3D Polygon Mesh Compression with Multi Layer Feed Forward Neural Networks

Emmanouil Piperakis
Tokyo Institute of Technology, Department of Computer Science,
West 8, Ookayama 2-12-1, Meguro, Tokyo, Japan
epiperak@kml.cs.titech.ac.jp

Itsuo Kumazawa
Tokyo Institute of Technology, Department of Computer Science,
West 8, Ookayama 2-12-1, Meguro, Tokyo, Japan
kumazawa@kml.cs.titech.ac.jp

Abstract

In this paper, an experiment is conducted which proves that multi layer feed forward neural networks are capable of compressing 3D polygon meshes. Our compression method not only preserves the initial accuracy of the represented object but also enhances it. The neural network employed includes the vertex coordinates, the connectivity and normal information in one compact form, converting the discrete and surface polygon representation into an analytic, solid colloquial. Furthermore, the 3D object in its compressed neural form can be directly - without decompression - used for rendering. The neural compression - representation is viable to 3D transformations without the need of any anti-aliasing techniques - transformations do not disrupt the accuracy of the geometry. Our method does not suffer any scaling problem and was tested with objects of 300 to 10^7 polygons - such as the David of Michelangelo - achieving in all cases an order of $O(b^3)$ less bits for the representation than any other commonly known compression method. The simplicity of our algorithm and the established mathematical background of neural networks combined with their aptness for hardware implementation can establish this method as a good solution for polygon compression and if further investigated, a novel approach for 3D collision, animation and morphing.

Keywords: Polygon Compression, 3D Representation, Neural Network, 3D Modeling, Implicitization

1. INTRODUCTION

3D geometry compression is a demanding Computer Graphics problem, which has been thoroughly investigated by many researchers. As the accuracy of representation increases due to the unceasing advance of the available scanning hardware, the size of the 3D models becomes prohibiting for internet based graphics applications, medical databases, virtual 3D world modelling, fast rendering and other applications. In this paper, we present a compression method that utilizes Neural Networks for both compressing and representing 3D objects. In the first part, we mention the most recent literature on representation methods and geometry compression techniques and we discuss the merits of our approach. In the second part we make a brief introduction of the mathematical background and the neural network architecture employed. In the third section our method is presented in detail and compared to the most popular, previously discussed, schemes of representation and compression. Finally, we discuss about the advantages and applications of our method followed by possible enhancements and our future goals.

Related work

Discussion and even enumeration of the current representation methods is a tedious task, therefore we restrain in pointing out some new methods. Some of the latest ones are the mesh based descriptions like the polygon, the bezier surfaces, the quadric and superquadric, the signal processing meshes, the hierarchical B-Splines, the wavelet schemes, the subdivision surface schemes and many more. The most dominant in CAD/CAM applications is the Constructive Solid Geometry (CSG) followed by some newer variations like the R-functions, B-rep, skeletal methods, accumulation modelling and volume based metamorphosis. Finally, implicit methods like blobby molecules, metaballs, soft objects, implicit deformations and patches, iterative, fractal and iso-surfaces are becoming prominent in specific kinds of applications. A detailed recent survey of most of these methods is available by [6]. For the basics on 3D Graphics the reader should study [3],[15].

The compression of 3D objects is generally achieved by compressing the triangle/vertex incidence graphs of their polygon based representation. Those algorithms are categorized by the types of objects they can be applied to: a) Height fields and parametric surfaces are usually compressed by triangulation, regular grid, hierarchical subdivision, feature refinement, decimation or optimal methods. b) Manifold Surfaces, are only compressed by refinement and decimation methods. c) Non-manifold surfaces. The latest methods that achieve the highest compression ratio are the Edgebreaker [13], the Face Fixer [7] and others [8], [5] and [14]. Finally there are some methods for compression of volumetric objects by [1] and lately introduced by [11] and [2].

Merits

Combining all the above information and extending the idea of using neural networks for 2D image representation [9], [10], we created a new representation that is an implicit function for both continuous and discrete volumetric representation. Our representation is an exact and lossless compression method. Only few elements are needed to represent an object achieving accuracy better than many other methods. In this paper we prove that this function - one function per 3D object - exists, and can represent any kind of real world or manually created object. We introduce neural network training techniques as an implicitization from real data. The advantages of our method are in detail presented in section 3.1.

2. EXPERIMENT

Algorithm

The network employed by our method is a multi layer feed forward network. All nodes are fully connected and the threshold values are set to 1. As an activation function for each node we use the sigmoid $f(x, \sigma) = 1/[1 + \exp(-\sigma x)]$. Also, $f'(x, \sigma) = -\sigma f(x, \sigma)(1 - f(x, \sigma))$ computes the first derivative of the sigmoid. The network takes as input the x, y, z coordinates and also x^2,y^2 and z^2 - normalized in $[0, 1]$ and produces one output with values in [0,1]. For the forward pass we use $a_i = \sum_j (w_{ij} y_j)$ and $y_i = f(a_i, \sigma)$ where y_i is the weighted summation a_i passed through the sigmoid, for each node. The output is compared to the expected value and the error δ_i is computed with $\delta_i = -(d_{pi} - y_{pi}) f'_i$. Afterwards the error is backwards propagated and computed for each hidden node using $\delta_i = f'_i \sum_k (w_{ki} \delta_k)$ and the weights are updated with $w_{new} = w_{old} - \eta \delta_i$ until the mean average squared error $E_{MSE} = \frac{1}{PN} (\sum_p \sum_i (d_{pi} - y_{pi})^2)$ is minimized. For more details about the multi layer feed forward networks and the back propagation training procedure refer to Haykin [4].

Training a network to represent the geometry of a 3D object is done in 3 steps: a) the vertices and their corresponding normal values of a 3D surface model are used to create a training set, b) the structure and parameters of the network are chosen, c) the network is trained multiple times with the training set, until the E_{MSE} is adequately small. Finally we produce the visual results and calculate the error values.

It was proven that training a network with only the surface vertices was not adequate. Therefore we use a simple heuristic to create extra training examples; for each vertex we use its normal value to create points lying inside and outside, in the vicinity of the surface point. Then we assign expected values for each point; for the surface points we assign the value 0.5, for the inner the value 1.0 and for the external ones the value 0.0. This simple heuristic proves sufficient although it sometimes fails - in the cases of sharp edges and spikes. The symmetry of the geometry and the fault tolerance of the neural network minimize the error caused by the heuristic.

The network we use has 6 input nodes and 1 output. We proved that only one hidden layer is adequate for representing complicated objects. The only unknown variable is the number of hidden neurons. There is no known efficient, fail proof way of determining this number for the specific problem. Depending on the complexity of the object 6 to 50 hidden nodes are needed for compression. Initial values between 15 to 20 usually provide an accurate result. For higher accuracy either the number of hidden nodes or the number of bits used for representing the weights of the network should be increased.

During the training procedure we observe E_{MSE} and based on its value we determine if the training is succeeding or not and whether we need to increase of decrease the number of hidden neurons. Finally, when E_{MSE} has decreased at least on order of $O(n)$, we test the result of the network in a unit cube by evaluating the network for each point in the cube with a predefined specified density of samples. When the output is greater than 0.5 then we draw a point in the unit cube, meaning we are inside the object.

Examples

Our set of trained 3D Objects was specifically chosen to prove that different classes of surfaces can be

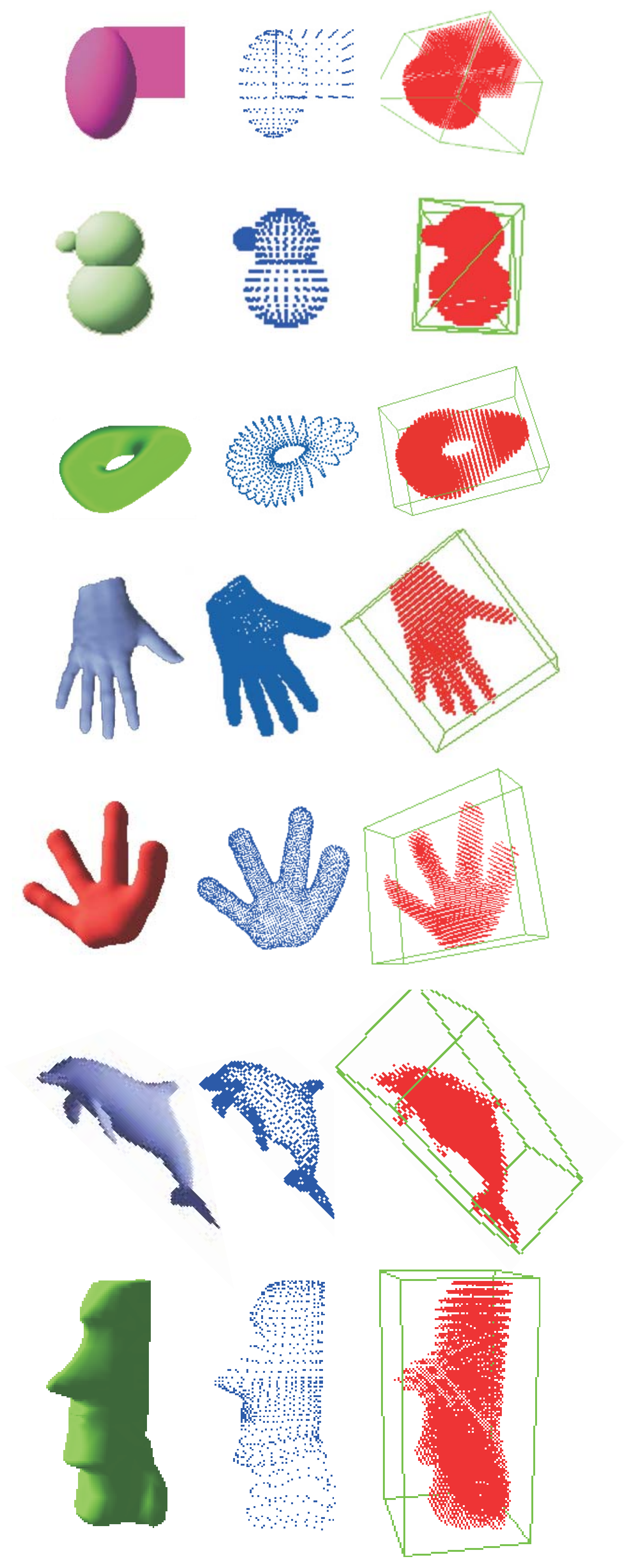
Figure 1: Results of Compression: Column 1; Objects in CG, Column 2; Surface Points, Column 3; Generated result of trained object.

represented - compressed. Objects with holes, sharp edges, products of boolean functions, slim surfaces and real world, complicated objects were successfully compressed. With minor modifications, our method can represent particle systems and flat or hollow objects. In our attempt to prove that our method does not suffer a scaling problem we used 3D models consisting of 300 to 410^6 and more points. The training time for the later models is considerable for Computer Graphics standards (approximatively 3 days to 1 week), but given the fact that training is analogous to the scanning of the 3D object and has to occur only once, the price to pay for such a high compression ratio is small. On figure 1 we display some of the successfully compressed objects and on figure 2 the low accuracy compressed object - training has not yet been completed due to time limitations.

Results and Comments

Our experiment was conducted with 2 custom made tools coded using java on a PC Pentium III 600MHz. The first program was used for the training of neural networks to represent/compress 3D objects and the second program performed rendering using the previously created networks.

From figure 1 it is clear that the networks represent solid 3D objects and not surfaces. This is a result of the generalization property of the network. The normal values can be easily and analytically computed from the network without any extra mathematical overheat [4], and therefore there is no need to store them. A comparison between our method and the most popular compression colloquial is displayed in figure 4 from which it is obvious that our method achieves at least an order of 3, $O(b^3)$, better compression with regards to number of bits used for storage of the objects geometry. From figure 3 it is clear that the accuracy is small enough to be acceptable for most kinds of applications. We measure the error as the distance of the actual edge point used for training, to the point in the normal direction, where the network evaluates as edge - equal to 0.5. The maximum error was 0.02% of the objects' size.

3. CONCLUSIONS

Advantages

Based on the results of the experiments, we can assume that the compression of 3D objects with the use of neural networks is not only possible, but also offers certain advantages in solving some of the major problems of computer graphics representation. It combines the implicit and volumetric representation

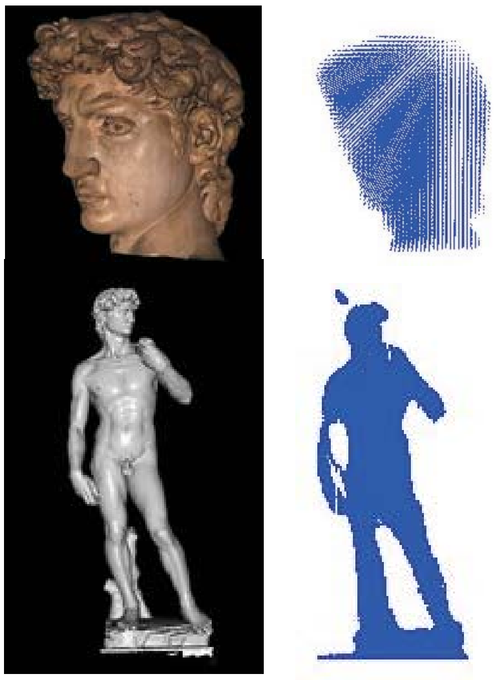

Figure 2: Generated Results only after the 2% of the training procedure. Left Column: Model to train, Right Column: Result.

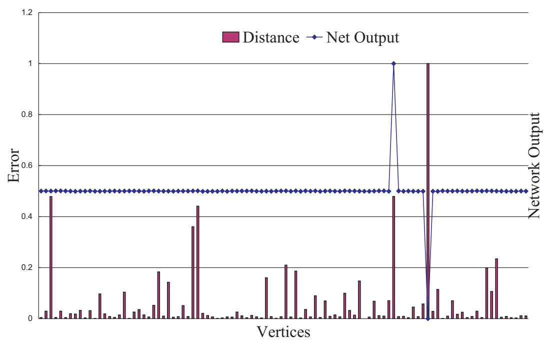

Figure 3: Error and corresponding Network Output.

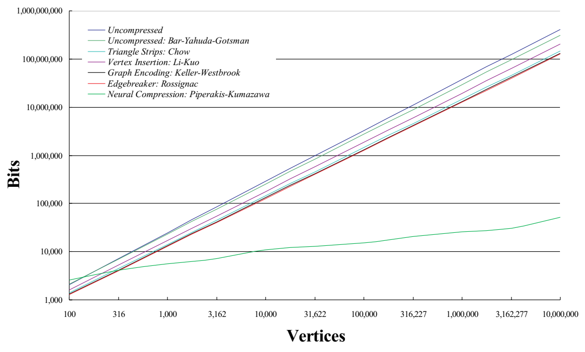

Figure 4: Compression chart in bits per vertex.

methods and inherits the advantages of the polygon and bi-cubic methods by being accurate and simple, without inheriting their drawbacks.

The representation of our method is continuous and partial derivatives can be computed at any point inside the object space. Normal values for the object are not stored, instead they are calculated at any given point of the object. Therefore no interpolation method is needed to compute the normal values. The generalization of the network enhances the representation of curves even further than the original training example, a result proving extremely useful in reconstruction of 3D objects based on real world objects. Using the parameter of the sigmoid function σ, transformations like blur, rough surface, and soft shadows can be implemented instantly.

Affine and non-affine transformations like rotations, scaling, moving, projections, twisting and others can be performed by using a neural network, connected to the object network, to transform the object space of the 3D object [12]. Object collision, based on volumetric or analytical function techniques is possible, in a fast and easy way. Anti-aliasing is not necessary, and resizing can be achieved by normalizing the object space in a domain other than $[0, 1]$, without affecting the quality or the size of the representation. Encapsulation of the representation is also achieved because of the input/output mapping property of neural networks. Morphing of 3D objects could easily be implemented by modifying the weights of one network to match the weights of another with a linear interpolation.

Applications - Future work

The obvious application of our method is compression of 3D models. Other future applications include: 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 from 2D Silhouettes. Our goal however is 3D Modelling and Rendering therefore the other applications have not been investigated. They still remain an open area for future research.

References

[1] Ali Bilgin, George Zweig, and Michael W. Marcellin. Efficient lossless coding of medical image volumes using reversible integer wavelet transforms. *Proceedings DCC'98 (IEEE Data Compression Conference)*, 1998.

[2] Ali Bilgin, George Zweig, and Michael W. Marcellin. Three - dimensional image compression with integer wavelet transforms. *Applied Optics: Information Processing, Special Issue on Information Theory in Optoelectronic Systems, Vol. 39, No. 11, pp. 1799-1814*, 2000.

[3] J. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Second Edition, Reading, MA: Addison - Wesley, 1990.

[4] S. Haykin. *Neural Networks, A Comprehensive Foundation.* Macmillan College Publishing Company, 1994.

[5] Toshiki Hijiri, Kazuhiro Nishitani, Tim Cornish, Toshiya Naka, and Shigeo Asahara. A spatial hierarchical compression method for 3d streaming animation. *ACM, VRML 2000, Monterey, CA USA*, 2000.

[6] Andreas Hubeli and Markus Gross. A survey of surface representations for geometric modeling. *citeseer.nj.nec.com/hubeli00survey.html*, 2000.

[7] Martin Isenburg and Jack Snoeyink. Face fixer: Compressing polygon meshes with properties. *UNC Technical Report TR-00-04*, 2000.

[8] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. *ACM, SIGGRAPH 2000, New Orleans*, 2000.

[9] I. Kumazawa. *Compact and Parametric Shape Representation by a Tree of Sigmoid Functions for Automatic Shape Modeling.* Pattern Recognition Letters 21, 651-660, 2000.

[10] I. Kumazawa. *Target Tracking by Matching a Shape Represented by a Tree of Sigmoid Functions.* Pattern Recognition Letters 21, 661-675, 2000.

[11] Peter Lindstrom. Out-of-core simplification of large polygonal meshes. *ACM, SIGGRAPH 2000, New Orleans*, 2000.

[12] E. Piperakis. *Affine Transformations on 3D Objects Represented with Neural Networks.* IEEE, Proceedings of the Third International Conference on 3-D Imaging and Modeling, 2001.

[13] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1*, 1999.

[14] Richard Souther, Edwin Blake, and Patrick Marais. Gems: Generic memoryless polygonal simplification, 2000.

[15] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques, Theory and Practice.* ACM Press, Addison - Wesley Publishing Company, 1994.