Surface Reconstruction with Growing Neural Gas

Christian A. Mueller, Nico Hochgeschwender, and Paul G. Ploeger

Abstract— **Surface reconstruction is a crucial step to convert unstructured point clouds to a compact representation. In this paper we introduce a modified Growing Neural Gas (GNG) algorithm for surface reconstruction of free-formed objects found in domestic environments. In our experiments we show that the algorithm generates consistent surfaces and is able to cope with noise in the point clouds. Therefore, the proposed algorithm is an attractive alternative to standard triangulation methods and also applicable as a basic processing step in a perception pipeline (e.g. object categorization) for service robots.**

I. Introduction

Service robots performing household tasks (e.g. pick and place) in unconstrained and domestic settings must be able to robustly perceive object information even in the presence of cluttered and dense environments. Thereby, independently from the concrete application, surface reconstruction is an important processing step in every perception system found in service robotics. However, surface reconstruction from real-world data is challenging: the observed point clouds from objects are partial, unorganized, and free-formed. For instance, they are not necessarily based on principle shapes as cylinders, spheres, boxes, or cones. Furthermore, with noisy sensors it is difficult to generate a consistent object representation because points are missing or displaced. Naïve triangulation approaches (e.g. Delaunay based) for surface reconstruction tend to generate surfaces which are potentially overfitted with noise. Therefore, we propose in this paper an alternative approach for surface reconstruction based on Growing Neural Gas (GNG) introduced by Fritzke [1]. The modified algorithm is able to cope with free-formed, noisy, and partial point clouds from noisy RGB-D cameras like the Microsoft Kinect©.

II. Related Work

Since several decades machine learning methods based on Artificial Neural Networks have been applied to different problems – ranging from data mining to computer vision. In this paper we interpret surface reconstruction as an unsupervised learning problem. The input vector is a point of a point cloud from an object and the expected output is the surface topology of the object. A similar learning problem has been proposed by Fritzke[1] where an unsupervised Growing Neural Gas (GNG) has been introduced to incrementally learn the topological structure of input vectors. The GNG preserves the topology of the input vectors and shows a contrary effect to noisy data and outliers. This is achieved through incremental adaptation of the input.

In contrast to related approaches such as Self-Organizing Maps (SOM[4]), GNG does not assume any dimensionality of the resulting network which is in particular useful for surface reconstruction of free-formed objects. Based on this fundamental works (Fritzke[1], Kohonen[4]) several authors proposed approaches for surface reconstruction (e.g. [2], [5], [3], [6]). However, the majority of these contributions do not take into account short response times and assume perfect point clouds. For instance, far more than several seconds for reconstructing a surface are required or high-resolution point clouds ($\gg$10000 points) from objects are gathered with less noisy sensors like laser range scanners. Furthermore, the evaluations are performed with noise-free CAD or synthetic models which are mostly not available in robotics and do not comply with the sensed point clouds from RGB-D cameras.

III. Surface Reconstruction with Growing Neural Gas

In an unsupervised learning manner, GNG reflects the topology of the input distribution as an undirected graph. Basically the GNG approach proposed by Fritzke[1] is used with modifications and extensions in this work. The basic algorithm is shown in Algorithm 1, whereas the modified algorithm is shown in Algorithm 2.

Consecutively a randomly selected point of a point cloud P is fired as a signal (input) into the Growing Neural Gas G (Algorithm 1: step 4). A competitive *Hebbian* learning-like algorithm(Algorithm 1: step 5-9) is applied to generate edges between neurons which are nearest to the signal. In the original algorithm [1] a neuron is only added after a certain interval λ (Algorithm 1: step 11). It is shown that after a sufficient number of iterations of firing signals into G, that G converges to a *Delaunay* triangulated mesh.

This algorithm offers further benefits over a simple meshing algorithm: due to the incremental adaptation of GNG, it *denoises* and *reorganizes* the input distribution in such a way that only concise properties of the point cloud are reflected in the number of neurons, position and the connections between the neurons. Eventually, the connected neurons of the neural gas generate a mesh for a cup as shown in Fig.1.

Concerning the surface reconstruction problem, adding a new neuron incrementally after a certain interval λ to G (Algorithm 1: step 11) does not consider the actual divergence between G and the P. In order to adapt to this problem a new neuron n_{new} is only added to G if the $\triangle error(\cdot)$ of an existing

Christian A. Mueller, Nico Hochgeschwende and Paul G. Ploeger are with the Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany - christian.mueller@smail.inf.h-brs.de, {nico.hochgeschwender,paul.ploeger}@h-brs.de

Algorithm 1 Basic GNG Algorithm – Fritzke[1]

```
 1: Select randomly two points $p_1$ and $p_2$ from $P$ with
    position $\omega_{p_1}$ and $\omega_{p_2}$ to initialize $G$ (GNG).
 2: Add $p_1$ and $p_2$ to $G$.
 3: Create an edge between $p_1$ and $p_2$.
 4: Select a randomly point (aka. signal) $p$ from $P$.
 5: Find nearest neuron $n_1$ and second nearest neuron $n_2$ in
    $G$.
 6: Increment the age of edges connected $n_1$.
 7: Add distance between $n_1$ and $p$ to error counter of $n_1$.
    $\triangle error(n_1) = ||n_1 - p||_2$.
 8: Update positions($\omega$) of $n_1$ and $n_n$ where $_n$ are neighbors
    of $n_1$. Move $n_1$ and $n_n$ towards $p$ by a fraction of $\varepsilon_b$ and
    $\varepsilon_b$.
    $\omega_{n_1} = \varepsilon_b(||n_1 - p||_2)$, $\omega_{n_n} = \varepsilon_n(||n_n - p||_2)$
 9: Connect $n_1$ and $n_2$ if they are not connected.If they were
    connected set the edge age to 0.
10: Delete edges which are older than $a_{max}$. Remove neurons
    which do not have any edge connected to.
11: After $\lambda$ selected signals add a new neuron $n_{new}$ to G as
    follows:
    •Identify the neuron $n_q$ with larges $\triangle error(n_q)$.
    •Insert $n_{new}$ between $n_q$ and the neighbor $n_f$ with larges
    $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
    •Connect $n_{new}$ with $n_q$ and $n_f$ and remove the edge
    between $n_q$ and $n_f$.
    •Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by $\alpha$ and set
    $\triangle error(n_{new})$ with $\triangle error(n_q)$.
12: Decrease all $\triangle error(\cdot)$ with $d$.
13: Continue with step 4 till stopping criterion(e.g. $G$ has
    reached max. number of neurons) is reached
```

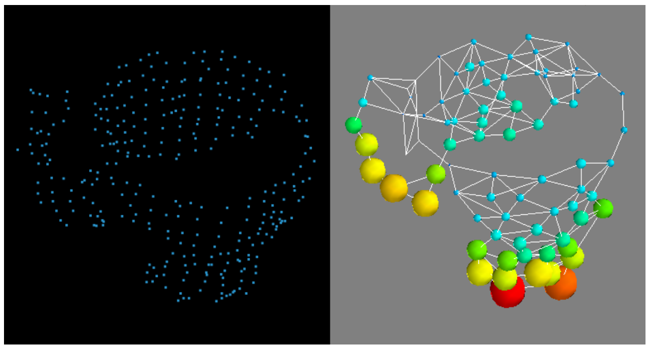


Fig. 1. A grown neural gas mesh(right) form a cup point cloud(left). Note that, a partial point cloud observation is given as input. In this and the following figures, the neurons in a GNG are sized and colored by the mean All-Pair-Shortest-Path(APSP) distances in order to illustrate the distribution and topology of a neuron with respect to the other neurons. Thereby with the APSP method the shortest paths in the GNG from each neuron to all other neurons is computed.

neuron in G exceeds a certain threshold t_γ (Algorithm 2: step 12).

As an further extension, G is *retrained*[1], i.e. the input set is repeatedly fired into G so that after each epoch, G is more adapted to the actual point cloud distribution. This guarantees that the position distribution of the neurons is close to the one of the points from the point cloud. The retraining process is stopped if the mean distance $\varepsilon(\cdot)$ between the position of the neurons n_i and the nearest points of the point cloud P reaches a lower bounded threshold t_α (Algorithm 2: step 18), instead of reaching a certain size of G (Algorithm 1: step 13). The computation of the mean distance $\varepsilon(\cdot)$ is defined as shown in Eq. 1 where N denotes the number of neurons in G. For each neuron n_i the distance to the nearest point of the point cloud P is computed; the euclidean distance (Eq. 2) is used as distance measure between two points p and q with the dimensionality of m[2].

$$\varepsilon(G) = \frac{\sum_{i=0}^{N} ||n_i - findNearestPointToNeuron(P, n_i)||_2}{N} \tag{1}$$

$$d(p,q) = ||p-q||_2 = \sqrt{\sum_{i=1}^{m}(p_i - q_i)^2} \tag{2}$$

This retraining criterion is computationally efficient and still provides a satisfying feedback about the condition of reconstruction process, for instance other measurements applied like the Hausdorff Distance in Yoon et al.[6] are computationally more expensive.

Additionally, if the distance of a neuron to the nearest point of P is above an upper bounded threshold t_β, i.e. the neuron is strongly diverted from the actual input distribution, then this neuron is removed from G – this procedure is denoted as *consistency check*($\Delta_1(\cdot)$, see Algorithm 2: step 15). This guarantees that in the current epoch of the surface reconstruction process, no nodes are contained which are certainly not part of the actual point cloud distribution. This will guide the reconstruction processes in situations where e.g. concave surfaces (e.g. cups or bowls) are reconstructed: due to the unsupervised learning manner of GNG Neurons might be attracted to the concave inner area – especially in the first epochs.

Moreover, rather than repeatedly firing the identical input set in each epoch into G, to each point of P a Gaussian noise is added ($\Delta_2(\cdot)$, see Algorithm 2: step 17) which *enriches the variation* of the input set; subsequently this results in an accelerated triangulation of a mesh compared to a mesh where adding noise is neglected (see Fig.5). We believe the acceleration can be explained by the larger variations of P with near-model-points (Gaussian noise added points of P). Due to this variation, the actual topology of the input is stronger reflected in the resulting G.

Finally in each iteration (epoch) a *refinement* procedure $\Delta_3(\cdot)$ (Algorithm 2: step 16) is applied, which only retrains

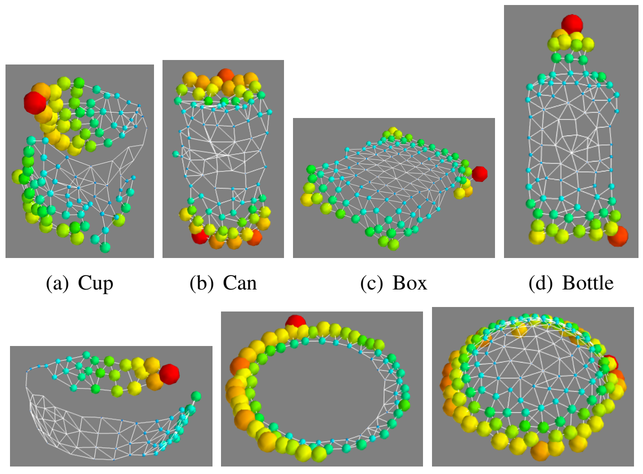
[1]Note that, a single retraining of G is also denoted as epoch.

[2]m = 3 due to the 3D Cartesian coordinates x,y,z of each point respectively neuron.

points from the point cloud which are positioned at a large curvature. This step will reinforce and consequently retain relevant surface properties of the point cloud in the GNG surface reconstruction process; e.g. surface properties which have a large curvature are edges or curves at box or cup instances.

Algorithm 2 Modified GNG Algorithm based on Fritzke[1]. Modifications are marked with $\star$.

```
1:  $\star$ Buffer $P$ in $P_{ori}$.
2:  Select randomly two points $p_1$ and $p_2$ from $P$ with position $\omega_{p_1}$ and $\omega_{p_2}$ to initialize $G$ (GNG).
3:  Add $p_1$ and $p_2$ to $G$.
4:  Create an edge between $p_1$ and $p_2$.
5:  Select a randomly point (aka. signal) $p$ from $P$.
6:  Find nearest neuron $n_1$ and second nearest neuron $n_2$ in $G$.
7:  Increment the age of edges connected $n_1$.
8:  Add distance between $n_1$ and $p$ to error counter of $n_1$.
    $\triangle error(n_1) = ||n_1 - p||_2$.
9:  Update positions($\omega$) of $n_1$ and $n_n$ where $_n$ are neighbors of $n_1$. Move $n_1$ and $n_n$ towards $p$ by a fraction of $\varepsilon_b$ and $\varepsilon_b$.
    $\omega_{n_1} = \varepsilon_b(||n_1 - p||_2)$, $\omega_{n_n} = \varepsilon_n(||n_n - p||_2)$
10: Connect $n_1$ and $n_2$ if they are not connected.If they were connected set the edge age to 0.
11: Delete edges which are older than $a_{max}$. Remove neurons which do not have any edge connected to.
12: $\star$ A new neuron $n_{new}$ is added to $G$ as follows: Identify the neuron $n_q$ with the larges $\triangle error(n_q)$. If $\triangle error(n_q) > t_\gamma$ then continue this step.
    •Insert $n_{new}$ between $n_q$ and the neighbor $n_f$ with larges $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
    •Connect $n_{new}$ with $n_q$ and $n_f$ and remove the edge between $n_q$ and $n_f$.
    •Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by $\alpha$ and set $\triangle error(n_{new})$ with $\triangle error(n_q)$.
13: Decrease all $\triangle error(\cdot)$ with $d$.
14: $\star$ Continue with step 5 till the set of points of $P$ are applied to $G$.
15: $\star$ Check for strong diverted Neurons in $G$, i.e. remove Neurons which exceed the distance $t_\beta$ to the nearest point $p$ in $P$: $G = \Delta_1(G, t_\beta)$
16: $\star$ Refine $G$: $G = \Delta_3(G)$
    • Create points set $P_{refine}$ from points in $P$ at locations with a high curvature.
    • Apply steps 5 to 15 where $P = P_{refine}$ – refinement is only once applied in each epoch.
17: $\star$ Adding Gaussian noise to each point of $P_{ori}$:
    $P = \Delta_2(P_{ori})$.
18: $\star$ Continue with a new epoch in step 5 till stopping criterion $\varepsilon(G) < t_\alpha$ is reached.
```


(a) Cup (b) Can (c) Box (d) Bottle

(e) Bowl (f) Plate (g) Ball

Fig. 2. A set of reconstructed surfaces via GNG is shown regarding the seven object categories (cup, can, box, bottle, bowl, plate, ball). Note that, missing connections like in (a) and (e) – at the rim – are due to the consistent non-existence of points in the related points clouds which also reflects the fact that *only* a partial observation from a certain perspective on the object is available; the point clouds are sensed with the Microsoft Kinect©. Moreover, note that these reconstructed surfaces do actual contain less noise, e.g. outliers.

IV. Evaluation

A. Setup

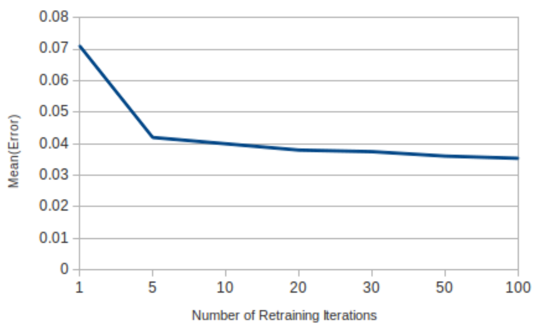
The modified GNG approach has been applied to a set of point clouds from seven different object categories, namely *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*. These object categories are chosen in order to analyze the reconstructed surfaces from primitive shapes like cylinders (*cans*) or spheres (*balls*) and also to analyze the ability to reconstruct surfaces from objects with different surface properties such as convex/concave (*cups* and *bowls*), planar (*plates* and *boxes*) or spherical (*balls*) surfaces. Some instances are illustrated in Fig.2. The surfaces are reconstructed from point clouds which do not contain more than 1000 points.

The some of the following experiments are focused on one single instance from the cup category (see Fig.1); this category is chosen due to its variety of different surface properties. Furthermore, the example point cloud from Fig.1(left) is applied in different experimental conditions in order to illustrate the behavior of the proposed GNG approach.

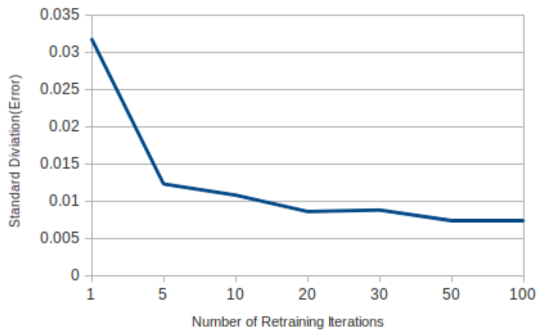
B. Experiments

Through a number of retraining epochs the GNG evolves and provides a more consistent topology projection of the input distribution. In Figure 3 the error $\varepsilon(\cdot)$ over number of epochs is analyzed between the GNG and the point clouds from the seven mentioned object categories – the error $\varepsilon(\cdot)$ is defined as shown in Eq. 1. It can be observed in Fig. 3 that the error rapidly drops after the first few epochs. Later the GNG steadily adapts to the point cloud and consequently the error decreases.

A further example is shown in Fig.4; as input the partial point cloud from Fig.1 (left) is used. It can be observed that

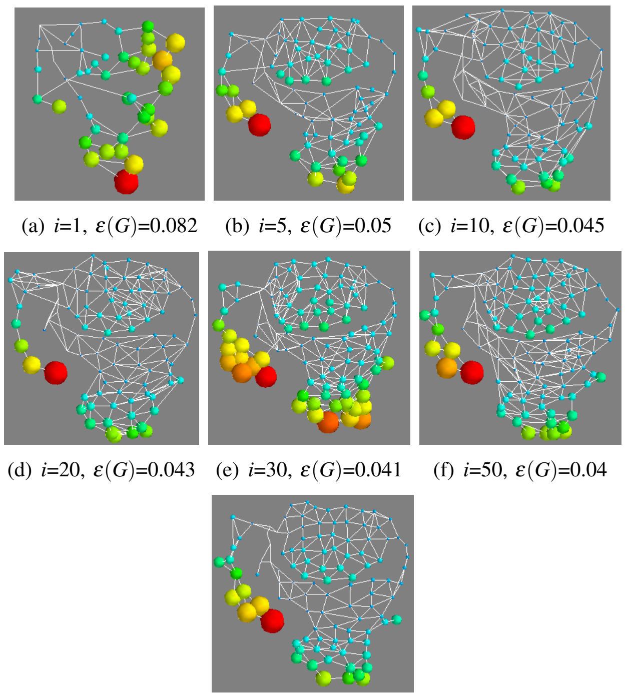


(a)



(b)

Fig. 3. Two plots which illustrate the resulting mean(a) and standard deviation(b) of the error $\varepsilon(\cdot)$ between GNGs and point clouds after a specific number of retraining iterations (epochs). The results of these plots are based on surface reconstructions from points clouds of the seven object categories (cup, can, box, bottle, bowl, plate, ball). 10 instances of each category have been used.



(a) i=1, $\varepsilon(G)$=0.082 (b) i=5, $\varepsilon(G)$=0.05 (c) i=10, $\varepsilon(G)$=0.045

(d) i=20, $\varepsilon(G)$=0.043 (e) i=30, $\varepsilon(G)$=0.041 (f) i=50, $\varepsilon(G)$=0.04

(g) i=100, $\varepsilon(G)$=0.041

Fig. 4. Example of a grown neural gas after i iterations of retraining. $\varepsilon(\cdot)$ denotes the mean error as defined in Eq. 1. The neural gas is constantly adapting to the input point cloud from Fig.1(left). The size and color of the neurons are displayed according to the mean APSP (see Fig.1).

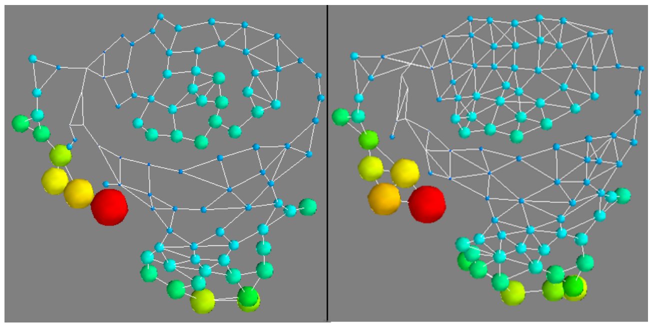


Fig. 5. Two grown neural gases trained with the point cloud from Fig.1(left). The left figure shows a grown neural gas without added noise. On the right side the figure shows a grown neural gas with added noise during retraining. In both cases the GNGs are retrained with 100 iterations. It is observable that the triangulation is more distinctive in the right figure.

the more epochs are applied the more the actual topology is projected and Delaunay triangles are observable. Also the error $\varepsilon(\cdot)$ as defined in Eq. 1 reduces over the applied epochs. However the error keeps almost steady after about 30 iterations, hence it can be assumed that the GNG is saturated by a sufficient number of neurons and their distribution is sufficiently organized over the surface. Nevertheless it is observable in Fig.4(e),(f) and (g) the error is similar with increasing iterations but the occurrences of triangles become more and more uniformly distributed.

Figure 5 shows how adding mild noise to the input point cloud[3] during the retraining process of the GNG (Algorithm 2: step 17) accelerates the Delaunay triangulation. Moreover, the number of nodes and edges have increased by applying the noise in each epoch: the number of nodes has increased from 97 to 107 whereas the number of edges from 158 to 237. As mentioned in the previous section III, we believe the acceleration of the triangulation is due to the variation enrichment of the points from the point cloud in each epoch. Based on this variation and the incremental learning process of GNG, the surface is able to be reconstruct in a more detailed and consistent manner.

Also mentioned in section III, the training process of GNG has a denoising effect on the input. This is shown in Fig. 6: the input point cloud from Fig.1(left) is reconstructed by the proposed GNG approach. Despite the noisy point cloud (e.g. caused by noisy camera input) the GNG approach is still able to cope with it and to learn the topology of the point cloud.

Finally runtime experiments have shown that the average response time is about 115 ms – considering the given seven object categories where 10 instances from each category are used.

V. Conclusion and Future Work

In this paper an approach is proposed for 3D surface reconstruction with Growing Neural Gas. Surface reconstruction based on point clouds which are sensed from the real world is challenging due to noisy sensors. The basic GNG

[3]10% Gaussian noise to each point of the point cloud is applied.

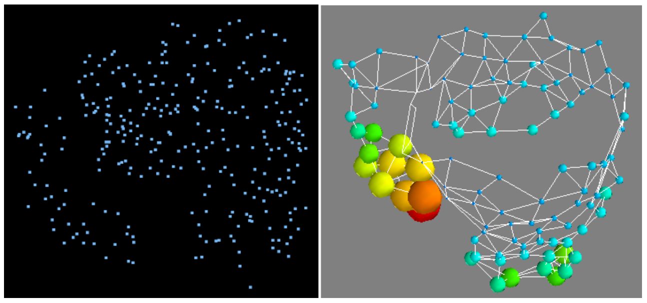


Fig. 6. The left figure displays the point cloud from Fig.1(left) with extensive noise. The resulting grown neural gas after 100 iterations is shown right. Note that the distribution – respectively topology – of the neurons is still similar to the GNG from Fig.1(right).

algorithm has been modified to adapt to the problem of 3D surface reconstruction. It has been shown that GNG provides some benefits which leads to an attractive alternative to standard triangulation approaches and also copes with noisy conditions and low response time constraints.

The future work is directed to object perception tasks, like recognition and categorization based on the proposed surface reconstruction algorithm. Typical reconstructed surfaces such as shown in Fig.2 will be used for object description purposes followed by typically learning algorithms in order to distinguish different object instances or categories. Moreover, additional investigations about the behavior of GNG are planned, e.g. measuring the quality of the triangulation and also the effect of the quality of the GNG compared to the resulting recognition, respectively categorization rate.

Acknowledgement

We gratefully acknowledge the continued support by the b-it Bonn-Aachen International Center for Information Technology.

References

[1] B. Fritzke, “A Growing Neural Gas Network Learns Topologies,” in *NIPS*, 1994, pp. 625–632.

[2] I. Ivrissimtzis, W.-K. Jeong, and H.-P. Seidel, “Using growing cell structures for surface reconstruction,” *2003 Shape Modeling International.*, vol. 2003, pp. 78–86, 2003.

[3] A. D. M. B. Junior, A. a. D. D. Neto, J. D. de Melo, and L. M. G. Goncalves, “An adaptive learning approach for 3-D surface reconstruction from point clouds.” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 19, no. 6, pp. 1130–40, June 2008.

[4] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[5] F. Krause, A. Fischer, N. Gross, and J. Barhak, “Reconstruction of freeform objects with arbitrary topology using neural networks and subdivision techniques,” *CIRP Annals-Manufacturing Technology*, vol. 52, no. 1, pp. 125–128, 2003.

[6] M. Yoon, I. Ivrissimtzis, and S. Lee, “Self-Organising Maps for Implicit Surface Reconstruction,” *Theory and Practice*, 2008.