Neural Mesh Ensembles

Ioannis Ivrissimtzis
MPI Informatik
ivrissim@mpi-sb.mpg.de

Yunjin Lee
POSTECH & MPI Informatik
jin@postech.ac.kr

Seungyong Lee
POSTECH & MPI Informatik
leesy@postech.ac.kr

Won-Ki Jeong
University of Utah
wkjeong@cs.utah.edu

Hans-Peter Seidel
MPI Informatik
hpseidel@mpi-sb.mpg.de

Abstract

This paper proposes the use of neural network ensembles to boost the performance of a neural network based surface reconstruction algorithm. Ensemble is a very popular and powerful statistical technique based on the idea of averaging several outputs of a probabilistic algorithm. In the context of surface reconstruction, two main problems arise. The first is finding an efficient way to average meshes with different connectivity, and the second is tuning the parameters for surface reconstruction to maximize the performance of the ensemble. We solve the first problem by voxelizing all the meshes on the same regular grid and taking majority vote on each voxel. We tune the parameters experimentally, borrowing ideas from weak learning methods.

1. Introduction

With the recent advances in 3D laser scanner technology, large geometric data sets, consisting of millions of points, are becoming the standard output of the acquisition stage of the graphics pipeline. In order to process these data, we need specialized algorithms that can cope with very large and noisy point sets. Neural networks and learning algorithms are good candidates for this task as demonstrated by their numerous successful applications in problems involving the processing of voluminous noisy data. Several surface reconstruction techniques have been developed based on different kinds of neural networks [8, 28, 29, 3, 12, 13, 11, 10].

In this paper, we use the technique of neural network ensembles to boost the performance of the surface reconstruction algorithm described in [10], which we call *Neural Meshes*. As it is usually the case with a learning algorithm, there are two main areas where boosting the performance of the basic algorithm is particularly needed. The first is the speed and the second is avoiding convergence to a local minimum, i.e., capturing correctly the global shape of the input data. Here we deal with the second problem only. The main idea is to exploit the probabilistic nature of the algorithm. We run the algorithm many times on the same input data set and then construct an average of the outputs.

Usually the speed and the accuracy of an algorithm are related to each other, where higher accuracy comes at the expense of speed and vice versa. Fortunately, in our case, we can safely separate them. The problem with the local minima arises at the early stages of the reconstruction while the problem with the speed at much later stages. The reason is that a neural mesh is only locally processed and quickly gets refined enough so that no significant changes occur in its global shape. For a typical model, the convergence to a local minimum or not is already decided after the first few thousand vertices. On the other hand, the problem with the speed arises when the neural mesh is large and the inquiries on it, especially finding the most and the least active vertices, becomes very costly. When the algorithm runs on a commodity PC, the speed becomes a problem only after several tens of thousands of vertices.

1.1. Related Work

The basic surface reconstruction algorithm we use here is described in [10]. It is based on an incrementally expanding neural network known as Growing Cell Structures (GCM's) [6]. The latter is a special kind of Kohonen's Self-Organizing Maps (SOM's) [16]. Similar surface reconstruction algorithms based on the same type of neural network can be found in [12, 13, 11].

SOM's and GCS's have already found many applications in geometric modeling and visualization problems. In [7], SOM's are used for the visualization of multi-dimensional data. In [8], SOM's are used for free-form surface reconstruction and in [28], GCS's are used for the same purpose.

SOM's have been used in surface reconstruction [29] and for grid fitting [3].

On the other hand, the majority of the other surface reconstruction techniques are based on a geometric construction, e.g., tangent plane fitting on k-neighborhoods [9], α-shapes [2, 25], spline fitting [17], 3D Voronoi diagrams [1], radial basis function fitting [4], and multi-level local fitting [19]. A common characteristic of these methods is that they are deterministic. For given input data, they always output the same model which may or may not be satisfactory. The latter case happens quite often because of the error contained in the input, which has as main sources the optics of the scanning procedure and the registration of the range images. In fact, surface reconstruction from scanned data is still a procedure requiring a considerable amount of human intervention, making it the bottleneck in the creation of 3D models.

In contrast, if a probabilistic algorithm fails in producing a satisfactory model, we can always run it again and hope that this time we obtain a better model or that we will be able to combine several of the outputs to generate a satisfactory model. The technique of combining several models to obtain a better one has been studied extensively, especially in the context of supervised learning, e.g., [22, 5, 20, 23]. We discuss the techniques in more details in Section 3.2. Interestingly, their main result is counter-intuitive. The optimization of the combined model does not require the optimization of each one of the constituent models.

1.2. Overview

The stages of the overall algorithm we propose here can be summarized as follows;

Neural Mesh Ensemble Algorithm

Stage 1: Run the neural mesh algorithm many times and create many coarse models.

Stage 2: Voxelize all the models on the same grid and find an average model with majority vote on each voxel.

Stage 3: Use the Marching Cube algorithm to find an average coarse mesh from the average voxelized model.

Stage 4: Run the neural mesh algorithm starting from the average coarse mesh.

2. Neural Meshes

The neural mesh we use here is a learning algorithm for surface reconstruction described in [10]. It starts with an initial triangle mesh $\mathcal{M}$, usually a tetrahedron, and learns from training data obtained from a point cloud $\mathcal{P}$ with random sampling. It expands its connectivity by splitting its most active vertices and removing its least active vertices. It creates boundaries by removing large triangles and creates handles by merging two boundaries that are near to each other. The algorithm can be summarized as follows;

Basic step:

- Sample the target space $\mathcal{P}$ and return one point s.
- Find the vertex v_w of $\mathcal{M}$ which is nearest to s and move it towards s.
- Smooth the 1-ring neighborhood of v_w.

Connectivity change:

- *Vertex split:* Split the most active vertex.
- *Half-edge collapse:* Remove the least active vertex.

Topology learning:

- *Triangle removal:* Remove triangles with large areas.
- *Boundary merging:* Merge two boundaries that are close to each other.

The last topology learning part is not used when we generate the neural meshes for the inputs of the ensemble algorithm. This enables us to have closed surfaces for the neural meshes, which makes the voxelization step easier and more robust. However, the part is included to allow topology changes when the neural mesh is refined in Stage 4 of the ensemble algorithm.

Fig. 1 shows an example of neural mesh reconstruction. The initial data from the Digital Michelangelo archive contain 240 million points. The algorithm processed 4 million points, with about 40 million iterations of the basic step. The reconstruction consists of 0.5 million points. For an analysis of the basic algorithm, we refer the reader to [6, 12, 11, 13, 10].

3. Neural Mesh Ensembles

A neural mesh ensemble is a collection of neural meshes created using the same input data set. We show that by running the neural mesh algorithm many times and averaging the results we are able to recover the correct global shape of the input point cloud with a higher probability than a single run of the algorithm would allow.

3.1. Competing Features in Neural Mesh Reconstruction

For a better understanding on how the ensembles boost the neural mesh performance, we first discuss how the problems with convergence to local minima appear. A neural mesh, as a special type of Growing Cell Structures, learns the input data through a *competitive learning* process. The

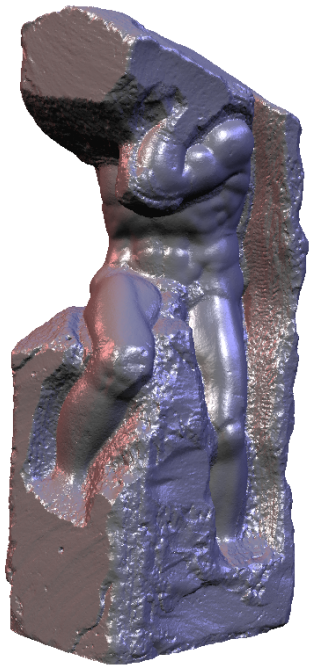

Figure 1. A neural mesh reconstruction of the *atlas* model with 500K vertices.

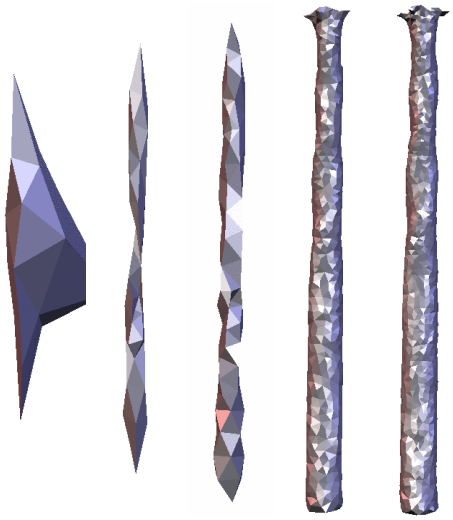

Figure 2. Neural mesh reconstructions of the Drill model with 20, 50, 100, 1K, 2K vertices.

term competition is used in the literature to describe the fact that only the vertex of the mesh which is nearest to the sample, the so called *winner*, will learn directly from it. The problem of convergence to a local minimum arises from the conflicting influence of the various features of the input data set during the competitive learning process.

It is worth stressing that the convergence to a local minimum does not come from the features themselves but from the conflict between two or more of them. As an example, Fig. 2 shows several stages of the reconstruction of the very elongated *Drill* model. In the early stages of the reconstruction, the two ends of the neural mesh are very active, containing the winners for all the samples from the part of the data set not covered yet by the mesh. Nevertheless, there is no conflict between these two very active areas because one expands upwards and the other downwards. As a result, the drill model can be correctly reconstructed by a single neural mesh with a high probability.

A completely different situation arises when there is conflict between two or more features. Fig. 3 shows such an example where four of the fingers are considered as competing features. As a result of the conflict, the algorithm may fail to properly distinguish between them. We notice that two kinds of errors can appear, false bridges between two fingers and false gaps on a finger. We also notice that there are no local minima on the thumb because it is quite far away from the other fingers.

On the other hand, it would be a mistake to think that the probability for a correct reconstruction always decreases with the distance between competing features. See for example the eleven reconstructions of the hand model in Fig. 8. We notice that the probability for a local minimum between the pointer and the index fingers is higher than between the index and the ring, despite the larger distance between the pointer and the index. The reason is that the space between the index and the ring fingers, rather than the fingers themselves, is a single concave feature which is learned relatively easily.

From the above discussion, it is clear that we should look for a statistics based solution to the problem of local minima, rather than try to investigate when this problem occurs and find an ad hoc remedy for each case. The ensemble is such a statistical approach, based on the observation that even though false bridges and gaps may appear with high probabilities in the reconstruction of a model, it is unlikely that these false bridges or gaps will appear at the same place in two different reconstructions. Moreover, it is highly unlikely that false bridges and gaps will appear at the same place in the majority of the runs of the algorithm, provided that we run it sufficiently many times. After the voxelization, the false bridges will correspond to misplaced voxels and the gaps will give missing voxels. According to the statistical property of false bridges and gaps, a majority vote on each voxel should iron out the local minima.

Figure 3. Even if the creation of a false bridge or gap is very likely, the probability that they will appear at the same place in two different reconstructions is low.

3.2. Weak Learning

The combination of several probabilistic models to obtain one with better properties is an intuitively appealing technique which has been applied in a variety of problems, achieving many times spectacular results. In supervised learning, where it has been studied more widely, the technique is known under the general term *boosting* [23]. There, the main problem one has to deal with is how to construct new models and append them on the ensemble, so that under certain assumptions its performance is provably boosted. In the unsupervised learning, which the neural meshes belong to, the situation is more difficult as we cannot incorporate in the procedure any feedback from testing the previous models. Nevertheless, in both cases, the same central question arises; what should be the properties of the constituent models in order to maximize the performance of the ensemble.

One may assume that the better the properties of each constituent model, the better the performance of the ensemble. However, usually this is not the case. To see this with a simple example, consider a categorical model with two categories, e.g., a YES or NO output, which in our case can represent the existence of a voxel or not. On one extreme, if we have a deterministic algorithm outputting models with correct answers in the same 90% of the voxels, then obviously any averaging cannot improve on this 90% rate of success. On the other extreme, if we have a probabilistic algorithm such that for *every* output model the probability of a correct answer is 51% for *any* given voxel, then in the limit, i.e., after averaging many models, the success ratio will converge to 1 for all voxels. This general idea of combining slightly better than random answers to make a highly accurate prediction comes in the literature under the name of *weak learning* [22].

In the setting of neural meshes, assume that we can tune their parameters to achieve one of the following objectives:

- The probability for the existence of local minima is as low as possible.
- The expected number of local minima is as low as possible.
- The places where local minima appear are as random as possible.

From the above discussion, it is clear that although the first two criteria are more reasonable to measure the performance of a single neural mesh, an ensemble would work better when applied to the meshes generated with the third criterion. This means that we should not be interested in lowering the probability for a local minimum to occur but in avoiding convergence to the same local minimum as we run the algorithm repeatedly.

In the neural meshes, the first two criteria and the third are many times conflicting. Indeed, for a usual 3D model, to minimize the probability for a local minimum to occur, one has to adopt a conservative strategy for the expansion of the neural mesh. The strategy includes many iterations of the Basic Step before each vertex split, repeated smoothing for a winner vertex movement, and a well-balanced frequency of half-edge collapses. However, such a conservative strategy is more prone to fail at the exactly same places every time, which makes the outcome more deterministic. In contrast, to satisfy the third criterion for a good member of an ensemble, we have to increase the flexibility of each neural mesh, which usually has a cost at the accuracy of the individual reconstruction.

Note that this conflict between accuracy and the dispersion of errors is a common problem in many applications and partly explains the popularity of weak learning methods.

3.3. Parameter Tuning

To have a better input for the ensemble, we tune the parameters of neural mesh reconstruction. For simplicity, we have all the neural meshes of the ensemble running with the same set of parameters, although, according to the above discussion, it is probably a better strategy to run the meshes with different parameters.

In an analogy between flexible neural networks like the Growing Cell Structures and the *active surfaces* [26, 27, 21], we can see the evolution of the neural mesh as controlled by some physical forces. For example, the movement of the winner towards the sample and the subsequent smoothing can be respectively considered as an external and an internal force applied on the mesh. Nevertheless, as they act locally on the mesh, they are not suitable for tuning the ensembles.

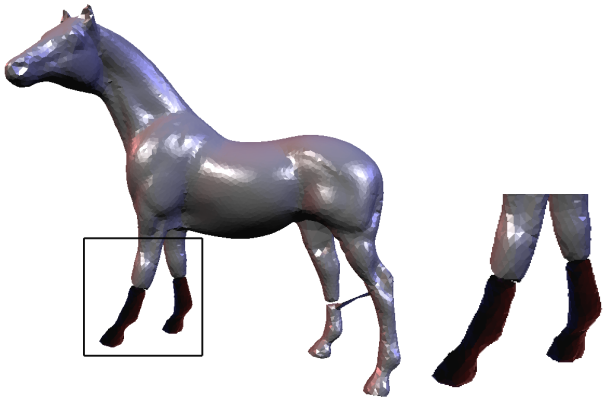

Figure 4. As the frequency of half-edge collapses increases, the surface becomes more flexible and several twists may appear.

Instead, to increase the global flexibility of the mesh, we increase the ratio C_{vs}/C_{ec} of the two parameters C_{vs} and C_{ec}. These parameters control how often we split the most active vertex of the mesh and how often we remove the least active vertex. In particular, we set $C_{vs} = 50$ as in [10], meaning that we split the most active vertex of the mesh after 50 iterations of the Basic Step. But we use $C_{ec} = 100$ instead of $C_{ec} = 250$ in [10], meaning that we remove the least active vertex and further check for other underperforming vertices every 100 iterations of the Basic Step.

As one intuitively expects, the global flexibility of the neural mesh increases considerably with a larger number of vertex removals. This property can be verified experimentally as shown Fig. 4. The extra flexibility may appear in the form of "twists" of the surface which usually do not appear when we follow a more conservative strategy for the expansion of the neural mesh.

3.4. Voxelization and Majority Vote

To average meshes with different connectivity, we first voxelize them on the same regular grid. We used the Depth-Buffer-Based voxelization algorithm [14], which is simple and does not take into account the orientation of a surface. The latter property means that twists similar to the one shown in Fig. 4 or an orientation change of the initial tetrahedron, as shown in some examples in Fig. 8, do not create any particular problem. We used the source code made publicly available by the authors with some modifications.

In the experiments, we used 7 bits for each dimension of voxelization, i.e., a 128^3 cubic grid. Although it is important to use a fine grid so that a voxelized model contains enough shape information, a very fine grid would unnecessarily complicate the situation by preserving details which

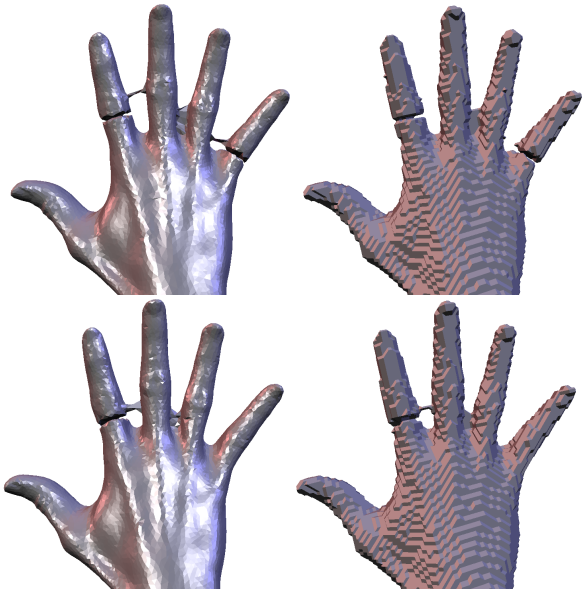

Figure 5. In the top row, the voxelization filters out the false bridge between the pointer and the index fingers. This does not happen in the bottom row.

can easily be captured at Stage 4 of the ensemble algorithm. In particular, a very fine grid would reproduce in details all the local minima we want to smooth out. Instead, Fig. 5 shows that the voxelization process, by its own, can filter out some of the local minima.

After the voxelization process, each voxel has a value of 0 or 1, depending on whether it belongs to the voxelized model or not. A voxel has the value of 1 in the average voxelized model if it belongs to the majority of the corresponding voxels in the models of the ensemble.

3.5. Meshing

The average voxelized model can be considered itself as the output of the ensemble. However, as our aim is to construct a detailed triangle mesh rather than a coarse voxelized model, we mesh this average voxelized model using the marching cube algorithm [18]. The obtained triangle mesh is used as the initial coarse mesh input for single neural mesh reconstruction and gets refined into a detailed triangle mesh.

4. Results

As we mentioned in Introduction, the convergence to a local minimum is determined at an early stage of the reconstruction. All operations on the neural mesh are local and

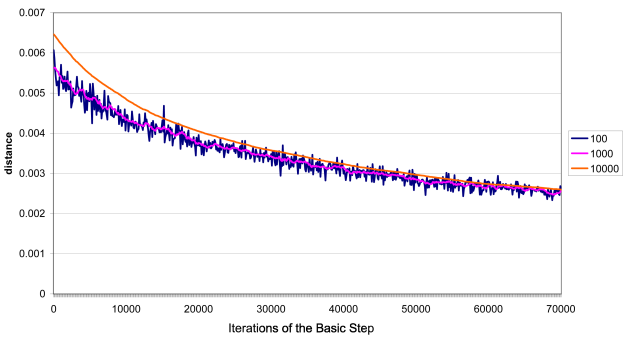

Figure 6. The moving average M_d of the distance between the sample and the best matching vertex. After few thousands vertices have been created, the learning process stabilizes.

thus, when the mesh is refined enough, only small changes occur in its global shape. This can be verified by the learning curve in Fig. 6, which depicts the moving average of the distance between the sample and the winner.

From the learning curve, it is clear that the neural mesh stabilizes after few thousand vertices. In our implementation, all the meshes of the ensemble have 10K vertices. The relatively small sizes of the meshes used for the ensemble reduces the computational cost, allowing the creation of ensembles with many meshes and thus increasing the likelihood of a reconstruction free of local minima. As it is the case with all neural network ensembles, the approach proposed here is ideally suited for parallelization, where each mesh of the ensemble can be constructed independently on a different machine. Nevertheless, due to the small cost of the reconstruction (about 3 minutes for each mesh), we did not attempt the parallelization.

Fig. 7 shows some neural mesh ensemble reconstructions. In Fig. 8, we show the eleven meshes of the ensemble for each of the Hand, Horse, Armadillo and David models. In the case of the Hand model, all of the eleven reconstructions contain local minima at some places but the ensemble of them generates a model without a local minimum. In the Horse model, the fifth reconstruction is free of local minima. In the Armadillo model, there is less conflict between the features, thus the second, fifth, seventh, ninth, and tenth reconstructions are free of local minima. The reconstruction of the David shows that the ensemble can capture the global topology of the reconstructed model.

Note that the Hand and the David models can also be reconstructed by a single neural mesh, as demonstrated in [11], provided that we optimize the parameters of the neural mesh.

Figure 7. The neural mesh ensemble reconstructions of the Hand, Horse, Armadillo and David models with 100K, 50K, 100K, 100K vertices, respectively.

5. Conclusion and Future Work

We experimented with the use of the ensemble technique to boost the performance of a probabilistic surface reconstruction algorithm. Our results show that we can reconstruct a surface free of local minima through averaging, even when all the individual reconstructions have one or more local minima.

In the future, we plan to experiment with different voxelization methods (e.g., [24]), and meshing methods (e.g., [15]), and analyze the influence on Stages 2 and 3 of the ensemble algorithm. We also plan to test the suitability of the boosting technique on deterministic surface reconstruction algorithms, after using random subsampling of the input to put them in a probabilistic setting.

Acknowledgements

We thank Christian Merkwirth for suggesting the neural mesh ensembles as a research direction. The Atlas and the David models are from the Digital Michelangelo Archive and the Drill and the Armadillo models are from the Stanford Repository. This work was supported in part by the Korea Ministry of Education through the Brain Korea 21 program and the Game Animation Research Center.

References

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi–based surface reconstruction algorithm. In *SIGGRAPH 98, Conference Proceedings*, pages 415–422, 1998.

[2] C. L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *SIGGRAPH 95, Conference Proceedings*, pages 109–118, 1995.

[3] J. Barhak and A. Fischer. Adaptive reconstruction of freeform objects with 3D SOM neural network grids. In *Pacific Graphics 01, Conference Proceedings*, pages 97–105, 2001.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH 01, Conference Proceedings*, pages 67–76, 2001.

[5] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[6] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. Technical Report ICSTR-93-026, International Computer Science Institute, Berkeley, 1993.

[7] M. Gross and F. Seibert. Visualization of multidimensional data sets using a neural network. *The Visual Computer*, 10(3):145–159, 1993.

[8] M. Hoffmann and L. Várady. Free-form modelling surfaces for scattered data by neural networks. *Journal for Geometry and Graphics*, 1:1–6, 1998.

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 92, Conference Proceedings*, pages 71–78, 1992.

[10] I. Ivrissimtzis, W.-K. Jeong, S. Lee, Y. Lee, and H.-P. Seidel. Neural meshes: Surface reconstruction with a learning algorithm. *available at http://www.mpi-sb.mpg.de/ ivrissim/neural.pdf*, 2004.

[11] I. Ivrissimtzis, W.-K. Jeong, and H.-P. Seidel. Neural meshes: Statistical learning methods in surface reconstruction. Technical Report MPI-I-2003-4-007, Max-Planck-Institut für Informatik, Saarbrücken, April 2003.

[12] I. Ivrissimtzis, W.-K. Jeong, and H.-P. Seidel. Using growing cell structures for surface reconstruction. In *Shape Modeling International 03, Conference Proceedings*, pages 78–86, 2003.

[13] W.-K. Jeong, I. Ivrissimtzis, and H.-P. Seidel. Neural meshes: Statistical learning based on normals. In *Pacific Graphics 03, Conference Proceedings*, pages 404–408, 2003.

[14] E.-A. Karabassi, G. Papaioannou, and T. Theoharis. A fast depth-buffer-based voxelization algorithm. *Journal of Graphics Tools*, 4(4):5–10, 1999.

[15] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature-sensitive surface extraction from volume data. In *SIGGRAPH 2001, Conference Proceedings*, pages 57–66, 2001.

[16] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[17] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH 96, Conference Proceedings*, pages 313–324, 1996.

[18] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algoritm. *Computer graphics*, 21(4):163–168, 1987.

[19] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, 2003.

[20] D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Advances in Neural Information Processing Systems*, volume 8, pages 535–541. The MIT Press, 1996.

[21] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.

[22] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[23] R. E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[24] M. Sramek and A. E. Kaufman. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):251–267, 1999.

[25] M. Teichmann and M. Capps. Surface reconstruction with anisotropic density–scaled alpha shapes. In *IEEE Visualization 98, Conference Proceedings*, pages 67–72, 1998.

[26] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:413–424, 1986.

[27] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH 87, Conference Proceedings*, pages 205–214, 1987.

[28] L. Várady, M. Hoffmann, and E. Kovács. Improved free-form modelling of scattered data by dynamic neural networks. *Journal for Geometry and Graphics*, 3:177–181, 1999.

[29] Y. Yu. Surface reconstruction from unorganized points using self-organizing neural networks. In *IEEE Visualization 99, Conference Proceedings*, pages 61–64, 1999.

Figure 8. Neural mesh ensembles for the Hand, Horse, Armadillo and David models.