3D Object Reconstruction and Representation Using Neural Networks

Lim Wen Peng[1] and Siti Mariyam Shamsuddin[2]
Department of Graphics and Multimedia,
Faculty of Computer Science and Information System, University Technology of Malaysia
wen_peng@yahoo.com[1], mariyam@fsksm.utm.my[2]

ABSTRACT – 3D object reconstruction is frequent used in various fields such as product design, engineering, medical and artistic applications. Numerous reconstruction techniques and software were introduced and developed. However, the purpose of this paper is to fully integrate an adaptive artificial neural network (ANN) based method in reconstructing and representing 3D objects. This study explores the ability of neural networks in learning through experience when reconstructing an object by estimating it's z-coordinate. Neural networks' capability in representing most classes of 3D objects used in computer graphics is also proven. Simple affined transformation is applied on different objects using this approach and compared with the real objects. The results show that neural network is a promising approach for reconstruction and representation of 3D objects.

KEYWORDS – Reconstruction, representation, back propagation, multilayer feed-forward neural networks, affined transformation, third order polynomial, object space.

1 INTRODUCTION

3D reconstruction has turned out to be an essential need in areas as diverse as medical imaging and artistic applications, product design, reverse engineering and rapid prototyping, among others. The manual creation of 3D models is time consuming and therefore cost expensive. For that reason, techniques are under investigation, which allow the automatic reconstruction of 3D objects. Those techniques can be subdivided into two kinds of methods, active and passive [Wolfgang Niem and Jochen Wingbermuhle 1997]. The disadvantage of the active methods (e.g. structured light, laser scanner, laser range maps, and medical magnetic resonance) is that the reconstruction process can turn out to be a high budget project. Therefore, the presented approach belongs to the passive methods which requires less equipment and can be applied more generally.

Traditional 3D reconstruction methods do not perform well in two senses: 1) they cannot handle the highly complex cases found in nature (e.g. human organs or microscopic images of tissue), and 2) they do not put the surface data in a form which is compact and fit for simulation, visualization or navigation.

In this paper, the multilayer feed-forward neural network reconstruction technique is an initial study on the ability of neural networks in reconstruction and thus only objects with moderate complexity are tested. Besides, 3D object representation using neural network is exact, accurate and compact at the same time. The above statement is proven by experiments and examples in this study.

Generally, the whole reconstruction procedure consists of the following steps: image data acquisition whether is 2D or 3D → registration and integration algorithm → surface fitting through surface representation method → display [Vit Zyka and Radim Sara 1997; Inge Soderkvist 1999]. However, in this study, we modify the flow as shown in Figure 1.

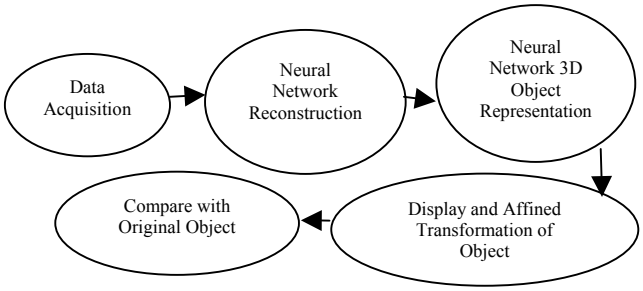


Figure 1: A Proposed 3D Object Reconstruction

Data is taken from 3D object file and not from the image. The process of training and learning using back propagation neural network is used to estimate the coordinate z object (depth object) in the reconstruction process. On the other hand, 3D coordinates of simple objects have become the inputs for the function in multilayer feed-forward neural network, and produces output that indicates either a point in an object space belongs to the object or not after training, learning and output generation [Emmanouil Piperakis and Itsuo Kumazawa 2001]. After representation process, 3D object is displayed using affined transformations. Finally, the object is compared with the original object from 3D file to prove its accuracy and efficiency using neural network.

2 OBJECTIVES

This paper is to prove that multilayer feed-forward neural networks is capable of reconstruction and representing most classes of 3D objects used in computer graphics. Back-propagation method is used for training multilayer feed-forward neural networks. Objects generated by neural network in both reconstruction and representation process are displayed in a window created using OpenGL. If the result shows that the similarity of generated object and original object is very high, then it is proved that 3D object reconstruction and representation can depend completely on neural network, and at the same time

increase the accuracy of object generated with less time consuming.

3 VARIOUS TYPES OF 3D OBJECT RECONSTRUCTION

There are five different cases of 3D reconstruction [Anders Heyden 1995]. The first case is where the images are taken with uncalibrated camera, making it possible to reconstruct the object up to projective transformations. Second, reconstruction from calibrated cameras, making it possible to reconstruct the object up to similarity transformations, is presented. Third, the algebraic properties of the multilinear functions and the ideals generated by them are investigated. The forth case uses Euclidean reconstruction technique when some information of the cameras are given. The final case is reconstruction of one image of an object or line drawing, which is known to be piecewise planar.

There are quite a number of reports and papers that described the methods for 3D reconstruction from multiple view images. However, all these approaches are not suitable for line drawing analysis and single view image. Literature of computer vision shows techniques for extraction of spatial information from images such as shading, lighting, shadowing, perspective, stereo and others. However, in this paper, none of the above is discussed because inputs data are taken from 3D Studio Max file.

This paper proposed a reconstruction technique using multilayer feed-forward neural network. Below are the advantages of 3D reconstruction using neural network:

(a) Neural networks with back-propagation technique are able to estimate the depth (z) of an object with higher accuracy than other methods. It also means that neural networks are able to reconstruct object from 2D image to 3D after training.

(b) This type of reconstruction is able to produce more points of an object or surface. Therefore neural network is able to reconstruct more complex object with smoother surface.

(c) Even with scattered or unorganized data of an object is provided, neural networks are able to regenerate the object when outliers are removed and the smoothness of the surface is maintained.

4 DISCUSSIONS ON 3D OBJECT REPRESENTATION

The representation of 3D objects is a well research area of Computer Graphics. The ability to manipulate the shape of an existing object depends greatly on the representation. The following are some of the most common and important methods of representation [Emmanouil Piperakis and Itsuo Kumazawa 2001].

First, representation of polygonal objects is approximated by a net of mesh or planar polygonal facets. Second, bi-cubic parametric patches are 'curved quadrilaterals'. This representation is similar to the polygon mesh, except for the fact that the individual polygons have been replaced by curved surfaces. Constructive solid geometry (CSG) is another method of exact representation to within certain rigid shape limits. CSG method is a volumetric representation; elementary volumes or primitives represent the shapes. Spatial subdivision techniques mean dividing the object space into elementary cubes, known as voxels, and labeling each voxel as empty or as containing part of an object. Another two types of representation are implicit and explicit representation. An implicit function is, for example: $x^2 + y^2 + z^2 = r^2$ which is the definition of a sphere. Explicit representation is the graph of a function $z = f(x, y)$.

The main comparisons and distinguishing factors between the above existing methods are: a) accuracy vs. data size, b) continuity vs. discrete representation and c) surface vs. volumetric [Emmanouil Piperakis and Itsuo Kumazawa 2001].

By using voxels and polygon meshes, the number of representational elements per object is likely to be high if accuracy is to be achieved. However, the complexity of the representation is low. On the contrary, in the case of bi-cubic patches, the complexity of representation is higher. But, the number of elements is seemed to be much lower in most contexts. As far as rendering is concerned, it is therefore more efficient to represent a shape with many simple elements, like polygons or voxels. This is better than represents it with far fewer and more accurate but at the same time more complicated elements, such as bi-cubic parametric patches. When increasing the polygonal resolution, hence the precision of the representation is increased at the same time. Nevertheless, the rendering cost is higher than usual.

A continuous representation is more accurate. The CSG representation is a combination from both discrete and continuous function. On the other hand, the object space is labeled according to object occupancy, greatly reduces the overall cost of rendering if we use volume representation.

Combining the above information, a new implicit function for both continuous and discrete volumetric representation is introduced. This kind of representation is exact, accurate and compact. Only a few of elements are needed to show an object representation is achieving accuracy better than that of bi-cubic patches. This paper has proven that neural network is able to represent any real-world or manually created objects with the concept of one function per 3D object.

5 ARTIFICIAL MULTILAYER NEURAL NETWORKS

Artificial neural network is biologically inspired model, based on the functions and structure of biological neurons. A neural network consists of numerous computational elements (neurons or nodes), highly interconnected to each other. A weight is associated to every connection. Normally nodes are arranged into layers. A multilayer perceptron is a feedforward neural network with one or more hidden layers. Typically, the network consists of an input layer of source neurons, at least one hidden layer of computational neurons. During a training procedure input vectors are presented to the input layer with or without specifying the desired output. According to these differences neural networks can be classified as supervised or unsupervised (self-organizing). Networks can also be classified according to the input values

(binary or continuous). The learning procedure contains three main steps: the presentation of the input sample, the calculation of the output and the modification of the weights by specified training rules. These steps are repeated several times, until the network is trained [Miklos Hoffmann and Lajos Varady 1998]. A multilayer perceptron with two hidden layers is shown in Figure 2 [Michael Negnevitsky 2002].

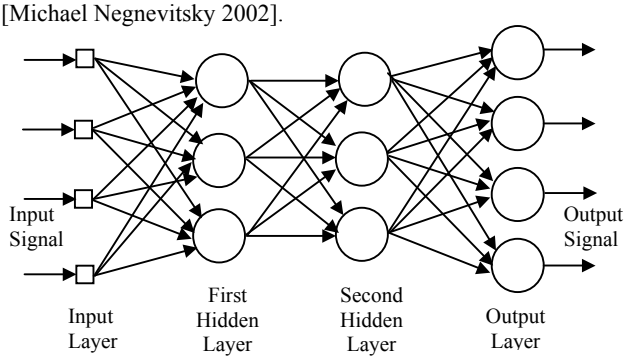


Figure 2: Multilayer perceptron with two hidden layers

5.1 BACK-PROPAGATION TECHNIQUE

Back-propagation is, by far, the most commonly used method for training multilayer feedforward networks [Emmanouil Piperakis and Itsuo Kumazawa 2001]. The term back-propagation refers to two different concepts. First, it describes a method for calculating the derivatives of the network training error with respect to the weights, by use of a clever application of the derivative chain-rule. Second, it describes a training algorithm, equivalent to gradient descent optimization which uses the derivatives from the first part, to adjust the weights in order to minimize the error.

Before back-propagation, most networks used nondifferentiable hard-limiting binary nonlinearities such as step functions, and there were no well knows general methods for training multilayer networks. The breakthrough was perhaps not so much the application of the chain-rule, but the demonstration that layered networks of differentiable nonlinearities could perform useful, nontrivial calculations and that they offer attractive features such as quick response, fault tolerance and the ability to "learn" from examples, as well as some ability to generalize beyond the training data.

6 METHODOLOGY

6.1 DATA ACQUISITION

Data is obtained from 3D *Studio Max* file (.3ds file). Object in .3ds file is represented using triangular meshes. This is due to the statement 'glBegin (GL_TRIANGLES)' in the program to load 3D object from .3ds file. Number of objects is determined to get the vertices value. The entire vertices x, y, z and the vertex indices of triangular faces are extracted from .3ds file and organized into triangular form based on the format of .3ds file by Jim Pitts (1994).

Besides extraction of surface points, the vertex normal value for each point is also very important. In the data acquisition process, the magnitude and vector of each point is calculated after running several functions in the program. Basically, face normals are first calculated, and then the average of all the normals around each vertex is taken. This can produce a better approximation for that vertex.

6.2 3D RECONSTRUCTION

In this phase, z value is estimated using vertices x and y with the aid of multilayer neural network. Object is divided into few parts before the z values are generated. As comparison, another method for 3D reconstruction which is 3rd order polynomial method is introduced. The object generated by neural network is compared with original object and also object generated using third order polynomial method.

6.2.1 3rd ORDER POLYNOMIAL METHOD

Before the actual test, an initial experiment is conducted to get a better understanding of object reconstruction. 3rd order polynomial method is used in this experiment to estimate the z values for each vertex of a polygon. Below is the equation of 3rd order polynomial:

$$Z = a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy + a_6x^3 + a_7y^3 + a_8x^2y + a_9xy^2, \quad (1)$$

where,

$a_0 \ldots a_9$: coefficient values

z, x, y: vertices object which is z-value, x-value and y-value.

Procedure to calculate coefficient value a_0 until a_9 is:

$$l = A\hat{Y}, \quad (2)$$

where,

$$A = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ : & : & : \\ 1 & x_n & y_n \end{pmatrix}, \hat{Y} = \begin{pmatrix} a_0 \\ a_1 \\ : \\ a_n \end{pmatrix}, l = \begin{pmatrix} z_0 \\ z_1 \\ : \\ z_n \end{pmatrix}$$

$$\hat{Y} = (A^TA)^{-1} A^T l .$$

Inverse metrics for metrics 10 x 10 is obtained using Gauss-Jordan Method.

Results of the study have proved that the 3rd order polynomial is able to calculate the coefficient of equation to estimate the z values. Besides, the z values for points which are not found in the input file can be estimated too. Among the disadvantages of this method are extremely high deviation and the training data must be the object itself.

6.2.2 BACK PROPAGATION ALGORITHM FOR FUNCTION APPROXIMATION

In a back-propagation neural network, the learning algorithm has two phases. First, a training input pattern is presented to the network input layer. The network then propagates the input pattern from layer to layer until the output pattern is

generated by the output layer. If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated [Michael Negnevitsky 2002].

As with any other neural network, a back-propagation one is determined by the connections between neurons (the network's architecture), the activation function used by the neurons, and the learning algorithm (or the learning law) that specifies the procedure for adjusting weights.

Below is the back-propagation training algorithm:

Step 1: Initialization

Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range (Haykin, 1994):

$$\left(-\frac{2.4}{F_i},+\frac{2.4}{F_i}\right) \quad , \tag{3}$$

where F_i is the total number of inputs of neuron i in the network. The weight initialization is done on a neuron-by-neuron basis.

Step 2: Activation

Activate the back-propagation neural network by applying inputs $x_1(p), x_2(p)\ldots x_n(p)$ and desired outputs $y_{d,1}(p), y_{d,2}(p), \ldots, y_{d,n}(p)$. P is the iteration.

(a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) + w_{ij}(p) - \theta_j\right] \tag{4}$$

(b) Calculate the actual outputs of the neurons in the output layer :

$$y_k(p) = sigmoid\left[\sum_{j=1}^{n} x_{jk}(p) + w_{jk}(p) - \theta_k\right] \tag{5}$$

where m is the number of inputs of neuron k in the output layer and w is the weight.

Step 3: Weight Training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

(a) Calculate the error gradient for the neurons in the output layer :

$$\delta_k(p) = y_k(p)\times\left[1 - y_k(p)\right]\times e_k(p) \tag{6}$$

where

$$e_k(p) = y_{d,k}(p) - y_k(p) \tag{7}$$

Calculate the weight corrections:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) - \delta_k(p) \tag{8}$$

Update the weights as the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \tag{9}$$

(b) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = y_j(p)\times\left[1 - y_j(p)\right]\times\sum_{k=1}^{l}\delta_k(p)\times w_{jk}(p) \tag{10}$$

Calculate the weight corrections:

$$\Delta w_{ij}(p) = \alpha \times x_i(p) - \delta_j(p) \tag{11}$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \tag{12}$$

Step 4: Iteration

Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.

6.2.3 Training and Output Generation

After the training process, a weight file which contains weight, architecture and parameters of the networks is created. Following are the inputs and outputs for the networks in both training and output generation process.

X and y coordinates of a clockwise Cartesian system are used as inputs for the network. For simplicity, the 3D object is scaled so that they belong to the domain of about [-15, 15].

Meanwhile, z coordinates of original 3D object are generated as network output. The entire z values is normalized to the domain of [0,1] because the output of neural network is always between 0 to 1. The object is separated into two parts according to z values before normalize. After training, the z values are denormalized again.

6.3 3D OBJECT REPRESENTATION

Object in .3ds file is represented in triangular meshes, which is also considered as one type of the object representation method. However, after the z values are estimated using only the x and y values, the object is reconstructed again from polygonal representation type to neural network representation. Below are the functions needed for neural network representation [Emmanouil Piperakis and Itsuo Kumazawa 2001]:

Error Function:

The error function measures the cost of differences between the network outputs and the desired values. The sum-of-squares error function (SSE) is a common choice.

$$E_{SSE} = \sum_{p}\sum_{i}\left(d_{pi} - y_{pi}\right)^2 \tag{13}$$

where,

P in the above equation indexes the patterns in the training set, i indexes the output nodes, and d_{pi} and y_{pi} are, respectively, the target and actual network output for the i^{th} output node on the p^{th} pattern.

The mean-squared-error is defined as:

$$E_{MSE} = \frac{1}{P\times N}\times E_{SSE} \tag{14}$$

Sigmoid Function:

Sigmoid function, being a bounded, monotonic and differentiable function is a standard activation function for back-propagation networks (Figure 3).

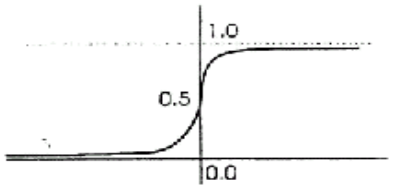


Figure 3: Sigmoid Function

To compute its first derivative an approximation is made using:

$$f(x,\sigma) = \frac{1}{1-\exp(-\sigma \times x)} \qquad (15)$$
$$f'(x,\sigma) = -\sigma \times f(x,\sigma)(1-f(x,\sigma))$$

Finally, the weights are updated using equation (18), where ŋ is the learning rate parameter.

$$\delta_i = -(d_{pi} - y_{pi}) \times f_i^1 \qquad (16)$$

$$\delta_i = f_i^1 \times \sum_k (w_{ki} \times \delta_k) \qquad (17)$$

$$w_{new} = w_{old} - \eta \times \delta_i \qquad (18)$$

The procedure is repeated until the E_{MSE} is sufficiently small, or a predefined number of iteration is reached.

6.3.1 Network Inputs

X, y and z coordinates of a clockwise Cartesian system are the inputs for the network. For purpose of simplicity and generality, we normalize the values of the coordinates to the domain of [-1, 1], meaning that x, y, z ε [-1, 1] [Emmanouil Piperakis and Itsuo Kumazawa 2001]. This normalization, resizes the 3D object to fit inside a cube of edge length equal to 2, is placed in a specific location of 3D space, surrounding the beginning of the axes in all directions with distance 1. It also has a positive effect on the training procedure. This cube is the Object Space, meaning at a space within which the 3D object is described.

6.3.2 Network Outputs

We only have neuron for the network's output, which is whether the point is part of the object or not. Due to the properties of the sigmoid function for each neuron is not Boolean value, contrary, they lie in [0, 1] range. To convert these values into Boolean values, either "yes" (1) or "no" (0), an edge threshold value, let's say is 0.5 is defined. All received values greater than this edge threshold value is denoted as Boolean "yes" and the rest as "no". Whenever the "yes" value is received, then we can know that the input point with coordinates x, y, z belongs to the 3D object.

6.4 DISPLAY AND AFFINED TRANSFORMATION

Simple affined transformation is used in order to prove that modeling process of a 3D object can be conducted with the aid of neural network. Moreover, this stage is aimed to observe the shape of the 3D object for comparison. Rotation function is called with statement 'glRotate( )' at x, y and z-axis. Meanwhile, translation and scaling processes also carried out using function 'glTranslate()' and 'glScale( )' respectively. Subsequently, the objects are displayed using vertices display function in OpenGL.

7 Results Obtained

7.1 Data Acquisition

There are seven data files created ; two files for points inside the object, two files for points outside the object, one file for vertex normals for all points, another file for vertex indices of entire triangular faces and the last file is for the exact points of the object.

7.2 3D Reconstruction

During the reconstruction process, we found that the 3rd Order Polynomial method is unable to estimate the z-value for surface object with more than 20 points if only unorganized data of an object is given. The calculated z-values are far more different from the original z-values. Hence, inaccurate output produced. Therefore, we separate out object into 12 and 20 points during reconstruction process using 3rd Order Polynomial method.

During experiments, we found that neural network is impossible to recognize and estimate z-value if there is more than one output z-value for the same vertex x and vertex y. In conjunction to it, we separate the object into two parts for reconstruction purpose. Next, these two parts are processed separately.

The parameters of the networks are shown in table 1.

Table 1: Parameter value of networks

PARAMETER	VALUE
Hidden Layer	1
Node for Hidden Layer	20
Momentum Rate	0.01
Learning Rate	0.9
Iteration	50000

The outputs are shown in three ways which are the neural network reconstruction method, 3rd order polynomial with 12 points and 3rd order polynomial with 20 points. Figure 4-6 below shows the results from the mentioned method.

Figure 4 to figure 6 showed the results of reconstruction which used every point in an object for training and output generation. The results proved that object (B) generated by neural network is more accurrate than object generated using 3rd Order Polynomial (C and D).

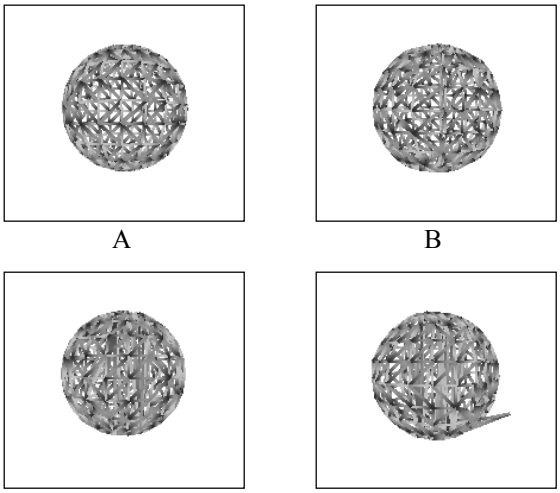


Figure 4 : Sphere after reconstruction

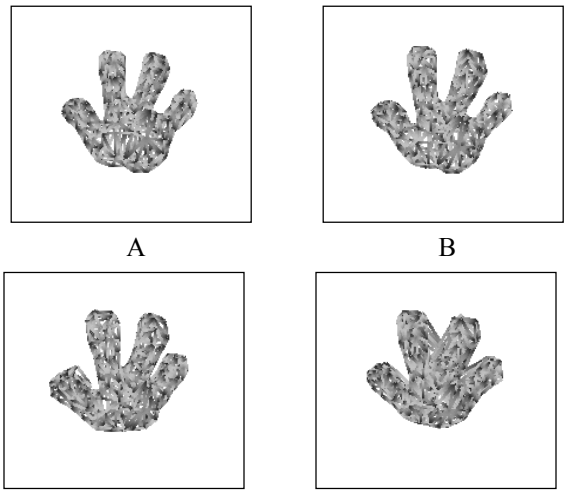


Figure 5 : Cartoon hand after reconstruction

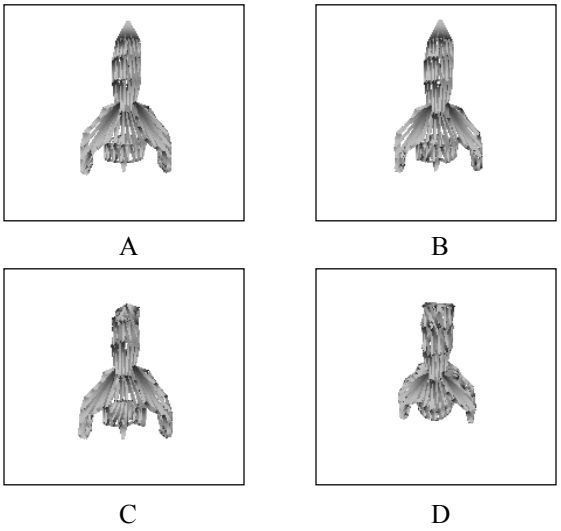


Figure 6 : Rocket after reconstruction

Figure 4-6 :
A=Original object
B=Object generated by Neural Network
C=Object generated by 3rd Order Polynomial (12 Points)
D=Object generated by 3rd Order Polynomial (20 Points)

Outputs above are created from original object, where every point in the original object are trained. Every z-value of the object is estimated during output generation process. However, neural network has its advantage as it can estimate other z-value of the points belongs to the object even only some points of the original object are given. To prove this statement, some points of the object which can show the object shape are used during the training process. The process of output generation will estimate the z-value for all the points which only consists of x and y coordinates. Figure 7-10 are some of the examples that show both the input object for training (A) and output object produced by neural networks (B).

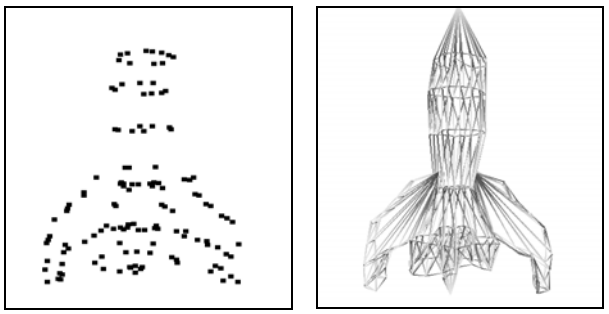


Figure 7 : Rocket after reconstruction

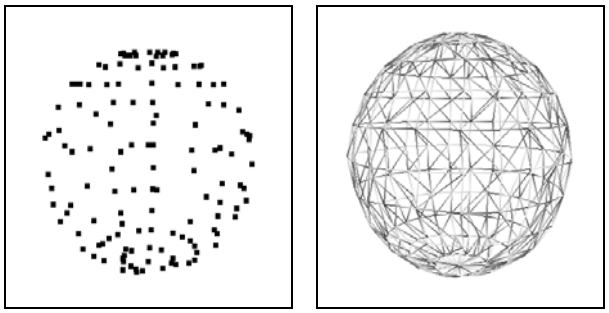


Figure 8 : Sphere after reconstruction

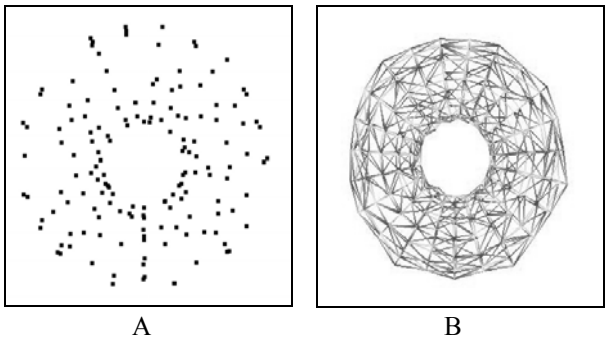
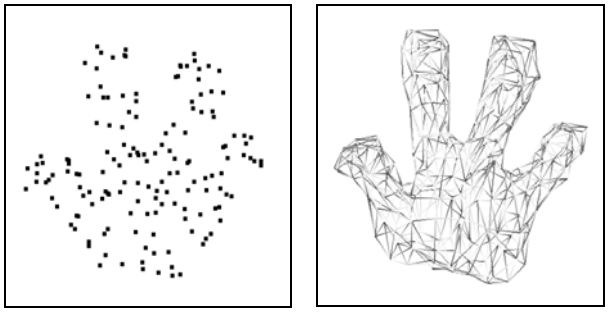


Figure 9 : Torus after reconstruction



A B

Figure 10 : Cartoon hand after reconstruction

Figure 7-10 :
A=Object with fewer points for training
B=Object generated by Neural Network

7.3 3D Representation

7.3.1 Training

In the experiments, we use neural network architecture with only 3 layers – one input, one hidden and one output layer. The output layer always has one node. The input layer have 6 nodes, which are the coordinate values x, y, z, x^2, y^2 and z^2. All the values are normalized so that x, y, z $\in$ [-1, 1]. We found that during our experiment the values from 16 to 22 hidden layer nodes are sufficient even for objects of higher complexity and curvature. Meanwhile, other parameters we used are the same as in 3D reconstruction.

Training a network by using surface points is inadequate so extra points are required to convert from a surface representation to a volumetric form. Therefore, by using a heuristic approach, we use normal values of the surface points to create 6 to 8 extra points of surfaces. A distance *d* is defined and the number of extra surfaces of points is chosen. Suppose that P (p_x, p_y, p_z) is a surface point and N (n_x, n_y, n_z) is its normal value [Emmanouil Piperakis and Itsuo Kumazawa 2001]. Then the new points P_1, P_2, P_3 and P4 are calculated using the following equations:

$$P_1 (p_x + d * n_x, p_y + d * n_y, p_z + d * n_z)$$
$$P_2 (p_x + 2* d * n_x, p_y + 2 * d * n_y, p_z + 2 * d * n_z)$$
$$P_3 (p_x - d * n_x, p_y - d * n_y, p_z + d * n_z)$$
$$P_4 (p_x - 2* d * n_x, p_y - 2 * d * n_y, p_z - 2* d * n_z), \quad (19)$$

where *d* is distance,
P_1 , P_2, P_3 and P_4 are new points for extra surfaces,
p_x,p_y and p_z are surface points,
n_x,n_y and n_z are normal values.

For each point P on the surface of the object, the expected value for the training procedure is set to 0.5. For the outside points P1 and P2, the expected value is set to 0.0 and for the inside ones P3 and P4, the expected value is set to 1.0.

7.3.2 Output Generation

The training result is tested in the specific object space which ranges from -1 to 1, with a predefined step. The visual result is displayed using a 3D window to draw points when the networks output value is greater than the edge threshold value, 0.5. Simple affine transformations are applied on those objects. The results of neural network representation are shown in figure 11-14. The original objects represented by triangular meshes are already shown in reconstruction part (figure 4-6).

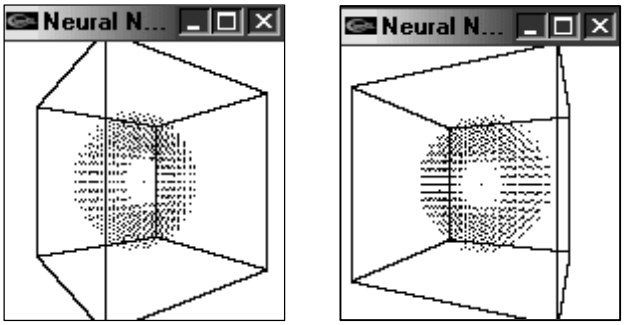


Figure 11: Torus represented using neural network

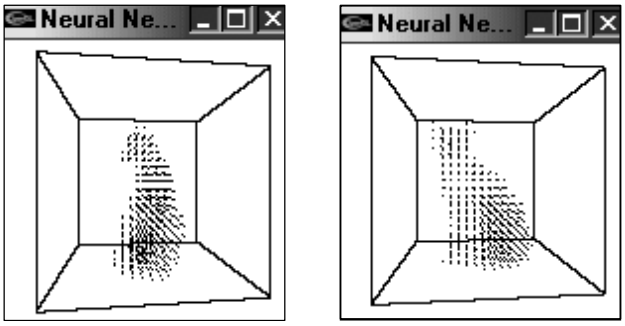


Figure 12: Cat represented using neural network

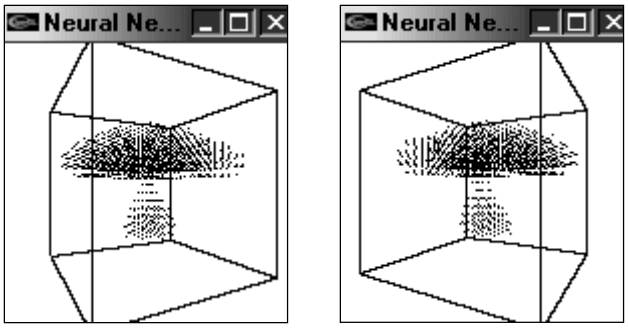


Figure 13: Mushroom represented using neural network

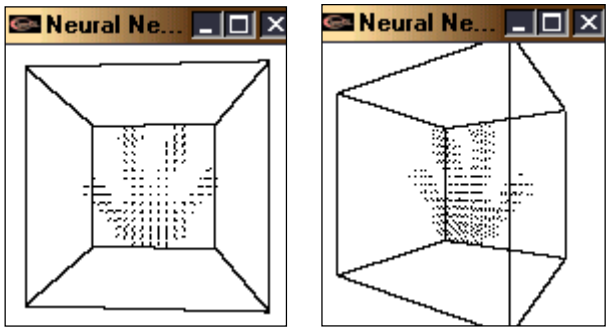


Figure 14: Cartoon hand represented using neural network

8 Comparison between Original Object and Generated Object

The objects generated using the 3rd order polynomial and neural network methods are compared with original objects in reconstruction process. Error or deviation of the reconstructed objects, which implement 3rd order polynomial, is compared with the error appeared for the objects reconstructed by neural network. However, the networks' ability to generalize and represent the object more precisely than the 3D objects used in training can only be evaluated visually.

Three tables are drawn to show the differences between original z-values and estimated z-values using three criterions which are sum, average and standard deviation . Table 2-4 shows the calculated values as comparison. The result proves that 3D reconstruction using neural network is more accurate and compact than using the 3rd order polynomial. Besides, the 3rd order polynomial method is unable to generate object with more than 20 points once.

Table 2: Comparison between original object and object generated by neural networks.

Object	Total Point	Standard Deviation	Sum of Differences	Average of Differences
Sphere	266	0.0745	25.6415	0.0964
Torus	309	0.2342	59.9357	0.1939
Mushroom	226	1.0345	172.6624	0.7639
Hand	307	0.3703	113.6136	0.3701
Rocket	260	0.6933	238.5729	0.9176

Table 3: Comparison between original object and object generated by 3rd Order Polynomial (12 points).

Object	Total Point	Standard Deviation	Sum of Differences	Average of Differences
Sphere	266	0.3460	25.2382	0.0949
Torus	309	0.7935	88.6878	0.2870
Mushroom	226	0.9532	115.0811	0.5092
Hand	307	0.7173	98.7630	0.3217
Rocket	260	2.0355	339.9537	1.2959

Table 4: Comparison between original object and object generated by 3rd Order Polynomial (20 points).

Object	Total Point	Standard Deviation	Sum of Differences	Average of Differences
Sphere	266	0.4451	51.1655	0.1924
Torus	309	0.8813	156.1833	0.5054
Mushroom	226	2.9712	375.0419	1.6595
Hand	307	0.6278	222.3982	0.7244
Rocket	260	1.0960	230.7018	0.8873

9 Conclusion

The reconstruction of 3D objects in 3D graphics systems using neural networks is an interesting subject for research. The applications of neural network representation are many, such as: Computer Vision and Error Detection based on neural network matching, Generation of 3D data for computer aided design, Databases of 3D objects for fast querying and finally 3D reconstruction. The goal of this paper is to prove that the process of reconstruction and representation of 3D object can depend completely on multilayer feed-forward neural network. The results show that object reconstructed by neural network is more accurate than object generated by 3rd order polynomial, while object represented by neural network is more exact, accurate and compact at the same time. Thus, the goal of this paper is achieved and this will be a basic step for a series of new ideas that should be explored in future.

On the other hand, research on reconstruction of 3D object is beneficial and is capable of bringing plenty of advantages for human being especially in medical realm. Conversion of images from 2D to 3D has facilitated the diagnosis process for doctors. Furthermore, viruses and tomography, which are complex objects, would also necessitate the reconstruction with estimation in order to simplify the declaration and recognition processes of researchers.

In relation to that, further research on 3D object reconstruction with the aid of neural network especially in medical field is highly recognized and expected as one the continuous effort of this project in the field of reverse engineering.

Acknowledgements

The authors are heartiestly appreciate Ministry of Science and Technology(MOSTE) for the IRPA grant and Research Management Center(RMC), University of Technology Malaysia for the support in making this project a success.

References

MICHAEL JUSCHKE. 1996. True Colour Holography. Deg Thesis, Department of Electrical and Electronic Engineering: University of Western Australia.

WOLFGANG NIEM, JOCHEN WINGBERMUHLE. 1997. Automatic Reconstruction of 3D Objects Using a Mobile Monoscopic Camera. In *Proceedings of the International Conference on Recent Advances in 3D Imaging and Modelling",*Ottawa, Canada

VIT ZYKA, RADIM SARA. 1997. Scale Model Refinement For 3D Reconstruction. Center of Machine Perception: Czech Technical University.

RUIZ, OSCAR EDUARDO, CADAVID, CARLOS ALBERTO, MIGUEL. 2001. Evaluation of 2D Shape Likeness for Surface Reconstruction. *XIII International Congress on Graphics Engineering*.

MICHAEL NEGNEVITSKY. 2002. Artificial Intelligence. A Guide to Intelligent Systems. Pearson Education Limited 2002, England, 164-178.

YING WU. 2001. An Introduction to ECE510 Computer Vision. Electrical and Computer Engineering: Northwestern University, Evanston.

INGE SODERKVIST. 1999. Introductory Overview of Surface Reconstruction Methods." Department of Mathematics: Lulea University, Sweden.

EMMANOUIL PIPERAKIS, ITSUO KUMAZAWA. 2001. Affine Transformations of 3D Objects Represented with Neural Networks. *3-D Digital Imaging and Modeling,Proceedings.* 213-223.

HOD LIPSON, MOSHE SHPITALNI. 2001. Correlation-Based Reconstruction of a 3D Object from a Single Freehand Sketch. Cornell University, University of Michigan, USA.

VOLKER KRUGER, GERALD SOMMER. 2000. Gabor Wavelet Networks for Object Representation and Face Recognition. Kiel, Germany.

GARY BRADSKI, STEPHEN GROSSBERG. 1995. Fast Learning VIEWNET Architectures for Recognizing 3D Objects from Multiple 2D Views. *Neural Networks Special Issue on Automatic Target Recognition*.

YOSHIHIRO KAWAI, TOSHIO UESHIBA, TAKASHI YOSHIMI, MASAKI OSHIMA. 1992 Reconstruction of 3D Objects by Integration of Multiple Range Data. Japan.

JOSE GASPAR, ETIENNE GROSSMANN, JOSE SANTOS-VICTOR. 2001. INTERACTIVE RECONSTRUCTION FROM AN OMNIDIRECTIONAL IMAGE. *9th International Symposium on Intelligent Robotic Systems*.

ANDERS HEYDEN. 1995. Geometry and Algebra of Multiple Projective Transformations. Dept of Mathematics, Lund University, Sweden.

MIKLOS HOFFMANN, LAJOS VARADY. 1998. Free-form Surfaces for Scattered Data by Neural Networks. *Journal for Geometry and Graphics*. Vol.2. 1-6.

JOARDER KAMRUZZAMAN, S.M.AZIZ. 2002. A Note on Activation Function in Multilayer Feedforward Learning. Monash University, University of South Australia, Australia.

CS. SZEPESVARI, A. LORINCZ. 1994. Self-Organized Learning of 3 Dimensions. Hungary.

TOHRU NITTA. 1994. Generalization Ability of the Three-Dimensional Back-Propagation Network. Ibaraki, Japan.

AC EVANS, NA THACKER, JEW MAYHEW. 1993. A Practical View Based 3D Object Recognition System. University of Sheffield.

RICHARD S. YOON, DON S. BORRETT, HON C. KWAN. 1995. Three-Dimensional Object Recognition Using a Recurrent Attractor Neural Network. *IEEE-EMBC and CMBEC Theme 2: Imaging*.