Growing Cell Structures Learning a Progressive Mesh During Surface Reconstruction – A Top-Down Approach

Tom Vierjahn[1,2], Guido Lorenz[1], Sina Mostafawy[1,3], and Klaus Hinrichs[2]

[1][rmh] new media GmbH, Cologne, Germany
[2]Visualization and Computer Graphics Research Group, University of Muenster, Germany
[3]Department of Media, FH Duesseldorf University of Applied Sciences, Germany

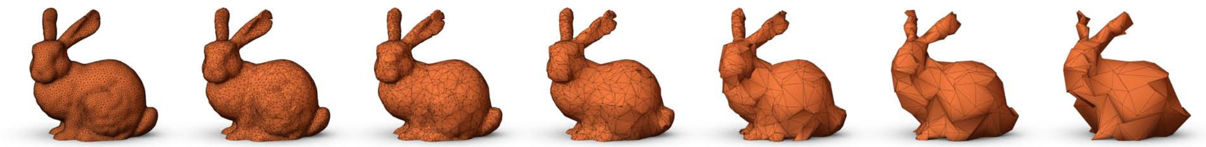

Figure 1: *Reconstruction of Stanford Bunny with 20,000 triangles (far left) and the automatically learned level of detail. For each step the number of triangles was halved until reaching 312 triangles (far right).*

Abstract

Growing Cell Structures (GCS) have been proven to be suitable for surface reconstruction from unstructured point clouds. The reconstructed triangle mesh can be represented compactly as a progressive mesh with integrated level of detail by storing only vertex split operations. However, half-edge collapse operations are used for GCS.
In this paper, we present an improvement to a GCS-based surface reconstruction technique by converting a half-edge collapse to a more general vertex removal to create a progressive mesh. We have evaluated the new technique with respect to running time overhead and mesh quality. Results indicate that this technique can be used for efficient surface reconstruction. We will use the presented findings as basis for future research.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling — Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems—

1. Introduction

Urban models reconstructed from aerial images play an essential role in different areas, e. g., map creation, surveying or disaster management. In the AVIGLE project [RGW*10] we are developing an automated reconstruction process using images taken by a swarm of miniature unmanned aerial vehicles [SFS*11]. A 3D point cloud, obtained by 3D reconstruction from the aerial images [RSH11], forms the input to the subsequent surface reconstruction process. A database server is used as a central storage for all processes.

Since we are expecting very large, dynamic point clouds, growing incrementally due to the acquisition and processing of additional images, we employ a reconstruction algorithm based on an artificial neural network capable of unsupervised learning. Since transfer of the mesh has been identified as a bottleneck, we improved the reconstruction process in such a way that the resulting mesh is represented compactly as a progressive mesh – without any post processing.

2. Related Work

First experiments in the AVIGLE project proved an artificial neural network to be generally usable for surface reconstruction [SFS*11]. Since our point clouds are dynamic, we have based our algorithm on Growing Cell Structures [Fri93] that iteratively adapt the size of the triangle mesh. [IkJpS03] and [AB10] presented improvements to this algorithm.

DOI: 10.2312/conf/EG2012/short/029-032

while $\varepsilon_c < \varepsilon_t$ **do**
    $p = \text{random Sample}(\mathbb{P})$
    $v_w = \arg\min(\|p - v\|)$
    $v'_w = v_w + \alpha_w (p - v_w)$
    for all $v_g \in$ 1-ring(v_w) **do**
        $v'_g = v_g + \alpha_g \mathcal{L}_t (v_g)$
    if vertex removal is necessary **then**
        remove $\mathbb{V}_x = \{v_x \in \mathbb{V} \mid \text{activity}(v_x) \leq \varepsilon_l\}$
    if vertex split is necessary **then**
        split $v_a = \arg\max(\text{activity}(v))$
end

Figure 2: *Learning algorithm*

A progressive mesh representation [Hop96] of a triangle mesh allows for compact storage and transfer, level of detail and incremental refinement in the visualization process. However, until now, a progressive mesh representation could only be created by post processing.

3. Technique

In the AVIGLE project an unstructured set of points $\mathbb{P}$ representing visible geometry is extracted from aerial images [RSH11]. A 2-manifold triangle mesh $M = (\mathbb{V}, \mathbb{T})$ consisting of a set of vertices $\mathbb{V}$ and a set of triangles $\mathbb{T}$ can be reconstructed from $\mathbb{P}$ using a technique based on improved Growing Cell Structures (GCS) [IkJpS03]. Finally M is stored in a database for subsequent visualisation.

3.1. Growing Cell Structures

In GCS $\mathbb{V}$ can be used as the set of neurons of the artificial neural network learning the shape of M. Vertex and neuron are used synonymously throughout this paper. For simplicity let $\mathbb{P}, \mathbb{V} \subset \mathbb{R}^3$. Nevertheless, this can be extended to $\mathbb{P}, \mathbb{V} \subset \mathbb{R}^n$ by including color, texture coordinates etc. A tetrahedron is used as the initial topology M_0 of the network.

Learning (cf. fig. 2) terminates if a certain quality criterion ε_c reaches a predefined threshold ε_t, e. g., the number of vertices or the average distance of all $v \in \mathbb{V}$ to all $p \in \mathbb{P}$. In each learning step for a randomly chosen signal $p \in \mathbb{P}$ the neuron $v_w \in \mathbb{V}$ closest to p with respect to the Euclidean distance $\|p - v_w\|$ is determined and moved towards p according to a learning rate α_w. To prevent fold-overs and convergence towards local minima Laplacian smoothing $\mathcal{L}_t$ weighted by α_g is applied to the neighbors v_g of v_w.

After a certain number of iterations a set of inactive neurons $\mathbb{V}_x$ can be determined that have to be removed from the network by a half-edge collapse. Furthermore, the most active neuron v_a can be determined representing a region with too few neurons. Thus, an additional neuron has to be added by a vertex split. For the mesh to grow vertex addition has to be triggered more often than vertex removal.

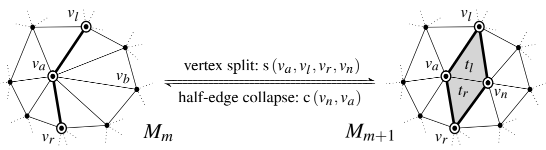

Figure 3: *Vertex split* s *and half-edge collapse* c

3.2. Vertex Split

A new vertex v_n and two triangles t_l and t_r are added to M by a vertex split s (cf. fig. 3). Let v_b denote the end vertex of the longest edge incident to v_a, then for simplicity $v_n = \frac{v_a + v_b}{2}$, but any position inside the triangles to the right of the edge-sequence v_r, v_a, v_l and incident to v_a will be suitable. This operation preserves the topological type of M [HDD*93].

For any mesh $M_m = (\mathbb{V}_m, \mathbb{T}_m)$ a vertex split $\mathrm{s}(v_a, v_l, v_r, v_n)$ is unambiguously defined by $v_a, v_l, v_r \in \mathbb{V}_m$ and the new vertex v_n as a transformation from M_m to a new mesh $M_{m+1} = (\mathbb{V}_m \cup \{v_n\}, \mathbb{T}_m \cup \{t_l, t_r\})$ [Hop96]. Having v_a and v_b then v_l and v_r are determined in such a way that $(|v_a| - |v_n|)^2$ is minimized after the split, with $|v|$ denoting the valence of v.

3.3. Half-Edge Collapse

An inactive neuron v_x and two triangles t_l, t_r can be removed from M by a half-edge collapse (cf. fig. 3). This operation is not guaranteed to preserve the topological type of M. Thus, removal of some $v_x \in \mathbb{V}_x$ must be deferred until the respective collapse has become legal [HDD*93].

For any mesh $M_m = (\mathbb{V}_m, \mathbb{T}_m)$ a half-edge collapse $\mathrm{c}(v_n, v_a)$ is unambiguously defined by $v_n, v_a \in \mathbb{V}_m$ as a transformation from M_m to a new mesh $M_{m+1} = (\mathbb{V}_m \setminus \{v_n\}, \mathbb{T}_m \setminus \{t_l, t_r\})$, if c is legal [Hop96]. Obviously split and collapse are invertible, so that $\mathrm{s}^{-1}(v_a, v_l, v_r, v_n) = \mathrm{c}(v_n, v_a)$ and vice versa.

3.4. Progressive Mesh

Hoppe presented a bottom-up method to find a sequence $(o) = (\mathrm{s}_0, \mathrm{s}_1, \ldots, \mathrm{s}_{n-1})$ and a coarse triangle mesh M_0 for any detailed triangle mesh M_n in such a way that M_n can be reconstructed from M_0 by applying (o): $M_0 \xrightarrow{\mathrm{s}_0} M_1 \xrightarrow{\mathrm{s}_1} \ldots \xrightarrow{\mathrm{s}_{n-1}} M_n$. Hoppe defined the tuple $(M_0, (o))$ as a progressive mesh (PM) representation of M_n [Hop96] – a compact way to store any triangle mesh including level of detail.

3.5. New Vertex Removal

Using GCS to reconstruct M'_n leads to a sequence $(o') = (o'_0, o'_1, \ldots, o'_{n-1})$ consisting of both, split and collapse operations. Thus, $\left(M'_0, (o')\right)$ is no PM representation of M'_n, but very similar to such a representation. Level of detail (LOD)

Figure 4: *Median running times of the new vertex removal (solid) compared to the classic half-edge collapse (dashed).*

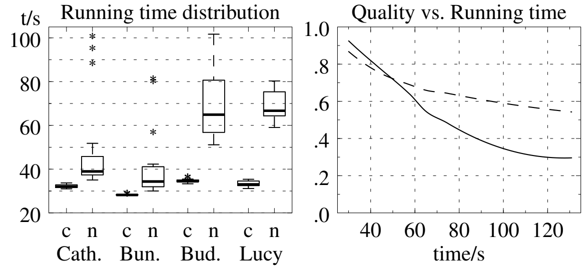

Figure 5: *Reconstruction of 10,000 vertices. Left: Running times of algorithm using half-edge collapse (c) and new vertex removal (n). Right: Quality vs. running time for a reconstruction of 10,000 vertices in terms of ratio of valence 5-7 vertices (dashed) and Delaunay triangles (solid) for the combined results of all four meshes.*

can still be achieved by subsequently applying the inverted split and collapse operations of (o') to $M'_n = (\mathbb{V}'_n, \mathbb{T}'_n)$. But during the LOD steps towards the coarser mesh, inverting any collapse operation $c'_i \in (o')$ increases the number of vertices. Since these superfluous vertices do not belong to $\mathbb{V}'_n$, their position is undefined. To overcome this, we replace a half-edge collapse in GCS by a more general vertex removal in such a way that the neural network learns a PM representation, iteratively.

Until the first v_x has to be removed from a mesh M_m, only vertex splits have been performed and thus the corresponding operation sequence $(o) = (\mathrm{s}_0, \mathrm{s}_1, \ldots, \mathrm{s}_k, \ldots, \mathrm{s}_{m-1})$ has been stored. Instead of adding another operation to (o) that removes v_x we want to find a sequence (o') not producing v_x from M_0 in the first place, i. e., erase v_x from history.

Let s_k be the last operation in which v_x is involved either as v_a or v_n, then (o) can be rewritten as $(o) = ((\underline{o}), \mathrm{s}_k, (\overline{o}))$. To remove v_x from M_m the inverted sequence $(\mathrm{s}_k, (\overline{o}))^{-1} = \left(\mathrm{s}_{m-1}^{-1}, \mathrm{s}_{m-2}^{-1}, \ldots, \mathrm{s}_{k+1}^{-1}, \mathrm{s}_k^{-1}\right)$ is applied to M_m simplifying it to M_k. If v_x was involved as v_n in s_k then $v_x \notin \mathbb{V}_k$, since v_x was removed by applying s_k^{-1} . Thus, v_x can be deleted.

If on the other hand v_x was involved as v_a in s_k then $v_x \in \mathbb{V}_k$, so it is still connected in M_k. Therefore, v_x must not be deleted. Instead v_n can be deleted since $v_n \notin \mathbb{V}_k$ after applying s_k^{-1}. The attributes assigned to v_n must be preserved by transferring them to v_x, the neighbor of v_n in M_{k+1}. So, if v_x was involved as v_a in s_k, removal of v_x is replaced by removal of v_n. As a consequence every reference to v_n in the operations of $(\overline{o})$ has to be replaced by a reference to v_x.

Special care must also be taken if no s_k exists, i.e., $v_x \in \mathbb{V}_0$ has not yet been involved as v_a or v_n in any split operation. In that case (o) can be rewritten as $(o) = (\mathrm{s}_0, (\overline{o}))$, and $(o)^{-1}$ must be applied to M_m simplifying it to M_0 with v_x still connected. As in the previous case, attributes of another vertex (v_n of s_0) must be transferred to v_x, the respective references in $(\overline{o})$ have to be replaced, and the other vertex can be deleted.

Finally, the sequence $(\overline{o})$ has to be reapplied to M_k to get back a detailed mesh M'_m reusing the original attributes from $\mathbb{V}_m$ with one caveat: Since a vertex has been deleted, v_l or v_r might have become invalid for some $\mathrm{s}_i = \mathrm{s}(v_a, v_l, v_r, v_n) \in (\overline{o})$. Thus, some s_i have to be repaired. A straight-forward repair is to identify a neighboring vertex v_b of v_a minimizing the angle $\angle v_b v_a v_n$. Thus, v_n is near to the edge v_a, v_b. Finally, v_l and v_r can be determined as in subsection 3.2.

Let $M_c = (\mathbb{V}_c, \mathbb{T}_c)$ be the mesh produced by removing v_x from M_m by a half-edge collapse and $M'_m = (\mathbb{V}'_m, \mathbb{T}'_m)$ be the mesh produced by removing v_x from M_m with the new technique. Then $|\mathbb{V}'_m| = |\mathbb{V}_c|$ and $|\mathbb{T}'_m| = |\mathbb{T}_c|$ with $|\cdot|$ denoting the number of elements in a set. Furthermore, the learned attributes associated to the vertices in M'_m are the same as those of M_c. So, M'_m is similar to M_c and the sequence $(o') = ((\underline{o}), (\overline{o}))$ is the one we had to find. By construction all operations of (o') are splits, and thus the tuple $(M_0, (o'))$ is a progressive mesh representation of M'_m.

4. Results

The presented algorithm was tested in a single thread on an Apple MacBook Pro, 2.66 GHz Intel i7 CPU, 8 GB RAM. Meshes were reconstructed from four point clouds: Bunny, Happy Buddha and Lucy, obtained from the Stanford scanning repository, and a synthetic model of the cathedral of Paderborn, Germany. Each reconstruction was tested with 15 different random seeds. To address runtime effects each test was repeated four times to determine mean reconstruction times and quality measures for the respective seed.

4.1. Reconstruction Times

The proposed technique does not impose severe running time overhead for the cathedral and Bunny (cf. fig. 4). When reconstructing a triangle mesh of 10,000 vertices, median running times increased from 32.0 s using half-edge collapse to 38.9 s using new vertex removal for the cathedral and from 28.2 s to 34.3 s for Bunny. When reconstructing meshes with very fine details, median running times increased from 34.4 s to 64.9 s for Buddha and from 33.0 s to 66.7 s for Lucy.

The new technique leads to a greater variation of running

Table 1: *Mesh Quality after reconstructing 10,000 vertices*

Mesh	Valence 5-7 old	Valence 5-7 new	Delaunay old	Delaunay new	Hausd. Dist. old	Hausd. Dist. new
Cathed.	79 %	77 %	86 %	82 %	1%	3 %
Bunny	87 %	84 %	91 %	89 %	3 %	4 %
Buddha	73 %	66 %	80 %	57 %	3 %	2 %
Lucy	78 %	64 %	85 %	50 %	4 %	7 %

times for different random seeds (cf. fig. 5 left). Nevertheless, running times for Cathedral, Bunny and Lucy do not deviate as much from the respective median time than the running times for Buddha do – a mesh not homeomorphic to the others.

4.2. Mesh Quality

Quality of the reconstructed meshes was evaluated in terms of regularity, i. e., the ratio of vertices with valence 5 to 7 and the ratio of triangles fulfilling the Delaunay criterion. The new technique reduces mesh quality of the reconstructed mesh only slightly for the cathedral and Bunny (cf. tab. 1) and moderately for Buddha and Lucy.

Hausdorff distances were determined between each reconstruction and the respective triangle mesh from which the point cloud was generated. With both, the old and our new technique, similarly small Hausorff distances were achieved (cf. tab. 1, normalized to the diagonal of the bounding box).

During vertex removal, sequences $(s_k, (\overline{o}))$ need to be reverted and reapplied. Short running times for a fixed number of reconstructed vertices indicate that those sequences are short whereas long running times indicate that those sequences are long. From fig. 5 right it can be seen that quality gets reduced for longer sequences. However, short sequences $(s_k, (\overline{o}))$ exist for many different random seeds resulting in good mesh quality. A visual example of the quality of the reconstructed mesh is shown on the far left of fig. 1.

4.3. Level of Detail

At any time, the reconstructed mesh can be reduced to a coarser mesh using the automatically learned level of detail (LOD) steps while preserving the shape of the reconstructed object with good visual quality. Fig. 1 shows an LOD sequence for Stanford Bunny reducing a mesh of 20,000 triangles to a coarser mesh of 312 triangles.

4.4. Memory Requirements

In a reconstructed triangle mesh $|\mathbb{T}| \approx 2|\mathbb{V}|$, $|(o)| \approx |\mathbb{V}|$. Assuming that an index to a vertex consumes the same amount of memory as a vertex' component, an indexed triangle list for the reconstructed 3D mesh M needs $3|\mathbb{V}| + 3 \cdot 2|\mathbb{V}|$ units of memory. A progressive mesh representation of M needs only 66 % of that, i.e., $3|\mathbb{V}| + 3|\mathbb{V}|$ units of memory.

5. Conclusion and Future Work

We have successfully modified surface reconstruction based on Growing Cell Structures in such a way that the reconstructed triangle mesh is directly represented as a progressive mesh. The running time overhead of the new technique is related to the complexity of the final mesh but typically low. The new technique allows for level of detail and compact storage and transfer while preserving a good mesh quality.

We plan to use the progressive mesh representation for transfer and visualization in the processing pipeline of the AVIGLE project. Furthermore we will investigate how to utilize the integrated level of detail in the triangle mesh rendering step to improve the rendering times.

Finally we plan to integrate recent improvements to GCS like [AB10] into our new system.

Acknowledgements

The project AVIGLE is part of the Hightech.NRW initiative funded by the Ministry of Innovation, Science and Research of the German State of North Rhine-Westphalia.

References

[AB10] ANNUTH H., BOHN C.-A.: Smart growing cells. In *Proc. of the International Conference on Neural Computation* (Valencia, Spain, 2010). 1, 4

[Fri93] FRITZKE B.: *Growing Cell Structures – A Self-organizing Network for Unsupervised and Supervised Learning*. Tech. Rep. TR-93-026, International Computer Science Institute, Berkeley, CA, USA, May 1993. 1

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proc. of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 19–26. 2

[Hop96] HOPPE H.: Progressive meshes. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 99–108. 2

[IkJpS03] IVRISSIMTZIS I. P., K. JEONG W., P. SEIDEL H.: Using growing cell structures for surface reconstruction. In *Shape Modeling International 03, Conf. Proc.* (2003), pp. 78–86. 1, 2

[RGW*10] ROHDE S., GODDEMEIER N., WIETFELD C., STEINICKE F., HINRICHS K., OSTERMANN T., HOLSTEN J., MOORMANN D.: Avigle: A system of systems concept for an avionic digital service platform based on micro unmanned aerial vehicles. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on* (oct. 2010), pp. 459 –466. 1

[RSH11] ROTERS J., STEINICKE F., HINRICHS K. H.: Quasi-real-time 3d reconstruction from low-altitude aerial images. In *Proc. of the 40th Urban Data Management Society Symposium* (sep 2011), vol. 40 of *UDMS Annual 2011*, CRC Press/Balkema, pp. 231–241. 1, 2

[SFS*11] STROTHOFF S., FELDMANN D., STEINICKE F., VIERJAHN T., MOSTAFAWY S.: Interactive generation of virtual environments using muavs. In *ISVRI 2011: Proc. of International Symposium on VR innovation* (2011), pp. 89 – 96. 1