Neural Gas for Surface Reconstruction

Markus Melato Barbara Hammer Kai Hormann
Department of Informatics, Clausthal University of Technology

Abstract

In this paper we present an adaptation of Neural Gas (NG) for reconstructing 3D-surfaces from point clouds. NG is based on online adaptation according to given data points and constitutes a neural clustering algorithm that is robust with respect to noisy data and efficient, because the runtime depends solely on the accuracy of the surface reconstruction and is independent of the size of the input data. NG is a mathematically well founded method with guaranteed convergence that provably induces a valid Delaunay triangulation of a given surface of arbitrary genus provided the sampling of the surface is sufficiently dense. To apply NG to surface reconstruction, we extend the basic model by techniques that reconstruct surface triangles from the NG neighbourhood graph and repair possible topological defects of the mesh. Moreover, we extend NG by a growing strategy which introduces a dynamic adaptation to the problem complexity such that a fast initial adaptation to the general global shape of the target model is achieved after which the finer details can be extracted.

1 Introduction

The task of reconstructing a surface from an unstructured point cloud of data samples still is a subject of recent research in the field of computer graphics [14, 16]. Although excellent results can be achieved with existing algorithms in certain cases, quite a few challenges remain.

The availability of high resolution scans results in very large data sets with millions of sample points. Many algorithms cannot handle such a quantity of input data at all and they rely heavily on pre-processing and thinning the samples. Other methods use costly post-processing like remeshing and simplification to generate a feasible triangle mesh.

The ability to cope with noisy input data is another topic of great importance, since automatically generated data (e.g. from a laser scan) are subject to noise and, with higher resolution, even small disturbances become evident in the object's surface. This subject is often likewise dealt with by applying pre- or post-processing algorithms for smoothing the surface.

Objects of higher genus (i.e. handles) are an additional obstacle for many existing methods. Often this is circumvented by special control features for attaching or merging some parts of the reconstruction in order to obtain a result of genus greater than zero, but the algorithms usually rely on heuristics and convergence to the correct topology is not proven.

In this work, we use the Neural Gas (NG) algorithm [12] for the problem of surface reconstruction. This approach is in a line with the neural approach introduced by Ivrissimtzis et al. [6, 8] that is based on Growing Cell Structures and gives remarkable surface reconstruction results. Due to the inherent error tolerance and online adaptation of neural systems, their method can also cope with large and noisy data sets, but it does not possess any mathematical guarantees that it converges to a topologically correct shape. Instead, the NG algorithm that we use adapts its neighbourhood graph automatically to the topology of the given data manifold, such that very strong theoretical guarantees hold.

We adapt NG to the reconstruction of 3D-surfaces by using the point cloud as training input. This approach makes the algorithm time independent from the size of the input data and capable of dealing with very large and noisy data sets, which are implicitly reduced and smoothed by the algorithm. A major strength of this method is the inherent ability to reconstruct objects of higher genus without the need to perform special operations in order to create handles. The ability of the NG algorithm to form a vector quantization of the data space and to construct a complete Delaunay triangulation can both be proven mathematically [12, 13].

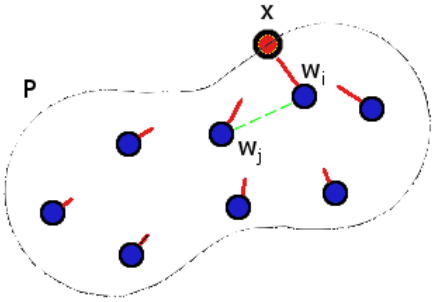


Figure 1: Adaptation of neurons $\mathbf{w}$ to a data sample $\mathbf{x}$ taken from point cloud P.

Our adapted Neural Gas algorithm can be summarized as follows:

1. Start with an initial number of neurons (vertices), distributed randomly in the bounding box of the point cloud.
2. Pick randomly one data sample from the input data set.
3. Move neurons towards the selected sample as shown in Figure 1.
4. Establish a connection (edge) between the closest and the second-closest neuron.
5. Remove connections which have not been refreshed for some time.
6. Repeat steps 2 to 5 until termination condition is met.
7. Generate faces from the edges of the constructed neighbourhood-graph.

2 Related Work

Popular approaches for surface reconstruction using interpolation methods are the *PowerCrust* [2] and the *Cocone* [1] algorithms as well as improved variants like *TightCocone*. These methods originate from Computational Geometry and treat every point of the input data as a vertex for the reconstruction. For a more detailed introduction to reconstruction algorithms using Computational Geometry we refer to [3]. Although these algorithms achieve good results for moderately sized and low-noise input data, they have severe problems when dealing with very large point clouds and noisy data.

An approach to deal with large and noisy point clouds was introduced by Ivrissimtzis, Jeong, and Seidel [6] using a neural approach, Growing Cell Structures, to reconstruct a watertight surface. Starting with a simple tetrahedron, consecutively adding new vertices, and moving the vertices towards the input data points, the algorithm achieves a consistent topology which approximates the data surface. One drawback of the original Growing Cell Structures, however, is their inability to reconstruct an object of higher genus. Moreover, the dynamics and convergence properties of the algorithm have not been investigated in a rigourous mathematical way. In subsequent papers [7, 8], the authors present an extension of Growing Cell Structures, Neural Meshes, which can approximate surfaces of arbitrary genus by implementing a special mending to detect merging or overlapping parts of the reconstruction. Further improvements add specific control based on the curvature of the surface and yield remarkable results.

Recent approaches to surface reconstruction include methods using Advancing Fronts for tracking the surface [14, 15]. Starting with a seed triangle and growing new ones at the boundary edges, this approach wraps the point cloud with a complete mesh, using a curvature prediction to control the density. The Competing Fronts algorithm as proposed in [16] can be considered as a combination of

Neural Meshes and Advancing Fronts. It uses an initial spherical object inside the data set and expands it towards the point cloud. The expansion is resumed for every part of the growing mesh which is not yet in contact with the data samples until the whole model is filled. Similar to Neural Meshes, this method needs special stitching operations to construct models of higher genus. Additionally, the algorithm uses time-consuming pre-processing of the point cloud in order to construct a guidance field, as well as a final post-processing using MLS projections.

3 Neural Algorithms

Neural Gas is based on the same basic principle as the approaches mentioned above regarding the ability of neural networks to adapt to different environments. A neural network is composed of neurons which are adjusted to given input data by means of a learning algorithms. Prototype based methods constitute a popular family of neural networks for unsupervised learning, as given in our setting. Neurons represent prototypical positions of the data space and they are trained according to the given data using Hebbian learning, i.e., neurons are moved towards the data points.

Neural algorithms are approximative methods, i.e., they inherently compute a statistical average over the noise contained in the data space. Thus, neurons are located at typical positions that represent the data manifold, and are placed in between the input samples. This approach has a number of consequences: First, the algorithm performs an implicit data reduction of the input data. The number of generated vertices, hence the accuracy of the resulting model, can be controlled explicitly by the parameters of the algorithm. Second, this approach has inherent smoothing abilities, which leads to a very high robustness of the generated shape with respect to noisy input data. Third, the runtime of the algorithm depends on the complexity of the manifold and the approximation quality rather than the number of given training data: in other words, the calculation time depends only on the number of neurons, which in turn corresponds to the number of vertices in the reconstructed triangle mesh, but not on the number of points in the given point cloud. Therefore, this approach is well suited for generating a very fast approximation of the sample data which can be further improved by using different parameter sets or post-processing methods for final refinements.

There exists a variety of unsupervised learning methods that are suitable for this purpose, including the Self Organizing Map (SOM) and Growing Cell Structures [9]. In the following, the details of the adaptation of NG to surface reconstruction are presented.

3.1 Neural Gas

The Neural Gas algorithm is a mathematically well founded and very robust vector quantization method. First proposed by Martinetz and Schulten [12], this model is an ideal quantizer for approximating and reducing large data sets with a fixed number of neurons.

Generating Vertices. Neural Gas can be derived as a stochastic gradient descent method of a cost function which enlarges the standard quantization error by neighbourhood cooperation. In general, we assume the data to stem from a vector space $\mathbb{R}^d$ (with $d = 3$ for surface reconstruction) and that a number N of neurons $\mathbf{w}_i \in \mathbb{R}^d$ has to be adapted according to the given data. A NG network represents the data by means of a winner-takes-all function: An input data $\mathbf{x} \in \mathbb{R}^d$ is mapped to the neuron that is closest in the Euclidean norm. Assuming a continuous data manifold according to the distribution P, NG optimizes the cost function

$$E \sim \sum_i \int e^{-\frac{\mathrm{rk}(\mathbf{w}_i, \mathbf{x})}{\sigma^2}} \cdot \|\mathbf{w}_i - \mathbf{x}\|^2 P(\mathbf{x})\, d\mathbf{x} \tag{1}$$

where

$$\mathrm{rk}(\mathbf{w}_i, \mathbf{x}) = \#\left\{\mathbf{w}_j : \|\mathbf{w}_j - \mathbf{x}\|^2 \leq \|\mathbf{w}_i - \mathbf{x}\|^2\right\}$$

denotes the rank of neuron $\mathbf{w}_i$ measured according to the distance from $\mathbf{x}$ and $\sigma > 0$ controls the degree of neighbourhood cooperation. Obviously, for $\sigma \to 0$ this results in the standard quantization, i.e., the

neurons are arranged within the data cloud in such a way that the averaged distance of data points from their closest neuron is minimized. For $\sigma > 0$, the vicinity is also taken into account, thus giving a very robust and smooth quantization method.

NG is derived from the cost function by means of a stochastic gradient descent. That means, neurons are initialized randomly. A random vector $\mathbf{x}$ is drawn according to P and all neurons are moved towards $\mathbf{x}$ according to their rank

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \cdot \mathrm{e}^{-\frac{\mathrm{rk}(\mathbf{w}_i, \mathbf{x})}{\sigma^2}} (\mathbf{x} - \mathbf{w}_i). \tag{2}$$

The value $\eta > 0$ is the learning rate which is decreased after every training step to ensure convergence (e.g., convergence can be guaranteed for learning rates which are not summable but their square is summable [10]), and σ is the adaptation radius which is also decreased after every training step to ensure convergence towards the standard quantization error. The fact that the cost function (1) provides a valid cost function for this adaptation rule has been proved in [11].

Constructing Edges. Unlike SOM or Growing Cell Structures, NG uses a neighbourhood structure which is determined according to the given data manifold. This fact can be used to construct an optimal topology of the neurons during training as demonstrated in [13]. Assume that two neurons are neighbours if and only if they possess neighboured receptive fields within the data manifold. This definition induces a valid Delaunay triangulation of the data manifold, as shown in [13]. Martinetz also proposes a competitive Hebbian learning rule to learn this topology: Initially, the connection strengths C_{ij} between all neurons $\mathbf{w}_i$ and $\mathbf{w}_j$ are set to 0. Each training step then iteratively selects data points $\mathbf{x}$ from the input data and finds the two neurons $\mathbf{w}_i$ and $\mathbf{w}_j$ that are closest to $\mathbf{x}$. Then, if $C_{ij} = 0$, the connection strength C_{ij} is set to 1, i.e. these two neurons are connected.

It was shown in [13] that competitive Hebbian learning provides a valid Delaunay triangulation of the data manifold if training data are dense on the manifold, i.e., for every point $\mathbf{x}$ in the manifold, the triangle spanned by $\mathbf{x}$ and its first and second winner completely lies in the data manifold. Obviously, this property depends on the sampling rate of the data manifold. As expected, problems can occur for regions with high curvature, sparse sampling, and sparse neurons within these regions.

Competitive Hebbian learning of the connectivity after training can be substituted by an online competitive Hebbian learning during training: Initially, all connection strengths C_{ij} of neurons $\mathbf{w}_i$ and $\mathbf{w}_j$ are set to 0. In each training step, this value is increased by 1 if and only if $\mathbf{w}_i$ and $\mathbf{w}_j$ are the closest two neurons with respect to the considered data point $\mathbf{x}$. Moreover, every value C_{ij} is decreased by a small factor in every step to allow forgetting of initial false connections.

Detecting Faces. After training, NG provides a set of vertices and connections that constitute a Delaunay triangulation of the data manifold, but no explicit surface structure is given. To construct a final triangulation of the object surface, we propose a post-processing algorithm: Starting with any given edge, all pairs of adjacent edges are inspected in a breadth-first search manner until a common neighbour is found. Obviously, for a closed surface, these three edges form a triangle, and the boundary edges are added to a pool of boundary edges. The next starting edge is selected from this pool until a closed surface has been constructed.

In practical applications, the NG triangulation can have defects due to the fact that the competitive Hebb rule did not yet converge or data are not dense on the manifold. Hence, the mesh is usually composed of a multitude of triangles plus a few larger polygons. These polygons result from sparsely sampled regions in the data set or from a pre-convergent end of the algorithm. It is possible to detect and reconstruct holes in the object in a correct manner and to triangulate these polygons in order to obtain a closed surface as shown in Figure 2. This is done by expanding the depth of the search tree from solely triangles detection to polygons of higher order up to a given maximum.

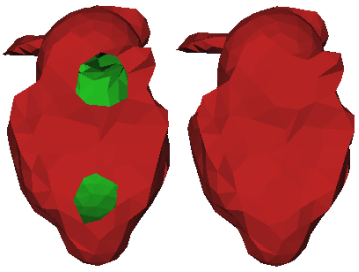
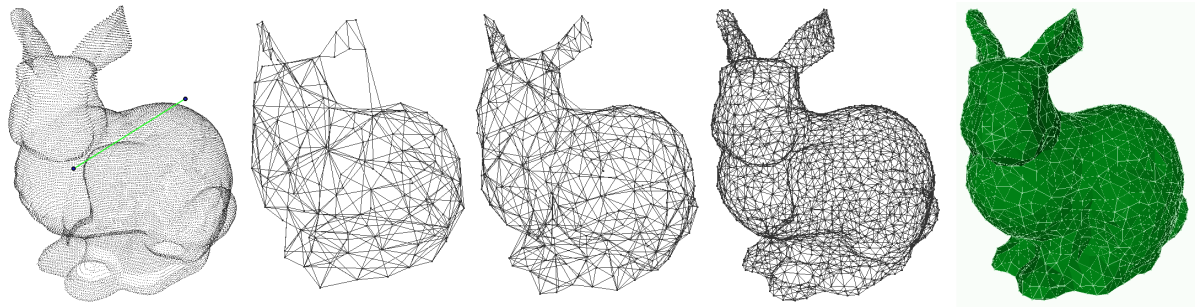


Figure 2: A coarse reconstruction of the Stanford Bunny before (left) and after (right) closing holes in the surface.



(a) (b) (c) (d) (e)

Figure 3: Reconstruction sequence of the Stanford Bunny using GNG. (a) Initial setup: Data set and two connected Neurons, (b) 127 vertices, (c) 284 vertices, (d) final stage: 1415 vertices, (e) surface added.

Further improvements. A major drawback of the NG approach is the limitation to an initial quantity of neurons. A small number of neurons leads to a very fast but coarse reconstruction of the surface without significant details whereas a large number of neurons hampers a fast convergence and may result in disconnected vertices in the interior part of the object. This can be prevented by two simple extensions.

First, instead of adapting all neurons, only a fixed number of the n nearest units are changed in a single training step. This approximates the behaviour of the algorithm in later training steps when the neighbourhood range σ is small. Values of n between 10 and 50 have proven a good choice in our experiments. In practice this reduces the effort of performing one learning iteration which is dominated by sorting N neurons ($O(N \log N)$) to a complexity of $O(n \log N)$ with fixed n under the assumption that an efficient spacial structure is used. The behaviour of the algorithm is only affected by this simplification in the very beginning, when the exponentially decreasing adaptation radius σ is still large and may lead to disconnected vertices.

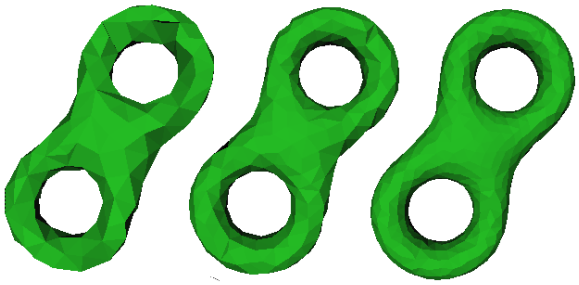
This effect is compensated nevertheless by the second improvement that is introduced to speed up the initial phase of exploration while maintaining a large quantity of vertices in the final reconstruction. The algorithm starts with a small number of neurons which adapt towards the general shape of the model in a few training steps. After a certain number of iterations, each neuron is duplicated. Thereby, the newly created vertices are placed on the surface, thus, they are connected. The subsequent iterations separate the duplicated neurons if the learning rate η in Equation (2) is chosen appropriately. Note that this procedure can be repeated until the desired granularity of the surface is reached.

3.2 Growing Neural Gas

For comparison, we also adapt the Growing Neural Gas (GNG) algorithm as proposed by Fritzke [5]. This method introduces a growing strategy to continuously increase the number of neurons during the adaptation process by creating new units in those regions with the largest deficit. In order to detect those regions an error value is calculated for every neuron depending on the mean distance to the nearest data samples. New units are created to split the edge connecting that neuron with the highest error value and the one with the highest error value in its neighbourhood. Apart from the growing strategy GNG differs from NG in the following ways:

The initial setup for the Growing Neural Gas is a very small network of two connected units. In every adaptation step the connectivity of the neurons is considered in order to adapt only those units which are connected to the best fitting neuron to the data sample. Furthermore the adaptation parameters are kept constant in order to enable the algorithm to process as long as desired without premature convergence. The adaptation of a neuron is calculated as follows:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \cdot (\mathbf{x} - \mathbf{w}_i) \tag{3}$$



(a) (b) (c)

Figure 4: Development of the reconstruction of the Eight model using GNG at different stages of the algorithm: (a) 200 vertices, (b) 400 vertices, (c) 800 vertices.

(a) (b) (c)

Figure 5: The Eight model reconstructed using Neural Gas from differently sized point clouds: (a) 12,500 samples, (b) 50,000 samples, (c) 200,000 samples.

with a learning rate $\eta = \eta_1$ for the neuron closest to the data sample, $\eta = \eta_2 < \eta_1$ for its directly connected neighbours and $\eta = 0$ in all other cases. The Growing Neural Gas algorithm has the advantage of adapting very quickly and dynamically to the complexity of the input data. The quality of the reconstruction can be further improved by additional iterations of the algorithm. The course of a reconstruction using GNG is shown in Figure 3. However, GNG does no longer follow a stochastic gradient descent of a cost function and the convergence of the derived topology towards a valid Delaunay triangulation has not been proved but it worked well in our examples.

Further improvements. In the original GNG algorithm the number of adaptation steps between the creation of new neurons is kept constant. With this approach it is likely that in later iterations a newly created neuron is not chosen as the nearest unit to a data sample until the next neuron is created. This leads to an increasing number of vertices with only two neighbours and thus contradicts the creation of a triangle mesh. Therefore we use a number of data samples depending on the actual quantity of neurons before a new unit is created.

4 Experiments and Discussion

We applied the presented algorithms to a number of point clouds with varying size and complexity. Table 1 summarizes the properties of our examples used in this section. All figures printed in red were reconstructed using the Neural Gas algorithm, whereas the green figures originate from the Growing Neural Gas algorithm.

As expected, both algorithms have no difficulties reconstructing models with higher genus. This inherent ability is demonstrated in Figures 4–6. Different stages of the reconstruction by GNG are shown in Figure 4. Already at the beginning of the reconstruction, with just a few vertices, the genus of the model is modelled correctly. After more iterations of the algorithm the accuracy of the reconstruction is further improved. The calculation time to generate the meshes is 0.2s for 200 vertices, 1s for 400 vertices and 6s for 800 vertices.

The same model is reconstructed using the Neural Gas algorithm as shown in Figure 5. This series is focused on the dependency of differently sized point-clouds. The algorithm was applied to construct a mesh with 800 vertices. The quality of the resulting mesh is independent from the size of the point cloud, since the same number of vertices is generated in each case. The time needed to generate the final reconstruction is independent from the quantity of input samples as well. Each mesh was constructed in less than 5 seconds.

(a) (b)

Figure 6: A model with genus 3, reconstructed using Neural Gas from differently sized point clouds: (a) 26,500 samples, (b) 425,000 samples.

(a) (b)

Figure 7: Reconstruction of the Bear model using (a) Neural Gas and (b) Growing Neural Gas.

Figure 6 shows an object of genus three, which was reconstructed from data sets consisting of 26,500 and 425,000 points, respectively. The reconstructions consist of 1000 neurons each. Again, the time needed to construct the meshes was independent from the size of the point cloud. The reconstruction took 6 seconds for each model.

To prove the ability of the neural algorithms to deal with very large data sets, we also applied them to a point cloud with 2 million samples (Figure 7). The data was obtained using a 3D-scanner and we applied the algorithms to the raw scan data. Both the NG and the GNG algorithm constructed a mesh with 1000 vertices and similar quality. This task took about 9 seconds for the Neural Gas and about 12 seconds for the Growing Neural Gas algorithm to complete. The increase in time needed to generate this mesh compared to the previous results originates from the requirement for a higher accuracy of the reconstruction. For the Neural Gas algorithm, the number of affected units had to be set to 20, whereas for the smoother shapes in the previous examples a value of 10 was sufficient.

To analyse the robustness of the algorithms with respect to noisy input, every data point of the Eight model were perturbed with randomly generated and evenly distributed noise. The intensity of the noise was measured relative to the diagonal of the models bounding box. In our experiments, Neural Gas performed slightly better than Growing Neural Gas on this task. As is shown in Figure 8 both methods dealt well with noise levels up to 4% of the model size. NG was also able to handle a noise of 6%, the reconstruction generated by GNG already displays a few flaws, similar to the NG model for 8% noise. Reaching a noise level of 10%, neither method was able to reconstruct the surface correctly any more.

Samples for more complex models are given in Figures 9 and 10. The Bunny was reconstructed with NG until 1,600 vertices were generated, which took about 34 seconds. The reconstruction of the Dragon was done by GNG. The generation of 1,800 vertices was finished after 55 seconds. Note the correct detection of the handle at the dragon's tail.

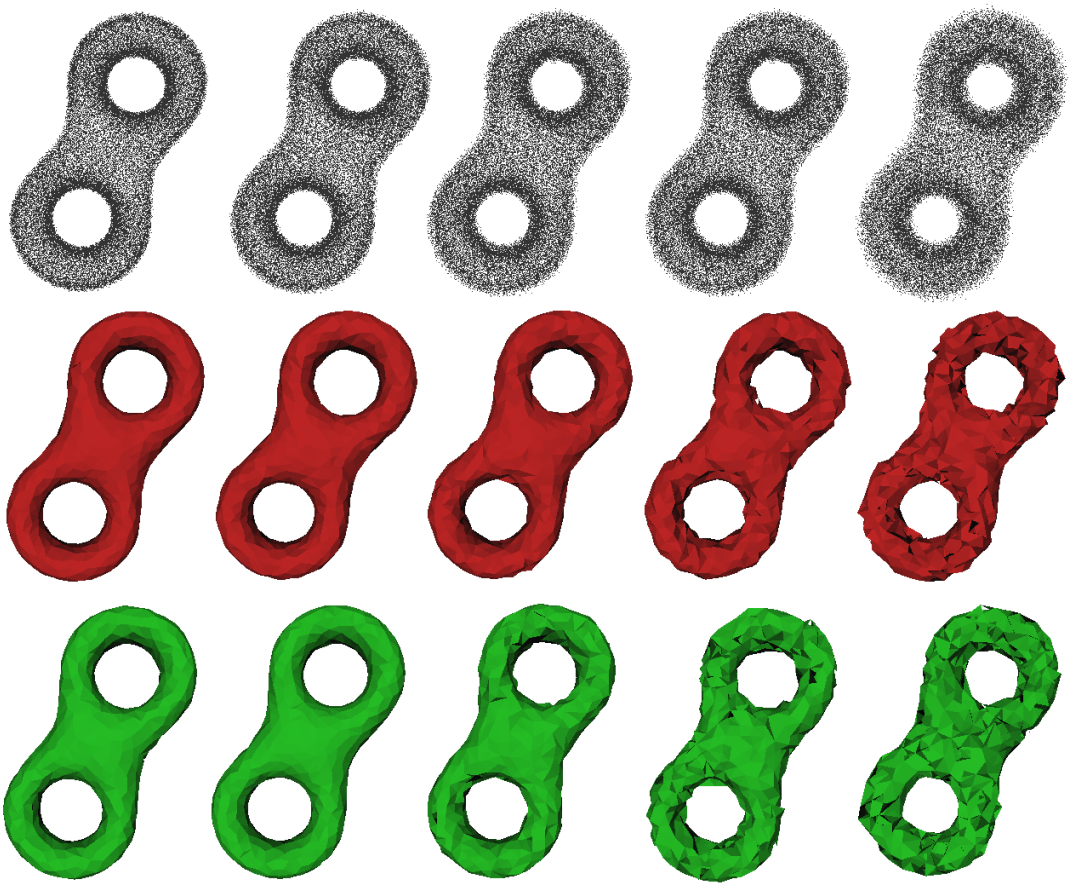


Figure 8: Comparison of different noise levels. From left to right: 2%, 4%, 6%, 8%, 10%.

Model	Figure	Method	Input vertices	Mesh vertices	Time
Bunny	3 (b)	GNG	36,000	127	< 0.1s
Bunny	3 (c)	GNG	36,000	284	0.2s
Bunny	3 (d,e)	GNG	36,000	1,400	34s
Eight	4 (a)	GNG	12,500	200	0.2s
Eight	4 (b)	GNG	12,500	400	1.1s
Eight	4 (c)	GNG	12,500	800	5.6s
Eight	5 (a)	NG	12,500	800	4.9s
Eight	5 (b)	NG	50,000	800	4.8s
Eight	5 (c)	NG	200,000	800	4.9s
Genus3	6 (a)	NG	26,500	1000	6.0s
Genus3	6 (b)	NG	425,000	1000	6.0s
Bear	7 (a)	NG	2,000,000	1000	12.2s
Bear	7 (b)	GNG	2,000,000	1000	9.1s
Eight	8	NG	50,000	800	4.8s
Eight	8	GNG	50,000	800	5.6s
Bunny	9	NG	36,000	1,600	34s
Dragon	10	GNG	437,000	1,800	55s

Table 1: Reconstruction properties of our examples.

Figure 9: Reconstruction of the Stanford Bunny using Neural Gas.

Figure 10: Reconstruction of the Dragon model using Growing Neural Gas.

5 Conclusion

In this paper we presented two neural algorithms for 3D-surface reconstruction. As has been shown, both work independently from the size of the point cloud and have high robustness with respect to noisy input data. Furthermore the inherent abilities to reconstruct shapes of higher genus has been demonstrated.

The proposed methods are suitable for a very fast generation of the rough shape and topology of the model. Using a small number of vertices, even huge data sets can be approximated in a few seconds.

Future Work. Although there are theoretical proofs for the convergence of the Neural Gas algorithm towards the model surface, it still takes too long for practical purposes to reconstruct the finer details. We are currently working on improvements for the later stages of the algorithms in order to speed up final convergence towards smaller features using revised growing strategies for both NG and GNG. Further improvements include final refinement operations for generating meshes of higher resolution as well as a projection of the vertices onto the data manifold, e.g. using an MLS projection.

References

[1] N. Amenta, S. Choi, T. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *16th ACM Symposium on Computational Geometry*, 213–222, 2000.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Sixth ACM Symposium on Solid Modeling and Applications*, 249–260, 2001.

[3] T. K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis.* Cambridge University Press, 2006.

[4] B. Fritzke. Growing cell structures – a self-organizing network in k dimensions. *Artificial Neural Networks*, 2(2):1051–1056, 1992.

[5] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, 625–632, 1995.

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

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

[8] I. Ivrissimtzis, Y. Lee, S. Lee, W.-K. Jeong, and H.-P. Seidel. Neural Mesh Ensembles. In *3D Data Processing, Visualization, and Transmission*, 308–315, 2004.

[9] T. Kohonen. Self-organizing maps. Springer, 2001.

[10] H. Kushner and D. Clark. Stochastic approximation methods for constrained and unconstrained systems. Springer, 1978.

[11] T. Martinetz, S. Berkovich, and K. Schulten. "Neural Gas" Network for Vector Quantization and its Application to Time-Series Prediction. In *IEEE Transactions on Neural Networks*, 558–569, 1993.

[12] T. Martinetz and K. Schulten. A "Neural Gas" Network Learns Topologies. In *Artificial Neural Networks*, 397–402, 1991.

[13] T. Martinetz and K. Schulten. Topology Representing Networks. In *Neural Networks*, 507–522, 1994.

[14] C. E. Scheidegger, S. Fleishman, and C. T. Silva. Triangulating Point Set Surfaces with Bounded Error. In *Eurographics Symposium on Geometry Processing*, 63–72, 2005.

[15] J. Schreiner, C. E. Scheidegger, S. Fleishman, and C. T. Silva. Direct (Re)Meshing for Efficient Surface Processing. In *Eurographics '06*, 527–536, 2006.

[16] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, and D. Cohen-Or. Competing Fronts for Coarse-to-Fine Surface Reconstruction. In *Eurographics '06*, 389–398, 2006.