3D Freeform Surfaces from Planar Sketches Using Neural Networks

Usman Khan, Abdelaziz Terchi, Sungwoo Lim, David Wright, and Sheng-Feng Qin

Brunel University, Uxbridge, Middlesex, United Kingdom
{Usman.Khan, Aziz.Terchi, Sungwoo.Lim, David.Wright, Sheng.Feng.Qin}@brunel.ac.uk

Abstract. A novel intelligent approach into 3D freeform surface reconstruction from planar sketches is proposed. A multilayer perceptron (MLP) neural network is employed to induce 3D freeform surfaces from planar freehand curves. Planar curves were used to represent the boundaries of a freeform surface patch. The curves were varied iteratively and sampled to produce training data to train and test the neural network. The obtained results demonstrate that the network successfully learned the inverse-projection map and correctly inferred the respective surfaces from fresh curves.

Keywords: neural networks, freeform surfaces, sketch-based interfaces.

1 Introduction

The preliminary stages of the conceptual product design process are characterised by a high degree of creative activity. Designers strive to convert new ideas into graphical form as soon as possible. It can be argued that sketching is an essential activity for creative design. The reasons are manifold. It permits the rapid exploration and evaluation of concepts [1]. It also assists the designer's short-term memory and facilitates communication with other people. When designers sketch shapes on a sheet of paper, they start with a vague concept, which they progressively refine into a final product. While numerous iterations are usually undertaken, the salient properties of the original idea are often maintained. Recently, the desire to automate the early phase of the conceptual product design have given impetus to the development intelligent tools to simulated the way of sketching is performed by designers [2-4]. However, most existing approaches are restricted to fairly simple objects such as planar and polygonal shapes. Consideration of complex free-form surfaces is a challenging process. The problem has surprisingly received little attention in the literature.

The problem of reconstructing a three dimensional (3D) shape from a planar drawing is fundamental problem in computer vision and computer aided geometric design. Clowes [5], developed a classification method based on labelling drawings and sorting their edges to recognise polyhedral shapes. Though, their method was extended to other line drawings [6-8], their work mainly involved determination of the depth from a 2D drawing consisting of flat surfaces with straight line edges. With regard to freeform surfaces some of the foundation work was developed by Igarashi *et al.* [3] who reproduced rough freeform models from freehand sketch input. Since then only moderate progress has been achieved in recovering freeform surfaces from on-line sketches. Michalik *et al.* [9] proposed a constraint-based system that reconstructed a B-spline surface from a sketch into 3D. These papers employ techniques based on rules or constraints to extract the correlation between the 2D drawings and their respective 3D shapes. In the same vein, the work of Lipson and Shpitalni [6] is also based on the notion of correlation.

Work in recognition of shape features from 2D input was reported by Nezis and Volniakos [10]. The topology of the input drawing was exploited to categorize the shape features. Peng and Shamsuddin [11] claimed that a neural network was able to estimate the pose of a 3D object from a 2D image from any arbitrary viewpoint. Reconstruction of 3D shapes by estimating their depth was done by Yuan and Niemann [12]. They represented objects using a triangular mesh from reverse engineered data and demonstrated that a neural network could reconstruct 3D geometry from 2D input.

Early work pertaining to reconstruction of freeform surfaces was covered by Gu and Yan [13]. A non-uniform b-spline (NURB) surface was fitted over scattered data from a reverse engineering source using an unsupervised neural network. Hoffman and Varady [14] and Barhak and Fischer [15] extended this line of research. However, their methods required that all three dimensions be available for reconstruction purposes.

The present paper proposes and develops a methodology for 3D freeform surface inference from freehand planar sketches. The methodology is based on neural networks. Specifically, an MLP neural network, trained with a momentum-augmented backpropagation learning algorithm, is employed to induce 3D freeform surfaces from 2D sketches. The reconstruction procedure consists of two steps: first a neural network is trained on pairs of normalised 3D surfaces and their corresponding projection curves, then the trained neural network is used to reconstruct unknown 2D sketches. The methodology is tested with a range of data and produced satisfactory results.

The remainder of this paper is organised as follows. In section 2 3D freeform surface reconstruction is formulated as an inverse problem. In section 3, neural networks together with their learning algorithms are discussed. The data generation procedure is discussed in section 4. The computational results are presented in section 5. Finally section 6 treats conclusions and future work.

2 Problem Formulation

Volumetric concepts originate in the mind of a designer as 3D entities. They are then transformed, via an isometric projection onto an arbitrary view plane, into planar sketches. Such a task is considered as the direct problem. The 3D freeform surface inference problem consists of extracting the 3D geometry from the 3D, i.e., to recover the depth information that was lost during the projection process. This process can be regarded as the inverse process of the original projection. The direct problem is, in general, a well-posed problem and can be solved analytically using concepts from projective geometry.

In contrast, the inverse problem is, in general, ill-posed. The solution may not be unique, may lack continuity could be highly influenced by the amount of noise present in the data. Therefore, 3D surface reconstruction is indeterminate in that an infinite number of possible 3D surfaces can correspond to the same 2D curve. To obtain a unique and physically meaningful solution requires additional information in terms of general assumptions, constraints and clues from experience. In the context of this paper, the planar curves are constrained to lie in the x-z or the y-z planes and their control points are restricted to vary only along the z-direction. Such constraint ensures the maintenance of the planar property of the inferred 3D surfaces and leads to a single one to one mapping from the input 2D curves to the expected 3D surfaces. This renders the inverse problem tractable.

Given a set of p ordered pairs $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1,\ldots, p\}$ with $\mathbf{x}_i \in R^2$ and $\mathbf{y}_i \in R^3$, the surface reconstruction problem is to find a mapping $F : R^2 \rightarrow R^3$ such that $F(\mathbf{x}_i) = \mathbf{y}_i$, $i = 1,\ldots, p$. In practice, the function F is unknown and must be determined from the given data $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1,\ldots, p\}$. A neural network solution of this problem is a two-step process: training, where the neural network learns the function from the training data $\{\mathbf{x}_i, \mathbf{y}_i\}$, and generalisation, where the neural network predicts the output for a test input. We demonstrate how an MLP neural network trained with a momentum-augmented backpropagation algorithm on a collection of 2D-3D dependencies, can approximate the inverse map in a computationally efficient form.

3 Neural Networks

Neural networks are connectionist computational models motivated by the need to understand how the human brain might function. A neural network consists of a large number of simple processing elements called neurons. Feedforward neural networks have established universal approximation capability [16] and have proven to be potent tool in the solution of approximation, regression, classification and inverse problems.

For this reason, a MLP neural network is selected for the solution of the reconstruction problem. The MLP neural network is composed of three layers: the input layer, the hidden layer and the output layer. The neurons of the input layer feed data to the hidden layer where it performs the following nonlinear transformation:

$$s_j = f\left(\sum_k w_{jk} x_k\right) . \tag{1}$$

where $\mathbf{x}_k$ are the neurons inputs signals, $\mathbf{s}_j$ are neural outputs and $\mathbf{w}_{jk}$ the synapses and f is an activation function. For MLP neural network, the sigmoid function is used as the activation function. The output layer of the neurons takes the linear transformation:

$$y_j = f\left(\sum_k w_{jk} s_k\right) . \tag{2}$$

where $\mathbf{y}_\mathrm{j}$ are the output layer neuron outputs, and $\mathbf{w}_\mathrm{ij}$ are synapses. Neural network training can be formulated as a nonlinear unconstrained optimisation problem. So the training process can be realised by minimising the error function E defined by:

$$E = \frac{1}{2}\sum_{k=1}^{p}\sum_{j=1}^{n}\left(y_{jk} - t_{jk}\right)^2 . \tag{3}$$

where $\mathbf{y}_{jk}$ is the actual output value at the j-th neuron of output layer for the k-th pattern and $\mathbf{t}_{jk}$ is the target output value. The training process can be thought of as a search for the optimal set of synaptic weights in a manner that the errors of the output is minimised.

3.1 Backpropagation Algorithm

Most learning algorithms are based on the gradient descent strategy. The backpropagation algorithm (BP) [17] is no exception. The BP algorithm uses the steepest descent search direction with a fixed step size α to minimise the error function. The iterative form of this algorithm can be expressed as:

$$w_{k+1} = w_k - \alpha g_k . \tag{4}$$

where **w** denotes the vector of synaptic weights and $g = \nabla$ E(**w**) is the gradient of the error function E with respect to the weight vector **w**.

In the BP learning algorithm the weight changes are proportional to the gradient of the error. The larger the learning rate, the larger weight changes on each iteration, and the quicker the network learns. However, the size of the learning rate can also influence the network's ability to achieve a stable solution. In a neighbourhood of the error surface where the gradient retains the same sign, a larger value of the learning rate α results in a rapid reduction of the energy function faster. On the other hand, in an area where the gradient rapidly changes sign, a smaller value of α maintains the descent direct along the error surface.

Despite its computational simplicity and popularity, the BP training algorithm is plagued by such problems as slow convergence, oscillation, divergence and "zigzagging" effect. The BP learning algorithm is in essence a gradient descent optimisation strategy of a multidimensional error surface in the weight space. Such strategy exhibits has inherently slow convergence; especially on large-scale problems. This trait becomes more pronounced when the condition number of the Hessian matrix is large. The condition number is the ratio of the largest to the smallest eigenvalue of the network's Hessian matrix. The Hessian matrix is the matrix of second order derivatives of the error function with respect to the weights.

In many cases the error hypersurface is no longer isotropic but rather exhibits substantially different curvatures along different directions, leading to the formation of long narrow valleys. For most points on the surface, the gradient does not point towards the minimum, and successive steps along the gradient descent oscillates from one side to the other. Progress towards the minimum becomes very slow. This suggests a method that dynamically adapts the value of the learning rate, α to the topography of the error surface.

3.2 Momentum-Augmented Backpropagation

One way to circumvent the above problem, the BP propagation in eq. 4 is augmented with *a momentum term*:

$$w_{k+1} = w_k - \alpha g_k + \beta(w_k - w_{k-1}) \ . \tag{5}$$

The momentum term, β has the following effects: 1) it smoothes the oscillations across narrow valleys; 2) it amplifies the learning rate when all the weights change in the same direction; and 3) enables the algorithm to escape from shallow local minima.

In essence, the momentum strategy implements a variable learning rate implicitly. It introduces a kind of 'inertia' in the dynamics of the weight vector. Once the weight vector starts moving in a particular direction in the weight space, it tends to continue moving along the same direction.

If the weight vector acquires sufficient momentum, it bypasses local minima and continues moving downhill. This increases the speed along narrow valleys, and prevents oscillations across them. This effect can also be regarded as a smoothing of the gradient and becomes more pronounced as the momentum term approaches unity. However, a conservative choice of the momentum term should be adopted because of the adverse effect that might emerge: in a narrow valley bend the weight movement might jump over the walls of the valley, if too much momentum has been acquired.

The learning algorithm requires the *a priori* selection of the learning rate and the momentum coefficient. However, it may not easy to choose judicious values for these parameters because a theoretical basis does not seem to exists for the selection of optimal values. One possible strategy is to experiment with different values of these parameters to determine their influence on the overall performance. The moment augmented backpropagation algorithm may be used both in batch and on-line training modes. In this paper the batch version is used.

4 Data Generation

The neural network used in this paper is trained in a supervised mode via a collection of input-output pairs to optimise the network parameters (i.e. synaptic weights and biases). Training is accomplished through a learning algorithm that iteratively adjusts the network parameters until the mean squared error (MSE) between the predicted and the desired outputs reaches a suitable minimum.

A training set was generated from a family of freeform surfaces whose edges also referred to as the boundaries, consisted of four orthogonally arranged planar curves. An example of a planar curve is shown in Fig. 1. Each curve was governed by four independent control points and represented by a Non Uniform Rational B-Spline (NURBS). Two control points determined the ends of the curve whereas the remaining ones controlled its general shape. NURBS control points need not intersect the curve and can lie anywhere in the 3D space. The curve was uniformly sampled and the coordinates of the sample points formed the input features for the neural network.

The planar curves were placed in the x-z plane or the y-z plane and their control points were only altered along the z-direction to maintain their planar property. Each of the four boundary curves were uniformly sampled at 10 positions. Hence a surface, whether represented in 2D or 3D, consisted of 40 sample points. A point on the 3D surface is represented by the x, y and z coordinates whereas in 2D, it is represented by its x and y coordinates. Therefore a 3D surface is represented by 120 independent features and its respective 2D curve by 80 features.

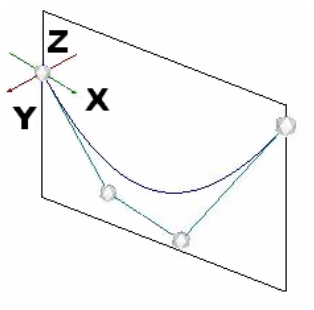


Fig. 1. Planar 3D NURBS curve. Each control point of the curve lies on the same plane as the others.

The positions of the control points were varied to produce a class of unique freeform surfaces. Each surface was projected onto the view plane to produce the respective 2D planar projection. The training set is composed of pattern pairs, each containing a 3D surface and its corresponding 2D curve.

The data set was normalised so that the input 3D pattern would fit within a unit cube and its respective 2D pattern within the unit square. Normalisation ensures that the values lie within the characteristic bounds of the activation functions.

Fig. 2 shows two examples of normalised pattern pairs that were used to train the neural network. The 2D input patterns are depicted in Fig. 2 (a) whereas their

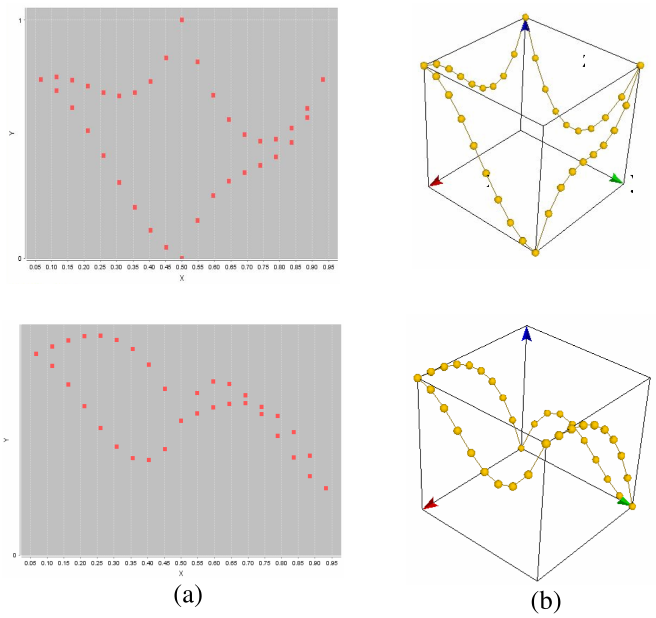


Fig. 2. Examples of 2D input patterns and corresponding 3D output patterns

corresponding 3D output patterns are shown in Fig. 2 (b). It can be seen that the boundary of the surfaces are described by a series of sample points and fits within a unit square for 2D and unit cube for 3D. Notice that the viewpoint of the 3D desired pattern coincides with the viewpoint of the 2D input pattern.

The entire data set was composed of 4096 patterns. The whole set cannot be used to train the network because no data would be left to test the network's ability to generalise into fresh inputs. Therefore the data set was randomly split, using three subsets that were used for training, validation and testing. Accordingly, the number of training, validation and testing patterns pairs were therefore 2867, 819 and 410 respectively. This corresponds to a 70, 20 and 10 percent split of the data.

5 Computational Results

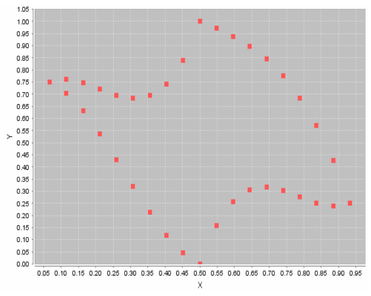
A three-layer MLP network was employed in our research. The input and output layer dimensions of the neural network were determined from the features of the training set. The input layer consist of 80 nodes and while the output layer consists of 120 nodes. The number of nodes in the hidden layer is freely adjustable and results in different network performance depending on the number of hidden nodes used. The parameters used in the network are shown in Table 1.

Table 1. Network Architecture and Parameters

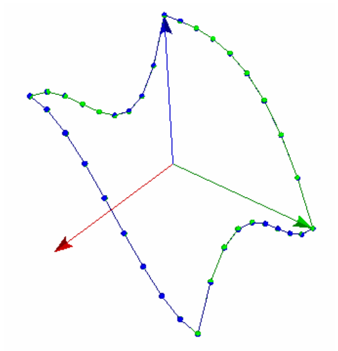
Number of Input Nodes	80
Number of Output Nodes	120
Learning Rate (α)	0.7
Momentum (β)	0.6
Number of Epochs	5000
Number of Training Patterns	2867
Learning Mode	Batch

The number of hidden nodes indicates the network complexity and governs how accurately it learns the mapping from the input patterns to the outputs. It also affects how long the network takes to perform each training cycle. The higher the number of hidden nodes, the more computation is required and hence a longer training time is needed. Experimentation with different numbers of nodes in the hidden layer was conducted. Multiple neural networks were trained with similar parameters such as the learning rate, momentum and training sets. In this case the learning rate was 0.7 and the momentum was 0.6. Only the number of hidden nodes was changed. It was found that a neural network of 50 hidden nodes produced the best reconstruction error over a fixed number of epochs. This was found by comparing the average reconstruction error of the networks based on a fresh test set containing 410 patterns.

Finally a new network of 50 hidden units was trained again for 5000 epochs. The final training error was 0.06. At the end of the training, the net was saved the test set applied to the network. The obtained results show that the neural network was able to infer the 3D shape of a freeform surface from its respective 2D input pattern.



(a)



(b)

Fig. 3. Test Input Patterns with Predicted and Desired Outputs

An example test pattern that was applied to the trained network is shown in Fig. 3 (a). The predicted and the expected 3D patterns that correspond to the 2D surface are shown in Fig. 3 (b). The predicted pattern is depicted in green whereas the desired pattern is in blue. It can be noticed from the plots in Fig. 3 (b) that the two surfaces per image are almost identical and hence that the neural network has inferred the correct shape that was desired. However, small deviations in the predicted patterns can be seen when observed closely. They relate to the network's ability to predict the desired surfaces. The distributions of errors are presented in Fig. 4. This shows the Euclidean distances between each point from the predicted surface and its corresponding point on the desired surface. The RMS error for this pattern was 0.33%.

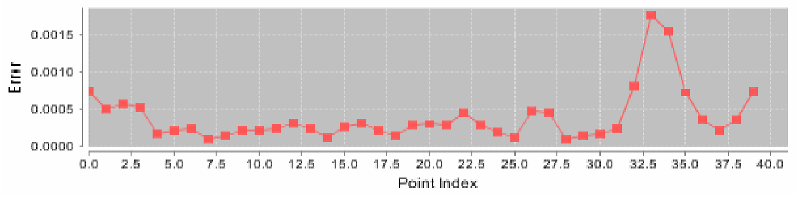


Fig. 4. Distribution of Squared Errors Between Predicted Output and Expected Output

6 Conclusions and Future Work

In this paper a methodology for the inference of 3D freeform surfaces from 2D surface representations using neural networks has been proposed. A representative dataset was generated by iteratively adjusting the control points of freeform surface boundary curves that were previously uniformly sampled. The dataset was normalised and randomly split into three subsets: training, validation and test sets. An MLP was optimised using different numbers of hidden nodes. The best network, i.e. the network with the lowest training RMSE, was trained with a representative family of 2D and 3D pattern pairs. The neural network was applied to a set of 2D patterns had not been encountered before. Obtained 3D results demonstrate that the target freeform surfaces can be reproduced from 2D input patterns within 2 % accuracy. Future work will extend the methodology to more complex shapes and reconstruct the 3D surface that corresponds to the inferred surface boundary.

Acknowledgements

This research is supported by the EPSRC, research grant [EPSRC GR/S01701/01]. The methodology was implemented in Java 1.5 using Netbeans 5.0 based on a Windows 2000/XP platform. The neural network was developed using the Java Object Orientated Neural Engine and the 3D graphics drawn using Jun for Java. The authors would also like to thank Yosh Nishinaka from Software Research Associates Inc. and Paolo Marrone from JOONE for their assistance.

References

1. Lim, S., Lee, B., Duffy, A.: Incremental modelling of ambiguous geometric ideas (I-MAGI): representation and maintenance of vague geometry. Artificial Intelligence in Engineering **15** (2001) 93-108
2. Karpenko, O., Hughes, J., Raskar, R.: Free-form Sketching with variational implicit surfaces. Eurographics **21** (2002) 585-594
3. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. 26th International Conference on Computer Graphics and Interactive Techniques (1999) 409-416
4. Alexe, A., Gaildrat, V., Barth, L.: Interactive Modelling from Sketches using Spherical Implicit Functions. Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa. Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa., Africa (2004) 25-34
5. Clowes, M.: On Seeing Things. Artificial Intelligence **2** (1971) 79-116
6. Lipson, H., Shpitalni, M.: Correlation-Based Reconstruction of a 3D Object from a Single Freehand Sketch. AAAI Spring Symposium on Sketch Understanding, Palo, Alto, USA (2002) 99-104
7. Varley, P., Martin, R.: Estimating Depth from Line Drawings. Symposium on Solid Modelling. ACM press, Saarbrucken, Germany (2002) 180-191
8. Shpitalni, M., Lipson, H.: 3D conceptual design of sheet metal products by sketching. Journal of Materials Processing Technology **103** (2000) 128-134
9. Michalik, P., Kim, D.H., Bruderlin, B.D.: Sketch- and constraint-based design of B-spline surfaces. Proceedings of the seventh ACM symposium on Solid modelling and applications. ACM Press, Saarbrücken, Germany (2002) 297-304
10. Nezis, K., Vosniakos, G.: Recognizing 2 1/2D shape features using a neural network and heuristics. Computer-Aided Design **29** (1997) 523-539
11. Peng, L.W., Shamsuddin, S.M.: 3D Object Reconstruction and Representation Using Neural Networks. Computer graphics and interactive techniques in Australia and South East Asia, Singapore (2004) 139-147
12. Yuan, C., Niemann, H.: Neural Networks for appearance-based 3-D object recognition. Neurocomputing **51** (2003) 249-264

13. Gu, P., Yan, X.: Neural network approach to the reconstruction of freeform surfaces for reverse engineering. Computer Aided Design **27** (1995)
14. M. Hoffman, Varady, L.: Free-form Surfaces for Scattered Data by Neural Networks. Journal for Geometry and Graphics **2** (1998) 1-6
15. Barhak, J., Fischer, A.: Adaptive reconstruction of freeform objects with 3D SOM neural network grids. Computer and Graphics **26** (2002) 745-751
16. Hornick, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2** (1989) 359-366
17. Rumelhart, D.E., McClelland, J.L.: Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1. MIT Press, Massachusetts (1986)