Neural Meshes: Statistical Learning based on Normals

W.-K. Jeong*, I.P. Ivrissimtzis, and H.-P. Seidel
MPI Informatik, Stuhlsatzenhusweg 85, Saarbrücken, 66123, Germany
{jeong,ivrissim,hpseidel}@mpi-sb.mpg.de

Abstract

We present a method for the adaptive reconstruction of a surface directly from an unorganized point cloud. The algorithm is based on an incrementally expanding Neural Network and the statistical analysis of its Learning process. In particular, we make use of the simple observation that during the Learning process the normal of a vertex near a sharp edge or a high curvature area of the target space, statistically, will vary more than the normal of a vertex near a flat area. We use the information obtained from the study of these normal variations to steer the Learning process in an adaptive meshing application, producing meshes with more triangles near the high curvature areas. The same information is used in a feature detection application.

1 Introduction

In the last few years, Neural Network based algorithms, once used in large scale applications only, have been successfully employed to solve problems in the areas of Computer Graphics and Scientific Visualization. One example of such an application is the *Neural Meshes* proposed in [11, 10] as a surface reconstruction algorithm based on an incrementally expanding Neural Network known as Growing Cell Structure [7].

As it is the case with most of the reconstruction algorithms the output of the Neural Meshes is a triangle mesh capturing the geometry and the topology of the point cloud. The main motivation behind this paper is the observation that this kind of output does not contain the whole information one can extract from the Learning process. We propose a method for storing information which can not be retrieved from the final mesh, and using it to further enhance the quality of the reconstruction.

In particular, we study the variation of the normals of the Neural Mesh during the Learning process. The heuristic argument behind the proposed method is very simple. As the Neural Mesh converges to the target space, the behavior of the normal of a vertex reflects the geometry of the target space in the vicinity of that vertex. If a vertex is near a feature of the target space, like a sharp edge or generally an area of high curvature, it is more likely that its normal will vary highly during the Learning process, while the normal of a vertex near a flat part of the target space will stay more or less constant during the Learning process.

1.1 Related Work

The beginnings of the particular type of Neural Networks we use here go back to the classic [14], where the Self Organizing Maps (SOM's) were introduced. The Growing Cell Structures introduced in [7] are a SOM type of Neural Network which grow incrementally, node by node. That gives them the ability to represent the shape of the target space more efficiently and also helps to avoid the convergence to a local minimum, a common problem with the traditional SOM's.

In [11, 10], inspired from the Growing Cell Structures, we proposed a method for surface reconstruction based on an incrementally expanding Neural Network. In a typical response to a signal s, randomly sampled from the point cloud, we find the point v_w of the Neural Network nearest to s, the so called *winner*, and move it towards s. Then, the neighborhood of v_w is smoothed in the tangential direction. Other steps of the algorithm, i.e. vertex splits and half-edge collapses ensure that the Neural Network grows and adapts its connectivity to the target space. The basic geometry learning step and the Laplacian smoothing of the mesh, can also be seen as an external and an internal force acting on the mesh, [19, 20, 17].

Some other applications of Neural Networks to the surface reconstruction and closely related problems were proposed in [8, 21, 23, 3].

Other methods for surface reconstruction, based more on geometry than on statistics, include [9, 2, 15, 1, 18, 13, 4], to mention only the most Computer Graphics oriented. Finally, feature recognition is also a very active research area [6, 12, 22].

*Won-Ki Jeong is now with the University of Utah.

Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG'03)
0-7695-2028-6/03 $17.00 © 2003 IEEE

1.2 Discussion

Many of the advantages of the algorithm proposed here are common with the original Neural Meshes. For example, the performance of the algorithm is independent from the size of the data set, and the reconstruction of the surface is Bottom-Up, giving a first coarse approximation instantly.

The main difference from the original Neural Meshes is that now the reconstruction has an adaptive level of detail, reflecting the curvature behavior of the target space. Notice that usually we need two steps to obtain an adaptive mesh from a point cloud. First we have a non-adaptive reconstruction from a point cloud, which usually consists of uniformly distributed scanned data, followed by a remeshing step. In comparison, the one step approach of the Neural Meshes offers better control over the quality of the final mesh.

The main drawback of our method is its static topology. Notice, that the dynamic topology in [10] is based on the assumption that the density of the Neural Mesh reflects the density of the target space.

2 Neural Meshes

In this section we outline the basic Neural Meshes algorithm, introducing some extra generality with the concept of the utility counter. More details can be found [7, 11, 10].

The input of the algorithm is a target space Ω, which usually is a point cloud but it can also be an implicitly defined surface. As Ω is randomly sampled it is sometimes convenient to see it as a probability space $\mathscr{P}$ with underlying set Ω. The Neural Mesh $\mathscr{M}$ can also be seen in two different ways, that is, as a plain triangle mesh and as a Neural Network. The nodes of the Neural Network are the vertices and store geometric information, that is, their position in $\Re^3$, while the connections of the Network are the edges and store connectivity information only.

The algorithm starts with a simple initial Neural Mesh, usually a tetrahedron, which learns the target space $\mathscr{P}$ by the following algorithm:

1. **Basic Step**
   - Find a random sample s of the target space $\mathscr{P}$.
   - Find the winner, that is, the vertex v_w of $\mathscr{M}$ which is nearest to s, and update its position by
   $$v'_w = (1-\alpha_w)v_w + \alpha_w s \tag{1}$$
   where α_w is a constant.
   - Apply C_L iterations of Laplacian smoothing, in the tangential direction, on the 1-ring neighborhood of v_w, with parameter α_L, where C_L, α_L are constants.
2. **Vertex Split:** After a constant number C_{vs} of iterations of the Basic Step, split the vertex with the highest utility counter. The utility counter is a real function attached to each vertex measuring the importance of its role in the reconstruction of the surface.
3. **Half-Edge Collapse:** After $v \cdot C_{ec}$ iterations of the Basic Step, where v is the number of vertices of $\mathscr{M}$ at the last call of the Half-Edge Collapse Step and C_{ec} is a constant, remove with a half-edge collapse all the vertices that were inactive since the last call of the Half-Edge Collapse Step.
4. Stop when certain criteria are met.

With the *Basic Step* the Neural Mesh $\mathscr{M}$ learns the geometry of Ω. The *Vertex Split Step* expands $\mathscr{M}$ by duplicating the vertices with the most significant role in the representation of Ω. The way we measure the importance of a vertex, that is, the way we calculate the utility counter is crucial for the properties of the reconstruction. In [7, 11, 10], the utility counter was called *signal counter* and was a function reflecting how many times a vertex was the winner, with the most recent activity counting more. Given that the winner is the vertex nearest to s, the signal counter captures spatial information, and gives reconstructions with the nice property that the density of the vertices of $\mathscr{M}$ follows the distribution of $\mathscr{P}$. Nevertheless, we know that in many applications this is not satisfactory, and some other distribution, reflecting for example the curvature of Ω would be preferable. This can be done with the use of the normal based utility counter described in Section 3.

Finally, the *Half-Edge Collapse Step* removes the most inactive vertices, that is, the vertices with the least role in the representation of Ω. This improves the quality of the reconstruction and corrects mistakes caused by initially misplaced vertices or by a convergence to a local minimum.

3 The Normal Counter

In this Section we describe the *normal counter*, an utility counter storing information about the variation of the normals of the Neural Mesh during the Learning process. But, notice that the input data set Ω is still an unorganized point cloud without any normal information.

The variation of a normal, that is, the change of its direction is

$$\delta = 1 - \vec{n_w}\vec{n_w}' \tag{2}$$

where $\vec{n_w}, \vec{n_w}'$ are the normals of the winner, before and after one iteration of the Basic Step. Let M_δ be the mean average of these δ's for the last C_δ iterations, where C_δ is a constant integer. A typical value for the C_δ we use in the applications is 1000. We define the normalized δ as

$$n_\delta = \delta / M_\delta \tag{3}$$

The reason for this normalization is that as the Neural Mesh converges towards a smooth surface the δ tends to 0, and this will cause numerical instability in the calculation of the normal counter. Notice that because the M_δ does not change significantly in one iteration of the Basic Step, the normalization of the δ's keep their ratios intact, while on the other hand we achieve numerical stability as the n_δ's tend to 1.

After every iteration of the Basic Step we update the normal counter of the winner by

$$n_c' = n_c + n_\delta \tag{4}$$

while all the normal counters are multiplied by a constant α_{nc}. A typical value for this parameter is $\alpha_{nc} = 0.95$ and its effect is to gradually erase from the normal counter any obsolete information.

Following [7], when a vertex splits its normal counter splits proportionally to the area of the Voronoi regions of the two new vertices, as they are approximated by the area of the square $F_v = (l_v)^2$, with

$$l_v = \frac{1}{\text{valence}(v)} \sum_{v_i \in \text{1-ring}(v)} \|v_i - v\| \tag{5}$$

3.1 Applications of the normal counter

There are many applications one can think for the normal counters. For example, they can be used to

1. Steer the Learning process and achieve a reconstruction with adaptive level of detail.
2. Detect features on the final mesh in a more robust way than any analysis of the final mesh only can achieve.
3. As a termination criterion.

In the first application we use the normal counter as utility counter. The splitting of the vertices with the highest normal activity will lead to a reconstruction with adaptive level of detail. That is, the areas with higher curvature will be more densely represented by the Neural Mesh, while in the flat areas there will be less detail. This is a standard requirement in many Computer Graphics applications as the faithful representation of the normals is sometimes more important than that of spatial information. The wireframe views shown in Fig. 1 verify the heuristically justified expectation that the areas with high curvature are meshed more densely than the flat areas.

A second application is feature detection. A problem we face here is that the normal counter described above is designed to find the most active vertices. Concerning the less active vertices, even after the normalization of the δ's, it runs into numerical instabilities because of the global multiplication by α_{nc}. For that reason we use a variation of the normal counter, updating at every iteration only the normal counter of the winner by

$$n_c' = \alpha_{fd} n_c + (1 - \alpha_{fd}) n_\delta \tag{6}$$

where α_{fd} is a constant. Then, using the simplest possible criterion, we designate a vertex as feature if its normal counter exceeds a threshold

$$C_{nc} \cdot Mean(n_c) \tag{7}$$

where C_{nc} is a constant and $Mean(n_c)$ is the mean average of the normal counters.

This way we can detect creases with arbitrarily high degree of confidence. Indeed, if there is a tangent plane E at a point $P \in \Omega$, and $\mathbf{n}$ is the normal of E, then, as the algorithm converges, the normals of all the vertices of $\mathscr{M}$ in the vicinity of P tend to $\mathbf{n}$ and the normal counters tend to 0. On the other hand, if P is on a crease there are two different planes E_l, E_r, tangent on the left and on the right of the crease, respectively. Then, during the Learning process, the normals of $\mathscr{M}$ in the vicinity of P will vary between the normals of E_l, E_r and their normal counters will not tend to 0. Therefore, for any required degree of confidence the algorithm will reach a stage where eventually it will confirm P as a feature point.

The situation is more complicated with features corresponding to smooth parts of the surface with high curvature. Near the areas of high curvature the normal counters are larger than those near flat areas, but the ratios do not increase as the algorithm progresses and the area of Ω covered by a vertex of $\mathscr{M}$ shrinks. That means that we cannot detect a feature of this kind with arbitrarily high degree of confidence. However, we think that this result is compatible with the natural ambiguity over what constitutes a feature and what not.

Fig. 2 at the end of the paper shows colormaps for the normal counters of reconstructed surfaces. Notice that using only the final mesh we can not retrieve the entire information the normal counters contain. In fact, the normal counters capture information not only about the final mesh but all the intermediate meshes as well, giving this way an innovative solution to the problem of answering very local queries with a high degree of confidence. That is, we do not use only one reconstruction, but analyze statistically many reconstructions.

Finally, the normal counters can be used in a criterion for the termination of the algorithm. For example, if

$$Mean(n_c) < \alpha_t \tag{8}$$

where α_t is a constant, then the algorithm terminates. Such a criterion gives us very good control over the visual quality of the final reconstruction.

4 Examples and Results

To run our experiments we first need to specify a set of parameters for the algorithm. A method for parameter tuning can be found in [10]. Here we use the set of parameters which was also used in [7, 11] with reasonably good results:

$$C_L = 5 \quad C_{vs} = 100 \quad C_{ec} = 10$$
$$\alpha_w = 0.06 \quad \alpha_L = 0.05 \quad \alpha_{nc} = 0.95 \quad \alpha_{fd} = 0.95 \quad (9)$$

For point clouds obtained from a triangle mesh by stripping out the connectivity information we can measure the quality of the reconstruction using the Metro-tool [5]. Table 1 shows the results of such an experiment on the Brain model. The numbers are the percentage mean square Bounding Box error of the reconstruction compared to the initial model. The adaptive method (ad) gave better results than the non-adaptive (nad) in almost all the experiments. The only exceptions appear in cases where the Neural Mesh had stacked at local minima.

At the end of the paper we show images of some Neural Meshes reconstructions. They images were rendered with Yataka Ohtake's *MeshEditor* software package.

Table 1. Error measurement results.

#v	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5	Aver.
500 ad	.732	.724	.732	.729	3.75	1.33
500 nad	.756	.753	.745	.744	.751	.749
2k ad	.427	.424	1.96	.431	.428	.734
2k nad	.433	.440	.441	.437	.439	.438
10k ad	.172	.173	.170	.172	.171	.171
10k nad	.197	.199	.197	.197	.199	.197

5 Future Work

One direction for our future work is towards an adaptation of the algorithm to preserve the features of the reconstructed surface. One can start with an existing method, like [16], and modify it to exploit the extra information gained during the Learning process. A second direction is towards a dynamic reconstruction of the topology, in the spirit of [10]. Finally, we plan to further experiment with different utility counters, using local energy functionals defined on the Neural Mesh.

References

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

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

[3] J. Barhak and A. Fischer. Adaptive reconstruction of freeform objects with 3D SOM neural network grids. In *Proc. Pacific Graphics 01*, 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 *Proc. SIGGRAPH 01*, pages 67–76, 2001.

[5] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[6] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. SIGGRAPH 99*, pages 317–324, 1999.

[7] 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.

[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 *Proc. SIGGRAPH 92*, pages 71–78, 1992.

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

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

[12] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proc. SIGGRAPH 01*, pages 57–66, 2001.

[13] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130, 1999.

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

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

[16] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Mesh smoothing by adaptive and anisotropic gaussian filter. In *Proc. Vision Modeling and Visualization 02*, pages 203–210, 2002.

[17] 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.

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

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

[20] D. Terzopoulos, J. Platt, A. Barr, and Kurt Fleischer. Elastically deformable models. In *Proc. SIGGRAPH 87*, pages 205–214, 1987.

[21] 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.

[22] J. Vorsatz, C. Rössl, L. Kobbelt, and H.-P. Seidel. Feature sensitive remeshing. In *Proc. Eurographics 01*, pages 393–401, 2001.

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

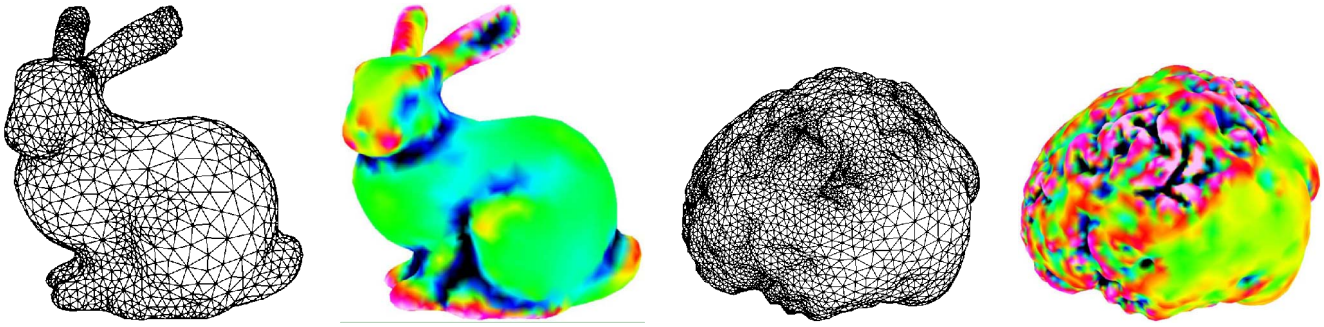

Figure 1. The comparison between the wireframe views and mean curvature colormaps shows that the high curvature areas are reconstructed in higher detail than the flat ones.

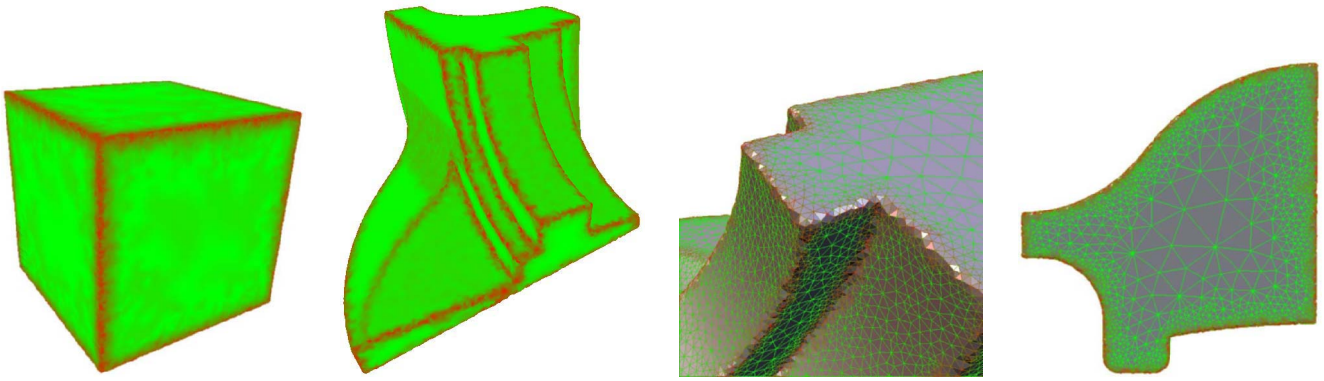

Figure 2. Colormaps of the normal counters.

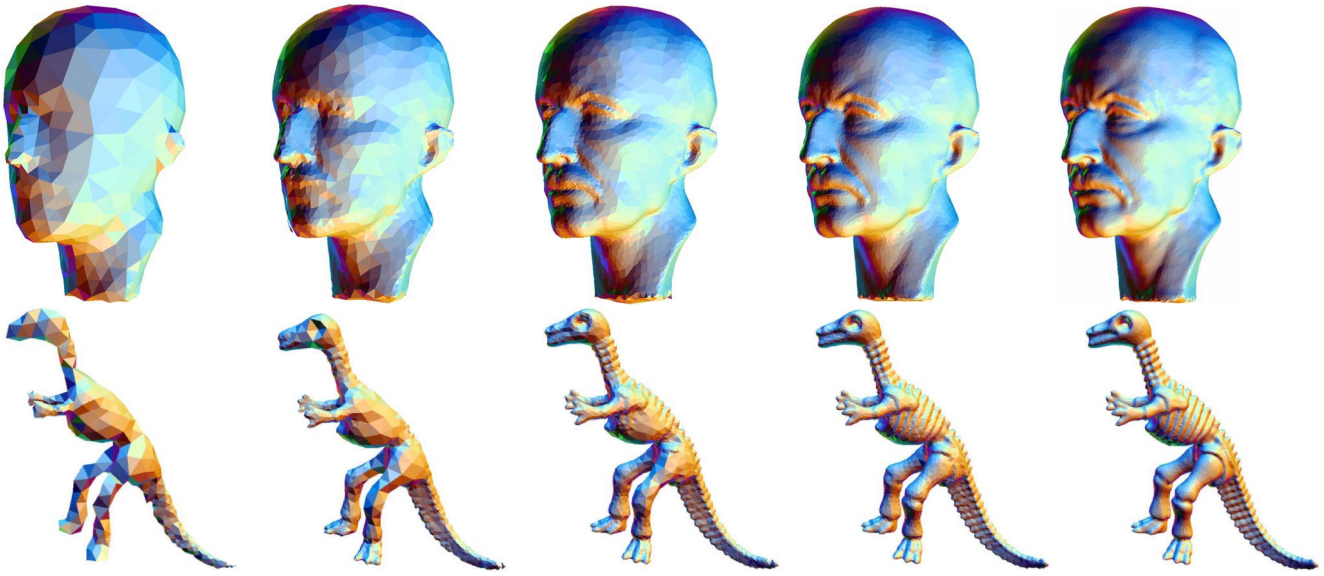

Figure 3. The Max-Planck and the Dino models, reconstructed from 100k, 225k points, respectively, at resolution [500,2k,5k,10k,20k].