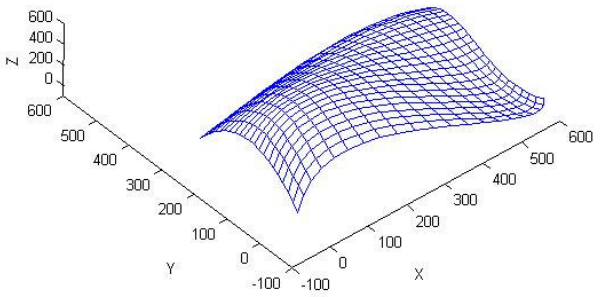

Figure 7 (c): *The resultant surface.*

Figure 7. *A design example*

5.Discussion and conclusion

This paper presents a RBF neural network-based surface recognition method from 2D sketches (or 2D drawings). The results have shown that ANN techniques can be used for shape learning from 2D sketches.

For creating training data, currently, we only employed known 3D shapes and their corresponding 2D projections. The reason for not directly using sketches for training is because that based on 2D sketches, we don't know their corresponding 3D information. If we have a design database holding freehand sketches and their corresponding 3D information, it is possible to use the real sketches for training.

Current training data is based on 44 3D points on 4 surface curves. Points on each curve have an equal parametric space. The reason for selecting 4 curves is that 4-sides surfaces are easy to compute. Dividing each curve by 10 in its parametric space is easy again to give 11 points. Actually, these 44 points can be on many numbers of curves such as 1, 2, 3, or 5. In the neural network, each coordinate component is separately represented by a neuron. Thus, it is believed that impact of their data structure on the network's performance is not significant. In theory, these points can be distributed randomly on the surface. Our examples have shown that four curves can have arbitrary topology such as full connected (4 side surface) or totally open (4 cross-sections). Furthermore, the training data can be control points for a surface as well.

On the other hand, from the trained neural network, we can only receive 44 normalised 3D points. In order to reconstruct 3D surfaces, we need to know some constraints such as first 11 points on one curve and the following 11 points on another. This is the other reason for using curves as constraints. Otherwise, we will receive a group of 3D data and have to use general surface approximation method to get a surface. This will have difficulties in determining boundaries of the surface.

For the training data, we assume that designers prefer to describe 3D shapes in a modelling system and then create their projections separately. For example, if a face is parallel to the YZ plane perceptually, points on the face will be assumed to have the same x coordinate component. Therefore, in our system, a 2D point (isometric projection) has different x and y coordinate components from its corresponding 3D points. This is different from the 2 ½ inflation system as [VSMM00, LS96]. We cannot simply learn z component only for each 2D training point.

Currently, the number of input layer nodes is related to the number of curves. This brings the limitation on freehand sketches. In order to support any types of sketched 2D input, we are going to use scan-line techniques to divide a 2D sketch after normalisation by a group of grid lines. Each grid point may require a node on the input layer. If we can map a corresponding 3D point for each 2D grid point, then the network can be used for general surface training. As a result, the final learnt surface could be represented by a set of 3D points, which can be constructed by reverse engineering methods.

In order to make learning more efficiently (in terms of both computation and number of examples), as pointed in [RYA02], using complex intermediate representations extracted from raw data rather than raw data itself will benefit future work on recognition and in particular, robust recognition under realistic condition. For example, Lipson and Shpitalni [LS02] have used 3D-2D geometric correlation such as angles to recognise 3D objects from 2D drawings. If we can learn from intermediate attributes rather than raw data, the learning processing may become more efficient and robust.

Acknowledgement

This work was funded by the UK EPSRC grant (GR/S01701/01).

References

[BF02] BARHAK J., FISCHER A.: Adaptive reconstruction of freeform objects with 3D SOM neural network grids. *Computers & Graphics* 26 (2002), 745-752.

[Clo71] CLOWES M.B.: On seeing things. *Artificial Intelligence* 2(1) (1971), 79-112.

[DSC98] 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.

[Ede93] EDELMAN S.: On learning to recognize 3D objects from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (8) (1993), 833-837.

[GD00] GROSS M.D., DO E.Y.L.: Drawing on the back of an envelope: A framework for interacting with application programs by freehand drawing, *Computers & Graphics*. 24 (2000), 835-849.

[GY95] GU P., YAN X.: Neural network approach to the reconstruction of freeform surfaces for reverse engineering. *Computer Aided Design* 27 (1) (1995), 59-64.

[Huf71] HUFFMAN D.A.: Impossible objects as nonsense sentences. In: Meltzer, Michie D., editors. *Machine Intelligence*, Edinburgh University Press. (1971), 295-323.

[LLTL91] LIN W.C., LIAO F.Y., TSAO C.K. AND LINGUTLA T.: A Hierarchical Multiple-view Approach to Three-Dimensional Object Recognition. *IEEE Trans. Neural Networks* 2 (1) (1991), 84-92.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3D freeform design. *SIGGRAPH Computer Graphics proceedings* (1999), 409-416.

[LS96] 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.

[LS02] LIPSON H., SHPITALNI M.: Corelation-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.

[Mac73] MACKWORTH A.K.: Interpreting Pictures of Polyhedral Scenes. *Artificial Intelligence* Vol. 4 (1973), pp. 121-137.

[PG90] POGGIO T., GIROSO F.: Regularization algorithms for learning that are equivalent to multi-layer networks. *Sci.* 247 (1990), 978-982.

[QWJ01] QIN S.F.,WRIGHT D.K., JORDANOV I.N.: A conceptual design tool: A sketch and fuzzy logic based system. *Proceedings of The Institution of Mechanical Engineers Part B-Journal of Engineering Manufacturing.* 215 (2001), 111-116.

[RYA02] ROTH D., YANG M.S. , AHUJA N.: Learning to Recognise Three Dimensional Objects. *Neural Computation* 14 (2002) 1071-1103.

[VSMM00] 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.

[VTMS04] VARLEY P.A.C.,TAKAHASHI Y., MITANI J., AND SUZUKI H., A two-stage approach for interpreting line drawings of curved objects, EUROGRAPHICS Workshop on Sketch-based Interface and Modeling, 2004.

[ZHH96] ZELEZNIK R.C., HERNDON K.P., and Hughes J.F.: SKETCH: an interface for sketching 3D scenes. *SIGGRAPH Computer Graphics Proceedings* (1996), 163-170.