SMART GROWING CELLS

Hendrik Annuth and Christian-A. Bohn

Computer Graphics & Virtual Reality, Wedel University of Applied Sciences, Feldstr. 143, Wedel, FR, Germany

Keywords: Neural networks, Unsupervised learning, Self-organization, Growing cells structures, Surface reconstruction.

Abstract: General unsupervised learning or self-organization places *n*-dimensional reference vectors in order to match
              the distribution of samples in an *n*-dimensional vector space. Beside this abstract view on self-organization
              there are many applications where training — focused on the sample distribution only — does not lead to a
              satisfactory match between reference cells and samples.
              Kohonen's self-organizing map, for example, overcomes pure unsupervised learning by augmenting an addi-
              tional 2D topology. And although pure unsupervised learning is restricted therewith, the result is valuable in
              applications where an additional 2D structure hidden in the sample distribution should be recognized.
              In this work, we generalize this idea of application-focused trimming of ideal, unsupervised learning and
              reinforce it through the application of surface reconstruction from 3D point samples. Our approach is based
              on Fritzke's *growing cells structures* (GCS) (Fritzke, 1993) which we extend to the *smart growing cells* (SGC)
              by grafting cells by a higher-level intelligence beyond the classical distribution matching capabilities.
              Surface reconstruction with smart growing cells outperforms most neural network based approaches and it
              achieves several advantages compared to classical reconstruction methods.

1 INTRODUCTION

The idea of developing the smart growing cells ap-
proach is driven by the need for an algorithm for
robust surface reconstruction from 3D point sample
clouds.

The demand for efficient high quality reconstruc-
tion algorithms has grown significantly in the last
decade, since the usage of 3D point scans has widely
been spread into new application areas. These in-
clude geometric modeling to supplement interactive
creation of virtual scenes, registering landscapes for
navigation devices, tracking of persons or objects in
virtual reality applications, medicine, or reverse engi-
neering.

3D points, retrieved by laser scanners or stereo
cameras, introduce two vital questions. First, how
can one recognize a topology of the originating 2D
surfaces just from independent 3D sample points and
without any other information from the sampled ob-
jects? Second, for further processing, how is it pos-
sible to project this topological information on a data
structure like a triangle mesh — meeting given con-
straints concerning mesh quality and size?

Although this issue has intensely been tackled
since the early eighties (Boissonnat, 1984) a general

concept that addresses all the problems of surface
reconstruction has not been determined up to now.
Noise contained in the sample data, anisotropic point
densities, holes and discontinuities like edges, and fi-
nally, handling vast amounts of sampling data with
adequate computing resources are still a big chal-
lenge.

Previous Work. The issue of surface reconstruc-
tion is a major field in computer graphics. There are
numerous approaches with different algorithmic con-
cepts. In (Hoppe et al., 1992) and (Hoppe, 2008) an
implicit surface is created from point clouds which
then is triangulated by the marching cubes approach.
(Edelsbrunner and Mcke, 1994) and (Kolluri et al.,
2004) reduce a delaunay tetrahedralization of a point
cloud until the model is carved out. Approaches like
(Storvik, 1996) or (Huang et al., 2007) utilize tech-
niques based on the Bayes' theorem.

In the area of artificial neural networks a famous
work is (Kohonen, 1982). They propose the *Self Or-
ganizing Map* (SOM) which iteratively adapts its in-
ternal structure — a 2D mesh — to the distribution of
a set of samples and enables clustering or dimension-
ality reduction of the sample data. While a SOM has
a fixed topology, the growing cells structures concept

(Fritzke, 1993; Fritzke, 1995) allows the network for dynamically fitting its size to the sample data complexity. SOM and GCS are suitable for processing and representing vector data like point samples on surfaces. (Hoffmann and Vrady, 1998) uses a SOM and (Vrady et al., 1999) and (Yu, 1999) a GCS for the purpose of surface reconstruction. Further improvements are made by (Ivrissimtzis et al., 2003b) where constant Laplacian smoothing (Taubin, 1995) of surfaces is introduced, and in (Ivrissimtzis et al., 2003a) the curvature described by the input sample distribution is taken to control mesh density. In (Ivrissimtzis et al., 2004a) the GCS reconstruction process is further enhanced in order to account for more complex topologies. (Ivrissimtzis et al., 2004b) use several meshes of the same model for a mesh optimization process, and (Yoon et al., 2007) present a concept for combining common deterministic approaches and the advantages of the GCS approach.

Overview. In the following, we outline the basis of our approach — the growing cells structures — and then derive our idea of the smart growing cells, which matches the specific requirements of reconstruction. Afterwards, an analysis is compiled discussing mesh quality and performance of our approach, and finally, we close with a summary and a list of future options of this work.

2 RECONSTRUCTION WITH SMART GROWING CELLS

Classical growing cells approaches for reconstruction tasks are based on using the internal structure of the network as a triangulation of the object described by a set of surface sample points. A 2D GCS with 3D *cells* is trained by 3D points. Finally, the cells lie on the object surface which the 3D points represent and the network structure — a set of 2D simplices (triangles) — is directly taken as triangulation of the underlying 3D object.

The reason for using a GCS scheme for reconstruction tasks are its obvious advantages compared to deterministic approaches.

- They can robustly handle arbitrary sample set sizes and distributions which is important in case of billions of unstructured points.

- They are capable of reducing noise and ply discontinuities in the input data.

- They are capable of adaption — it is not required to regard all points of the sample set on the whole.

Further, incrementally retrieved samples can be used to retrain the network without starting the triangulation process from scratch.

- They guarantee to theoretically find the best solution possible. Thus, approximation accuracy and mesh quality are automatically maximized.

Nevertheless, these advantages partly clash with the application of reconstruction. On the one hand, discontinuities are often desired (for example, in case of edges or very small structures on object surfaces). On the other hand, smoothing often destroys important aspects of the model under consideration (for example, if holes are patched, if separate parts of the underlying objects melt into one object, or if the object has a very complex, detailed structure). In such cases, GCS tend to generalize which may be advantageous from the physical point of view, but which mostly lets vanish visually important features which the human is quite sensitized to.

The presented smart growing cells approach accounts for these application-focused issues and emphasizes that modification of the general learning task in the classical GCS is suitable for many novel application fields.

2.1 Unsupervised Learning and Growing Cells Structures

General unsupervised learning is very similar to *k-means clustering* (MacQueen, 1967) which is capable of placing k n-dimensional reference vectors in a set of n-dimensional input samples such that they are means of those samples which lie in the n-dimensional Voronoi volume of the reference vectors.

Adaption of reference vectors is accomplished by randomly presenting single n-dimensional samples from the input sample set to the set of n-dimensional reference vectors and moving them in n-dimensional space, described as follows.

Place k reference vectors $\mathbf{c}_i \in \mathbb{R}^n$, $i \in \{0..k-1\}$ randomly in nD space of input samples.

repeat

Chose sample $\mathbf{s}_j \in \mathbb{R}^n$ randomly from the input set.

Determine reference vector $\mathbf{c}_b$ (*best matching* or *winning unit*) closest to $\mathbf{s}_j$.

Move $\mathbf{c}_b$ in the direction of $\mathbf{s}_j$ according to a certain strength ε_{bm}, like $\mathbf{c}_b^{\text{new}} = \mathbf{c}_b^{\text{old}}(1-\varepsilon_{bm}) + \mathbf{s}_j \cdot \varepsilon_{bm}$.

Decrease ε_{bm}.

until $\varepsilon_{bm} \leq$ certain threshold ε_0.

Surface reconstruction with pure unsupervised learning would place a set of reference vectors on object surfaces, but does not determine information about the underlying surface topology. This leads to the Kohonen Self Organizing Map.

Kohonen Self Organizing Map. The SOM is based on reference vectors which now are connected as a regular 2D mesh. The learning rule is extended to account for the direct neighborhood of a best matching unit as follows.

> **for all** $\mathbf{c}_{nb} \in$ neigborhood of $\mathbf{c}_b$ **do**
>
> > Move $\mathbf{c}_{nb}$ in the direction of $\mathbf{s}_j$ according to a certain strength ε_{nb}, like $\mathbf{c}_{nb}^{\text{new}} = \mathbf{c}_{nb}^{\text{old}}(1 - \varepsilon_{nb}) + \mathbf{s}_j \cdot \varepsilon_{nb}$.
> >
> > Decrease ε_{nb}.
>
> **end for**

Insertion of this neighborhood loop into the general unsupervised learning algorithm (after moving of $\mathbf{c}_b$) leads to the phenomenon that the reference vertices now are moved by accounting for the regular 2D mesh topology of the SOM. Training a plane-like sample set leads to an adaption of the SOM grid to this implicit plane — the sample topology is recognized and finally represented by the SOM mesh.

Nevertheless, the mesh size of a SOM is fixed and cannot adjust to the sample structure complexity. The growing cells structures approach overcomes this drawback.

Growing Cells Structures. To a certain degree, GCS may be seen as SOM which additionally are capable of growing and shrinking according to the problem under consideration which is defined by the sample distribution. This mechanism is based on a so called *resource term* contained in every reference vector and which — in the original approach — is a simple counter. It counts the reference vector being a best matching unit. A high counter value signalizes the requirement for insertion of new reference vectors.

With a GCS we could train a sample set lying on a certain object surface and the network structure would fit the object surface at a certain approximation error. The problem is that in reconstruction tasks sample distributions are often not uniform. The represented surfaces usually contain discontinuities like sharp edges and holes, and the objects to be reconstructed are not that simple like a plane or a tetrahedron — which usually are chosen as initial network and which can hardly adapt to complex topologies.

Only objects which are homeomorphic to the start object can be represented satisfactorily.

Thus, general unsupervised learning should evolve to a kind of constrained unsupervised learning which detects and adapts to certain structures which the sample set implicitly contains.

2.2 Smart Growing Cells

Smart growing cells are an application-focused, six-way adaption of the general learning scheme of the classical growing cells structures approach. The SGC basic structure is identical to general GCS. There are n-dimensional cells which we now term *neural vertices* connected by *links* through an m-dimensional topology.

We let $n = 3$ since neural vertices are directly taken as vertices of the triangulation mesh and $m = 2$ since we aim at 2D surfaces to be reconstructed.

The main training loop is outlined in Fig. 1. Here k_{del} and k_{ins} are simple counter parameters defined below (see section 2.3). Movements of vertices and their neighbors slightly differ from the classical SOM. Again, there are two parameters for the learning rates, ε_{bm} for the winner and ε_{nb} for its neighbors, but these are not decreased during learning since vertex con-

> **repeat**
>
> > **for** $j = 1$ to k_{del} **do**
> > > **for** $i = 1$ to k_{ins} **do**
> > >
> > > > Choose sample s from point cloud randomly, find closest neural vertex and move it together with neighbor vertices towards s.
> > > >
> > > > Increase signal counter at s (the resource term mentioned above) and decrease the signal counters of all other vertices.
> > >
> > > **end for**
> > >
> > > Find best performing neural vertex (with highest signal counter value) and add new vertex at this position (see Fig. 2).
> >
> > **end for**
> >
> > Find worst performing neural vertices, delete them and collapse regarding edges (see Fig. 2).
>
> **until** certain limit like approximation error, or number of vertices is reached.

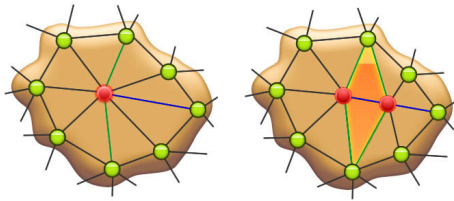Figure 1: Classical growing cells structures algorithm.

Figure 2: Neural vertex split operation (from left to right) to increase mesh granularity locally, and edge collapse (from right to left) to shrink mesh locally.

Figure 3: Statue's bottom is not represented by samples. On the right, the acute-angled triangles expose a degenerated mesh region.

nections automatically become smaller together with the learning rates. For drawing the neighboring vertices, a smoothing process like described in (Ivrissimtzis et al., 2003b) and (Taubin, 1995) is applied, which replaces the classical movement, and which makes the adaption of the topology more robust.

As initial network, usually a tetrahedron or a plane with random vertices is suitable.

Vertex Split. A neural vertex split operation adds three edges, two faces, and a new neural vertex. The longest edge at the neural vertex with the highest resource term is split and a new vertex is added in the middle. The signal counter value is equally spread between the two vertices (see Fig. 2).

Edge Collapse. All neural vertices with resource terms below a certain threshold r_{min} are removed together with three edges and two connected faces (see Fig. 2). The determination of the edge to be removed is driven by connectivity irregularities as proposed in (Ivrissimtzis et al., 2003b).

It follows our adaption of the mentioned learning cycle by six modifications driven by the application needs of surface reconstruction.

2.2.1 Cell Weeding

Aggressively deleting neural vertices which are not part of a sound underlying mesh structure is the most important new training rule of the SGC approach. It is essential for giving the network the chance of adapting to any topology despite of its initial topology (overcoming the homomorphic restriction). Before the edge collapse operation is applied at a vertex, it will be tested if the vertex is contained in a *degenerated mesh region* (definition follows below). If so, an aggressive cut out of the vertex and its surrounding vertices is started.

It has been shown that degeneration of a part of a mesh serves as perfect indicator for a mesh topology which does not fit the underlying sample structure correctly. For example, consider a region where
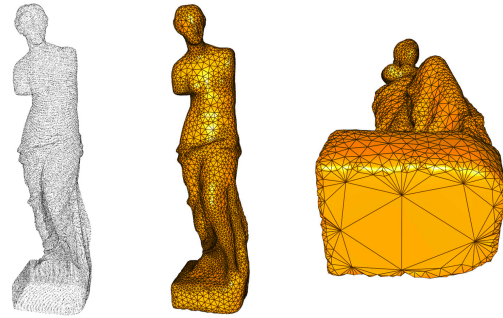
sample densities equal zero. Although vertices are not directly drawn into it by training adjustment, their neighbors may be moved there through their mesh connections. Due to their resource terms, these vertices will be deleted by edge collapse operations, but their links remain and mistakenly represent the existence of some topology. In this case, the structure of the links is degenerated, i.e., it usually shows a surpassing number of edges with *acute-angled*[1] vertices (see Fig. 3).

The reason for terming this deletion "aggressive" are the triggering properties which are easy to match — suspicious neural vertices will be cut out early.

Criterion for Degenerated Mesh Regions. In (Ivrissimtzis et al., 2004a) a large area of a triangle is taken as sign for a degenerated mesh structure, but it has been shown that this criterion warns very late. Also, anisotropic sample densities are mistakenly interpreted as degenerated mesh regions. Our proposal is a combination of *vertex valence*[2], triangle quality, and quality of neighboring vertices. If all of the following conditions hold, a deleting of the mesh structure at that vertex is started.

1. Vertex valence rises above a certain threshold $n_{degvalence}$.

2. The vertex is connected to at least $n_{degacute}$ acute-angled triangles.

3. The vertex has more then n_{degnb} neighboring vertices for which condition (1) or (2) hold.

The latter condition says that deletion is only started if at least one or two neighbors have the same inconsistencies in their local mesh structure. This is rea-

---

[1]A triangle is termed acute-angled if the ratio of its area and the area which is spanned by a second equilateral triangle built from the longest edge of the first lies below a certain threshold ε_{acute}.
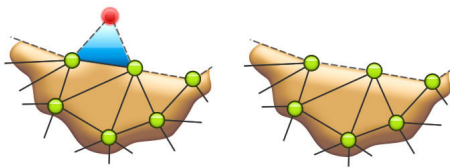
[2]Vertex valence is the number of connected vertices.

Figure 4: Curing a boundary with a spike.

Figure 6: Cut out process of a needle eye.

sonable since single degenerated vertices do not necessarily expose a problem but may arise by accident.

Curing Boundaries after Weeding. It is obvious, that after an aggressive extinction of a neural vertex and its surrounding faces, a boundary will be left behind which may consist of unfavorable mesh structure elements. Curing finds these structures along the boundary and patches them discriminating between four cases.

Spike. A boundary vertex with a valence of 2 (see Fig. 4) is termed spike. Such a vertex is very unlikely to support a correct reconstruction process since it will be adjusted to an acute-angled triangle after few iteration steps. A spike must be deleted completely.

Nasty Vertex. A nasty vertex is a neural vertex with at least $n_{nastyacute}$ acute-angled triangles and/or triangles with a valence greater than $n_{nastyval}$ (see Fig. 5). These vertices are suspected to be part of a degenerated mesh region and are deleted.

Needle Eye. A needle eye is a neural vertex that is connected to at least two boundaries (see Fig. 6). At these locations the mesh does not have a valid mesh structure. To delete a needle eye, all groups of connected faces are determined. From these, the group with the most faces is kept and all others are deleted.

Bridge. A bridge is very likely to be part of a degenerated mesh region. If a mesh has a hole that consist of three vertices, then it would soon be closed by a coalescing process (see section 2.2.2). This is not allowed if exactly one of the edges of this hole would additionally be connected to a face (which we term "bridge", see Fig. 7) since an invalid edge with
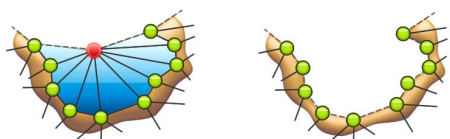
three faces would arise. The entire bridge structure is deleted and the hole will be closed with a new face.

Multiple Boundary Search Through. After the deletion of a neural vertex by the cell weeding process the curing mechanism will search for unfavorable structures along the boundary. There is more than one boundary to be considered, if the deletion destroys a coherent set of faces and multiple separate groups of faces arise.

Four cases may appear. First, the usual case with no additional boundaries. Second, when a needle eye is destroyed, the boundaries of all groups of connected faces need to be tested. Third, when surrounding faces of a vertex are interrupted by boundaries. And fourth, when a needle eye is connected to the surrounding faces of a vertex (see Fig. 8). In other words, these cases happen since the faces that are deleted may not necessarily be connected to a further face due to another deletion process.

2.2.2 Coalescing Cells

Like the mesh can be split through deletion of vertices, it must also be possible to merge two mesh boundaries during training. For that, a coalescing test is accomplished each time a vertex at a mesh boundary is moved.

Coalescing Test. It determines if two boundaries are likely to be connected to one coherent area. For that, a sphere is created with the following parameters. Given the neigboring boundary vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ of $\mathbf{c}_b$, then we define $\mathbf{c} = \mathbf{1/2}(\mathbf{v_1} + \mathbf{v_2})$. A *boundary normal* $\mathbf{n}_c$ is calculated as the average of all vectors originating at $\mathbf{c}$ and ending at neighbors of $\mathbf{c}_b$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are not taken into account. The boundary

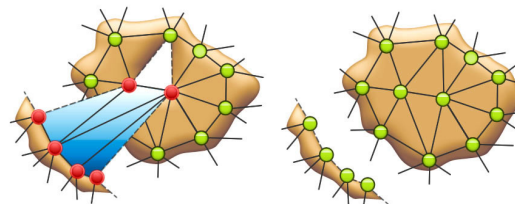Figure 5: Cut out process of a nasty vertex.
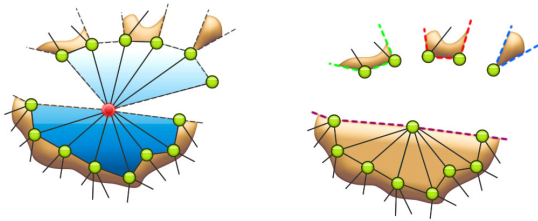

Figure 7: Curing a bridge.

Figure 8: Cut out of a needle eye with a row of faces. Here, each face is not necessarily connected to another face. In contrast, if a needle eye has several groups of connected faces then there are some omissions of faces around it.

normal can be seen as a direction pointing to the opposite side of the boundary. We define a sphere with the center at $\mathbf{c} + \mathbf{n}_c r$ with radius r as the average length of the edges at $\mathbf{c}_b$.

The coalescing condition at two boundaries hold, i.e., merging of the boundaries containing $\mathbf{c}_b$ and $\mathbf{q}$ on the opposite side happens, if

- $\mathbf{q}$ is contained in the defined sphere, and
- scalar product of the boundary normals at $\mathbf{c}_b$ and $\mathbf{q}$ is negative.

Coalescing Process. After detecting the neural vertex $\mathbf{q}$ to be connected with $\mathbf{c}_b$, the according faces must be created starting with one edge from $\mathbf{c}_b$ to $\mathbf{q}$. There are two cases which have to be considered.

Corner. A corner of the same boundary arises when $\mathbf{c}_b$ an $\mathbf{q}$ have one neighboring vertex in common (see Fig. 9). A triangle of the three participating vertices is created.

Long Side. Here, two boundaries appear to be separated. After determining the new edge, there are four possibilities for insertion of a new face containing the edge (see second picture in Fig. 10). The triangle with edge lengths which vary fewest is taken in our approach (see third picture in Fig. 10) since it is the triangle with the best features concerning triangle quality. Finally, to avoid a needle eye, a further triangle must be added — again, we take the face with
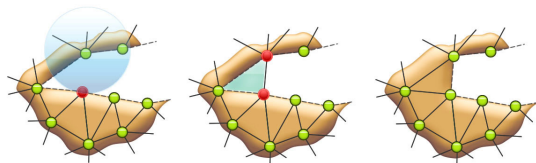

Figure 9: Coalescing process at a mesh corner. On the left, the search process of a coalescing candidate. In the middle, one edge is created, on the right, the only face capable of being added is the corner face.
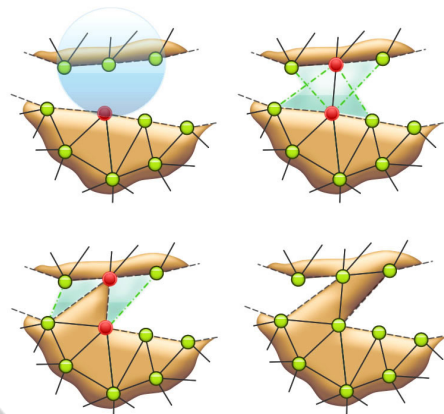

Figure 10: Coalescing of two separate boundaries. In the second picture, the edge is determined, in the third, the triangle with smallest variance of edge lengths is added, in the fourth, another triangle must be added to avoid a needle eye.
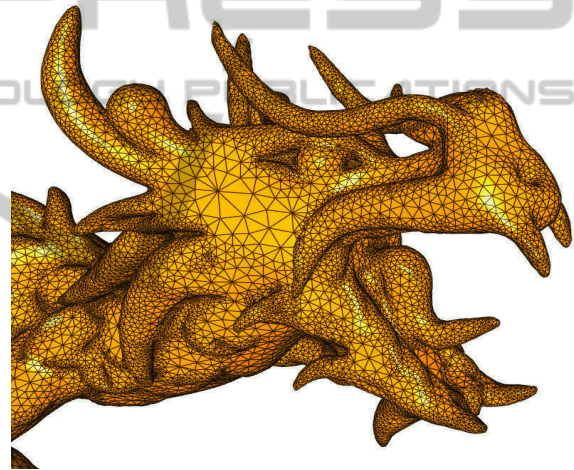

Figure 11: Roughness adaption correlates surface curvature with mesh density, details of the model are exposed.

the greatest edge similarity (see fourth picture in Fig. 10).

2.2.3 Roughness Adaption

Up to now, the SGC are able to approximate an arbitrary sample set by a 2D mesh. What remains is an efficient local adaption of the mesh density in a way that areas with a strong curvature are modeled by a finer mesh resolution (see Fig. 11). This also relieves the influence of the sample density on the mesh granularity making the SGC less vulnerable to sampling artefacts like holes or regions which are not sampled with a uniform distribution.

Each time a vertex is adapted by a new sample we calculate the estimated normal $\mathbf{n}_k$ at a neural vertex $\mathbf{v}_k$ by the average of the normals at the surrounding

faces. The curvature $c_k \in \mathbb{R}$ at a vertex is determined by

$$c_k = 1 - \frac{1}{|\mathcal{N}_k|} \sum_{\forall \mathbf{n} \in \mathcal{N}_k} \mathbf{n}_k \cdot \mathbf{n} \qquad (1)$$

with the set $\mathcal{N}_k$ containing the normals of the neighboring neural vertices of $\mathbf{v}_k$. Each time a neural vertex is selected as winner, its curvature value is calculated and a global curvature value $\bar{c}$ is adjusted. Finally, the curvature dependent resource term r_k at $\mathbf{v}_k$ is adapted through $r_k^{new} = r_k^{old} + \Delta r_k$, and

$$\Delta r_k = \begin{cases} 1, & \text{if } (c_k < \bar{c} + \sigma_{r_k}) \\ \left[c_k / (\bar{c} + \sigma_{r_k}) \right] (1 - r_{min}) + r_{min} & \text{else,} \end{cases} \qquad (2)$$

with the deviation σ_{r_k} of the resource term r_k, and a constant resource r_{min} that guarantees that the mesh does not completely vanish at plane regions with a very small curvature.

2.2.4 Curvature Cells

Each time after a vertex $\mathbf{v}_c$ has been moved we apply a smoothing mechanism like mentioned at the beginning of section 2.2.

Roughness adaption (see section 2.2.3) leads to the fact that in regions of high curvature the density of neural vertices will increase. These vertices then will get fewer sample hits, since they have a smaller Voronoi region, and thus, Laplacian smoothing is applied fewer times.

We found out, that this significantly reduces mesh quality in areas of high curvature. To avoid this, neural vertices in regions with high curvature are marked as such and smoothing these is strengthened by repeating it n_L times where

$$n_L = \lfloor (c_k - \bar{c}) / \sigma_{c_k} \rfloor - 1 \qquad (3)$$

with c_k and $\bar{c}$ like defined in section 2.2.3 and σ_{c_k} the deviation of the curvature at vertex $\mathbf{v}_k$. The value is limited to a maximum of N_L to intercept looping at extraordinary curvature values.

2.2.5 Discontinuity Cells

A sampled model that exposes discontinuities like edges is difficult to be approximated by the neural network mesh. Discontinuities are smoothed out since the network tries to create a surface over them. This might be acceptable in many application areas since the approximation error is fairly small, but this effect is unfavorable in computer graphics since it is clearly visible. And even worse: edges are quite common in real world scenarios.

Therefore, we propose discontinuity neural vertices which, first, are only capable of moving in the
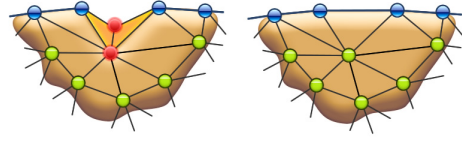

Figure 12: A dent (left picture) on a sharp edge is solved (right picture) by an edge swap operation. Finally, connections of discontinuity vertices model object edges.

direction of an object edge to represent them more properly, and second, the smoothing process is not applied to them.

Recognizing those vertices is accomplished as follows. We determine the curvature values of those neighbors which have a distance of two connections from the vertex (the "second ring" of neighbors). Then the average δ_{ring} of the squared differences of consecutive curvature values on the ring is calculated.

If a curvature value clearly deviates from the average curvature value, then we assume that it is a discontinuity vertex if the average of the neighbors' (second ring) curvature gradient differs to a certain amount. Thus, we define a vertex $\mathbf{v}_k$ being a discontinuity vertex if

$$(c_k > 2\sigma_{c_k}) \wedge (\forall c \in C_k : \delta_{ring} > 4\sigma_{c_k}^2) \qquad (4)$$

with C_k the set of curvature values of the second ring of neighbors.

For approximating the edge normal we take the average of the normals of two of the neighboring vertices of $\mathbf{v}_k$, either those with the highest curvature value, or those which are already marked as discontinuity vertex. Finally, the normal is mirrored if the edge angle lies above $180°$, which is indicated by the average of the surrounding vertex normals; in the first case it points in the direction of $\mathbf{v}_k$.

Edge swap. If two connected discontinuity vertices grow into an edge, they nicely represent this edge by a triangle edge. But if the line is interrupted by a non-discontinuity vertex, a dent arises since this vertex is not placed on the edge. Thus, we propose an edge swap process which minimizes this effect.

Each time a discontinuity vertex is moved towards a sample, the need for an edge swap operation will be determined by collecting the three consecutive faces with the most differing face normals. In case of a dent, the face in the middle is assumed to be the one which is misplaced and an edge swap operation is applied (see Fig. 12). Then, if the difference of the normals is now lower than before, edge swap is accepted, if not, the former structure is held.

Edge swap results in models where finally edges are represented by mesh boundaries (see Fig. 13).

2.2.6 Boundary Cells

Similar to discontinuity vertices which are capable of moving to object edges, boundary vertices are able to move to the outer border of a surface (see Fig. 14). They are recognized by being part of a triangle edge which is connected to one face only.

Then, these vertices are moved only in the direction of the boundary normal like described in section 2.2.2 in order to avoid vertices just lying in the average of the surrounding samples but directly match the surface boundaries at their locations.

2.3 Results

For the full algorithm of this approach see the pseudocode in Fig. 16. To keep it comprehensive, the outermost loop of the algorithm is neglected, and vertex split and edge collapse operations are triggered by counters.

Parameters which have been proven to be reliable for almost all sample sets we took for reconstruction are $\varepsilon_{bm} = 0.1$, $\varepsilon_{nb} = 0.08$, $r_{min} = 0.3$, $\varepsilon_{acute} = 0.5$, $n_{degacute} = 4$, $k_{ins} = 100$, $k_{del} = 5$, $n_{degnb} = 1$, $n_{nastyacute} = 4$, $n_{nastyval} = 3$.

The following results have been produced on a *Dell*®Precision M6400 Notebook with *Intel*®Core 2 Extreme Quad Core QX9300 (2.53GHz, 1066MHz, 12MB) processor with 8MB 1066 MHz DDR3 Dual Channel RAM. The algorithm is not parallelized.

Some visual results are exposed in Fig. 15. All pictures are drawn from an SGC mesh. Most models stem from the *Stanford 3D Scanning Repository*.

Besides visual results, reconstruction with SGC comes up with impressive numbers compared to classical approaches, listed in table 1.

It can be seen that mesh quality, i.e. the percentage of perfect triangles in the mesh lies at 96% at average. This is an outstanding result, nevertheless this is usually expected when using an approach from the field of unsupervised learning, since this guarantees an ideal representation of the underlying training
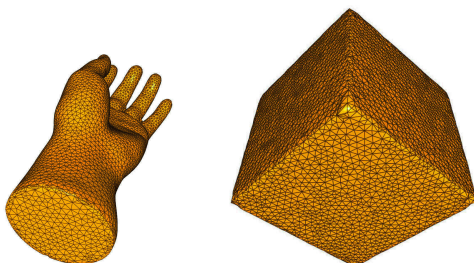

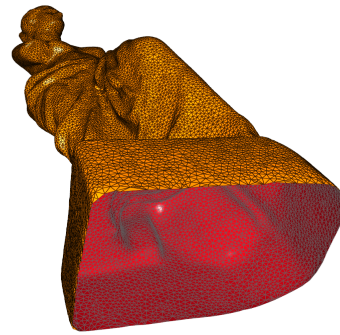Figure 13: Discontinuity vertices focus on edges. Edge swap operations let mesh edges map to object edges.

Figure 14: Mesh boundary due to the missing bottom of the statue is represented exactly by boundary cells.

Table 1: Results with sample sets from the *Stanford 3D Scanning Repository*. "Quality" means percentage of triangles which hold the Delaunay criterion. RMS/Size is the root of the squared distances between original point samples and the triangle mesh, divided by the diameter of the sample set.

	Samples	Vertices	Time [m:s]	Quality	RMS/ Size
	36K	30K	0:39	95.6%	4.7e-5
	438K	100K	2:47	95.5%	3.3e-5
	544K	260K	9:15	93.1%	1.7e-5
	14,028K	320K	12:17	98.5%	1.3e-5
	5,000K	500K	21:5	95.9%	2.7e-5
	511K	10K	0:11	99.8%	6.6e-5
	38K	5K	0:6	99.0%	15e-5
	346K	5K	0:6	98.3%	0.7e-5

sample distribution.

Further, the distance (RMS/object size) between samples and mesh surface is negligible low — far below 1% of the object size at average. This is even more pleasant, since usually, the problem at edges generate big error terms. Also the computing times needed are very short, few minutes in each case.

All those measurements seem to be far better than those from classical approaches, as long as we could extract them from the regarding papers. Our algorithm works very robustly. There are nearly no outliers visible in the mesh.
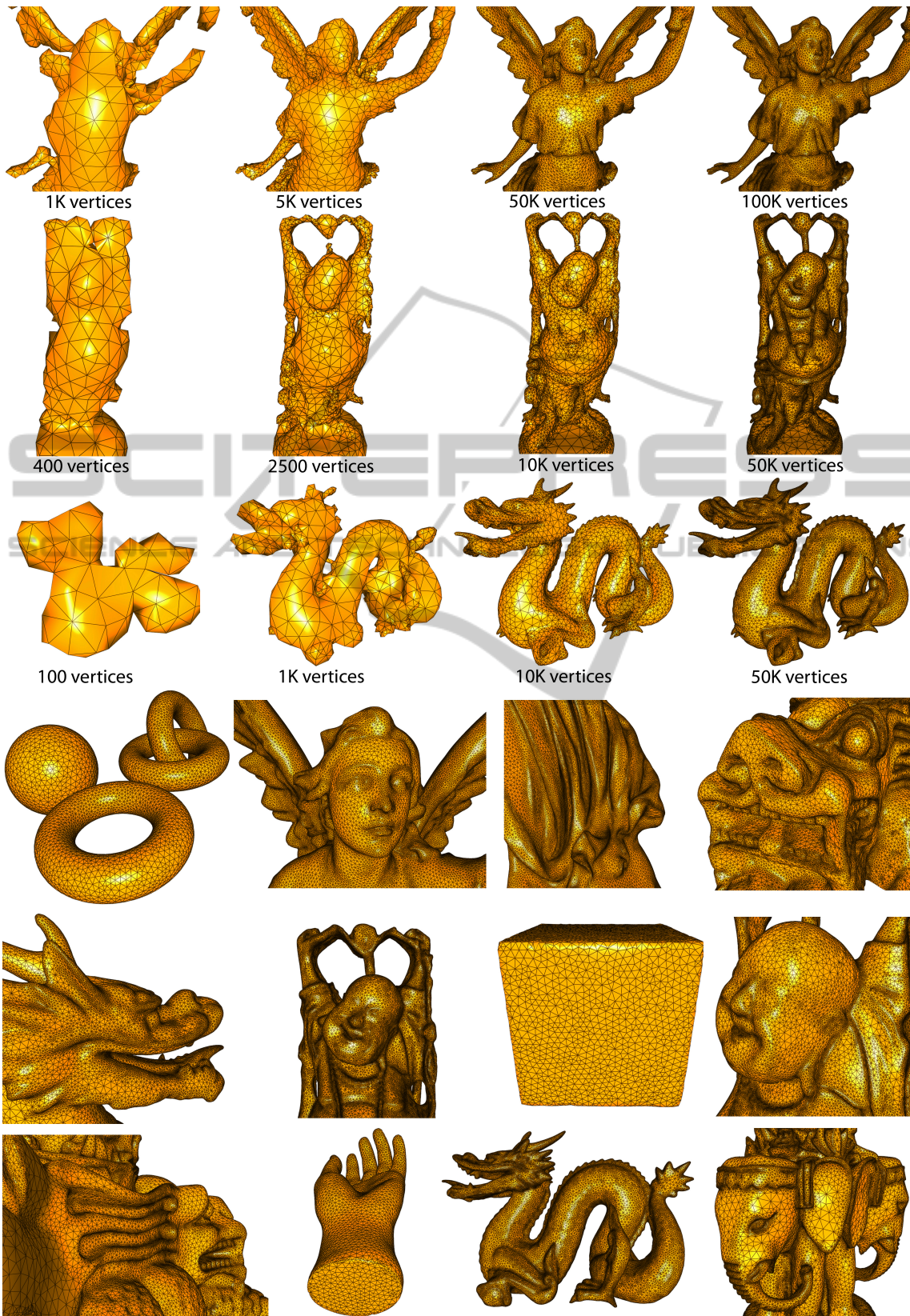
Figure 15: Upper lines: mesh training stages with number of vertices, lower lines, assorted pictures of reconstructed models.

1K vertices 5K vertices 50K vertices 100K vertices

400 vertices 2500 vertices 10K vertices 50K vertices

100 vertices 1K vertices 10K vertices 50K vertices

```
Adjust samples regarding roughness.

Calculate average curvature and deviations.

Recognize and sign discontinuity cells.

Recognize and sign curvature cells.

for all Boundary cells do
    if ∃ coalescing candidate then
        Melt boundary.
        for all Weeding candidates do
            Weeding process.
        end for
    end if
end for

if Edge collapse operation triggered then
    Collapse edge.
    for all Weeding candidates do
        Weeding process
    end for
end if

if Vertex split operation triggered then
    Split vertex.
end if
```

Figure 16: Outline of the complete SGC algorithm.

3 CONCLUSIONS

We presented a new neural network approach, the smart growing cells, which is a modification of the classical growing cells structures approach.

The modification type is new in a way that it changes the pure, general unsupervised learning scheme ad hoc to match training requirements of specific applications.

Thus, drawbacks of using unsupervised learning approaches can be avoided while advantages be retained, and nevertheless, SGC training keeps its roots at general unsupervised learning.

We encourage this idea by one specific application case — surface reconstruction from 3D point samples. Here, we add six topics to the classical unsupervised learning scheme, and finally the approach outperforms classical approaches concerning quality, efficiency, and robustness. Surface reconstruction with SGC is able to handle arbitrary topologies and millions of samples. It recognizes and solves discontinuities in the sample data and it is capable of adapting to varying sample distributions. Finally, the network is able to reorganize its topology to match arbitrary surface structures. Altogether these advantages can not be found in any of the classical approaches of surface reconstruction.

The essential issue which transforms GCS to SGC is the mechanism of weeding cells as a network cleaning mechanism for ill-formed structures. Further, face normals are regarded and included in the neural network training loop to adapt to mesh roughness and to make the reconstruction process independent from the sample distribution. Additionally, we propose coalescing cells which can connect to others, curvature cells which recognize very small structures, and discontinuity cells which account for certain discontinuous structures like sharp edges.

The proof of concept of our approach is enriched by the achieved quality and performance measures. For the tested geometries which each hold specific challenges of reconstruction, we got approximation errors for comparable mesh resolutions that lie far below 1% at average. Mesh quality, measured by the percentage of triangles which comply the Delaunay criterion, lies at 96% at average. And the time needed to compute meshes of several hundreds of thousands of polygons were just few minutes.

Future Work. This work shows that application-focused unsupervised learning is able to solve practical problems efficiently. Computation times are that small that we think of a real-time reconstruction approach through multithreaded sample adjustment.

REFERENCES

Boissonnat, J.-D. (1984). Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286.

Edelsbrunner, H. and Mcke, E. P. (1994). Three-dimensional alpha shapes.

Fritzke, B. (1993). Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7:1441–1460.

Fritzke, B. (1995). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA.

Hoffmann, M. and Vrady, L. (1998). Free-form surfaces for scattered data by neural networks. *Journal for Geometry and Graphics*, 2:1–6.

Hoppe, H. (2008). Poisson surface reconstruction and its applications. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 10–10, New York, NY, USA. ACM.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. A., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. In Thomas, J. J., editor, *SIGGRAPH*, pages 71–78. ACM.

Huang, Q.-X., Adams, B., and Wand, M. (2007). Bayesian surface reconstruction via iterative scan alignment to an optimized prototype. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 213–223, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Ivrissimtzis, I., Jeong, W.-K., Lee, S., Lee, Y., and Seidel, H.-P. (2004a). Neural meshes: surface reconstruction with a learning algorithm. Research Report MPI-I-2004-4-005, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany.

Ivrissimtzis, I., Jeong, W.-K., and Seidel, H.-P. (2003a). Neural meshes: Statistical learning methods in surface reconstruction. Technical Report MPI-I-2003-4-007, Max-Planck-Institut fr Informatik, Saarbrücken.

Ivrissimtzis, I., Lee, Y., Lee, S., Jeong, W.-K., and Seidel, H.-P. (2004b). Neural mesh ensembles. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 308–315, Washington, DC, USA. IEEE Computer Society.

Ivrissimtzis, I. P., Jeong, W.-K., and Seidel, H.-P. (2003b). Using growing cell structures for surface reconstruction. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 78, Washington, DC, USA. IEEE Computer Society.

Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43:59–69.

Kolluri, R., Shewchuk, J. R., and O'Brien, J. F. (2004). Spectral surface reconstruction from noisy point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21, New York, NY, USA. ACM.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281 – 297. University of California Press.

Storvik, G. (1996). Bayesian surface reconstruction from noisy images. In *In Interface 96*.

Taubin, G. (1995). A signal processing approach to fair surface design. In *SIGGRAPH*, pages 351–358.

Vrady, L., Hoffmann, M., and Kovcs, E. (1999). Improved free-form modelling of scattered data by dynamic neural networks. *Journal for Geometry and Graphics*, 3:177–183.

Yoon, M., Lee, Y., Lee, S., Ivrissimtzis, I., and Seidel, H.-P. (2007). Surface and normal ensembles for surface reconstruction. *Comput. Aided Des.*, 39(5):408–420.

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