Local Versus Global Triangulations

Lars Linsen Hartmut Prautzsch

Universität Karlsruhe, Germany

Abstract

Free form surfaces are commonly represented by triangular or quadrilateral meshes. Often these meshes are obtained from unorganized point sets sampled from some object's surface.

We show that local rather than global triangulations of point sets are equally well suited for object representations and that the local triangulations proposed in this paper may even lead to fast triangulation routines.

1. Introduction

For many computer-aided applications in manufacturing, geography, medicine, design etc. it is necessary to reconstruct three-dimensional objects. With todays scanning methods it is easy to obtain large dense sets of points on a given object surface. We will call such sets point clouds.

To obtain a continuous surface representation various methods have been developed to generate triangular meshes from point clouds. Given a triangular net standard techniques can be used to visualize the underlying object, to reduce the amount of data and/or reduce noise due to the scanning process, and to modify and edit the object.

In Section 3 of this paper we show that very local triangulations suffice to visualize an object given by a point cloud. Further, in Section 4 we present ideas for a fast triangulation routine based on our local triangulation. In Section 5 we develop a smoothing operator and in Section 6 we evaluate our method by comparing it to related work.

2. Related work

In the nineties various approaches were presented to generate triangular meshes out of point clouds. The algorithms are based on spatial subdivision (e.g. [1, 2, 3, 5, 6, 8, 10, 14, 25]), distance functions (e.g. [6, 14]), warping (e.g. [1]), and incremental surface-increase (e.g. [4, 5, 10, 19]). A survey is given in [20].

To obtain high accuracy and resistance against error distortion the measuring techniques nowadays produce up to many millions of sampling points. Thus, usually point clouds are downsampled before a surface reconstruction algorithm is applied. For the data reduction some heuristics like grouping of points are used [9, 24, 29, 31].

Smoothing operators for triangular meshes were developed in [7, 12, 15, 28].

Already in 1992 Szeliski and Tonnesen presented oriented particles [27]. These are point clouds, where each point has an orientation, compatible with the normal direction of the represented surface. To force oriented particles to group themselves into surface-like arrangements, they apply potential energies. For rendering purposes they use axes, discs, or after triangular mesh generations wireframes and shaded triangulations.

3. Visualization

In particle animations of fire, fog, water, etc. point clouds are visualized by drawing only all the points [22, 26]. However, for a solid object this simple technique does not lead to a realistic plastic impression as illustrated in Figure 4(a). Raycasting gives better results, but the point cloud has to be rather dense and for frame rates of 1-2 fps approximately one hour of preprocessing is required [11, 21].

In our method we compute for each point $\mathbf{p}$ a k-neighbourhood consisting of k pointers to points $\mathbf{p}_1, \ldots, \mathbf{p}_k$ of the cloud close to $\mathbf{p}$ as described further below. The neighbours $\mathbf{p}_i$ are determined such that the k triangles $\mathbf{p}\mathbf{p}_i\mathbf{p}_{i+1}$ form a fan that approximates a "disc", i.e. neighbourhood, of $\mathbf{p}$ on the surface represented by the point cloud.

With $k = 6$ all k-neighbourhoods take about the same storage as a triangular mesh. However, since the triangle fans do not form one coherent mesh, they are much faster to compute, see Section 6 for a comparison.

To determine a k-neighbourhood of a point $\mathbf{p}$ we determine the k nearest neighbours $\mathbf{p}_1, \ldots, \mathbf{p}_k$, compute the plane

P with the least sum of squared distances to $\mathbf{p}, \mathbf{p}_1, \ldots, \mathbf{p}_k$ and project all points into P. Then we sort, i.e. permute the indices of $\mathbf{p}_1, \ldots, \mathbf{p}_k$ such that their projections $\mathbf{q}_1, \ldots, \mathbf{q}_k$ form increasing angles $\varphi_i = \angle \mathbf{q}_1 \mathbf{q} \mathbf{q}_i$ with the projection $\mathbf{q}$ of $\mathbf{p}$. In this order the points $\mathbf{p}_i$ form a triangle fan or k-neighbourhood of $\mathbf{p}$, see Figure 1.

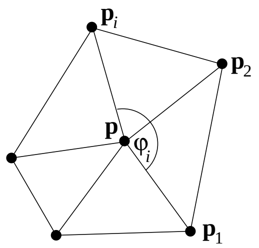


Figure 1: *A k-neighbourhood for $k = 5$.*

If the point density varies sharply around $\mathbf{p}$, then the neighbourhood may not enclose $\mathbf{p}$, see Figure 2(a). Therefore if $\nabla\varphi_i = \varphi_i - \varphi_{i-1} > 90^\circ$, we replace $\mathbf{q}_k$ by the

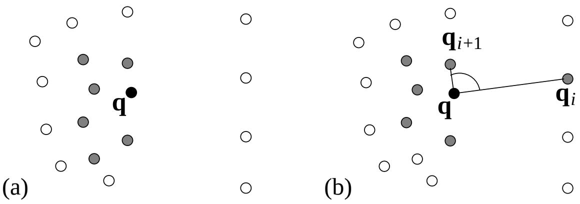


Figure 2: *Necessity of the angle criterion for choosing the neighbours of a point* $\mathbf{p}$.

$(k+1)$st neighbour and if necessary by further next neighbours till the angle criterion $\nabla\varphi_i \leq 90^\circ$ is met or a certain threshold number of replacements has been reached.

Along sharp edges the best fitting plane may be normal to the surface, see Figure 3. Then the angle criterion could not

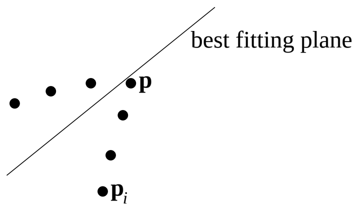


Figure 3: *Best fitting plane for an edge point.*

be satisfied. In such a case we flip the fitting plane around $\mathbf{p}_{i-1}\mathbf{p}_i$ by 90° and try again to build the neighbourhood.

If the angle criterion can still not be met, then we assume that $\mathbf{p}_{i-1}$, $\mathbf{p}$, $\mathbf{p}_i$ lie on the boundary of the surface.

To visualize the object represented by the point cloud we render the cloud of all triangle fans, see Figure 5(a). Although the fan cloud does not form a coherent triangular mesh it is so close to it that no artefacts can be seen in a shaded image, see Figure 4(c) and the next section for a detailed discussion.

To use Gouraud or Phong shading we associate with each point the normal of the best fitting plane which has been computed above. A consistent orientation of the normals can be computed with the minimal spanning tree described in [14]. A result is shown is Figure 4(c) for $k = 8$.

Rendering discs or similar surface pieces instead of our triangle fans as done in [27, 32] does not lead to a comparable well surface visualization, see Figure 4(b). In [23] Rusinkiewicz and Levoy develop their QSplat approach to overcome this problem. However, for good results this approach needs a triangular mesh. Moreover for best results they resort to antialiasing techniques which further raise the time complexity of their method.

4. Global triangulation via fan cloud

A fan cloud consists of $k \cdot n$ triangles whereas a triangular mesh consists of only $2n$ triangles. However, there are many duplicates in a fan cloud. Without them a fan cloud consists of about $2.5n$ triangles.

Further there are quadrilaterals covered by three or four triangles of a fan cloud, i.e. by one or two superfluous triangles. Removing these superfluous triangles reduces the number of triangles to about $2.1n$ in Figure 5(b). This reduced fan cloud can be viewed as several different triangulations of the point cloud on top of each other.

In fact it is possible to obtain a triangular net from the reduced fan cloud. We can simply grow a triangular net by successively adding on triangles. The result is illustrated in Figure 5(c). It has few $(0,01\%)$ selfoverlaps since we neglected any geometric aspects and based the construction only on topological aspects. Since by construction there are no edges with more than 3 triangles, the overlaps correspond to holes that fold back onto themselves as illustrated in Figure 6 by heavy lines.

This triangulation is fast. Its running time is given in Table 1.

5. Smoothing

For triangular nets several smoothing operators are known. Kobbelt [15] presents a discrete Laplacian smoothing operator that moves any point $\mathbf{p}$ to the centroid $\mathbf{q}$ of its neighbours. This operator can also be used to smooth point clouds with our k-neighbourhoods.

Since the Laplacian smoothing operator shrinks the object, Taubin [28] proposes from a signal processing point of

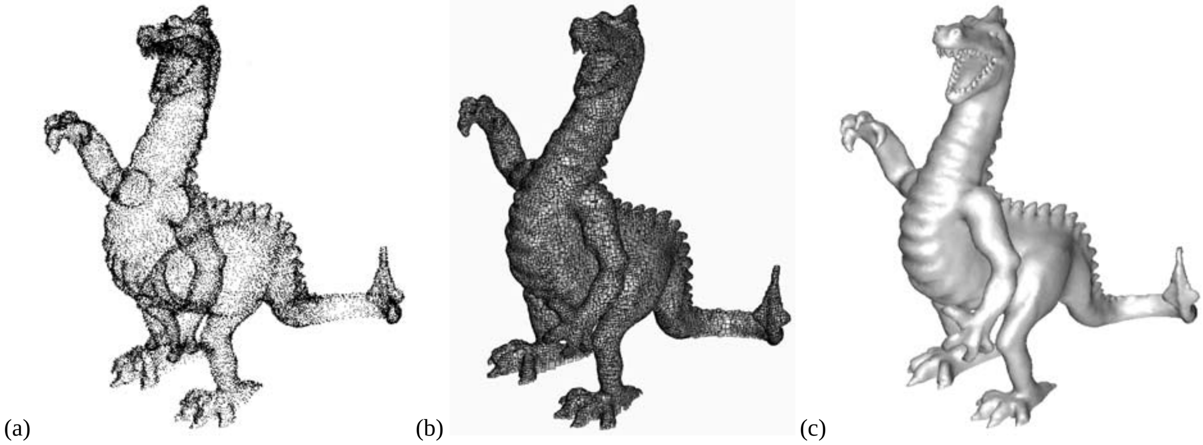


Figure 4: *(a) Plotting the points of a point cloud. (b) Drawing an accumulation of pieces of the represented surface. (c) Visualization of the fan cloud.*

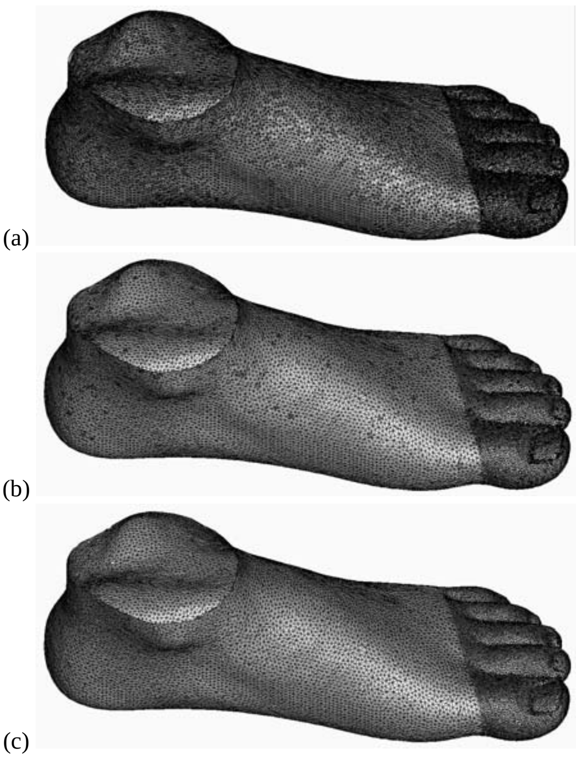


Figure 5: *From a fan cloud to a triangular net.*

view to use the operator

$$\mathbf{p} := (1-\lambda)\mathbf{p} + \lambda\mathbf{q}$$

alternately with positive and negative λ's. This causes the object to alternately shrink and grow.

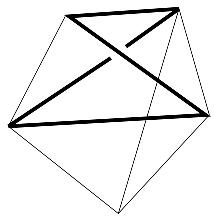


Figure 6: *Overlapped folded hole.*

As a side effect these smoothing operators also equalize the shape of the triangles. This can affect texture and colour. Therefore Guskov et al. [13] develop a smoothing operator that takes the geometry into account and approximately preserves the shape of the triangles.

Their smoothing operator gives a mesh with a minimal sum

$$E = \sum_e (D_e^2)^2$$

of squared second differences

$$D_e^2 = \sum_{x \in \{i,j,k,l\}} c_{e,x}\mathbf{p}_x \ ,$$

where with the notation given in Figure 7

$$\begin{aligned} c_{e,i} &= \frac{d_{jk}}{A_{ijk}A_{lkj}} A_{lkj} & \qquad c_{e,j} &= -\frac{d_{jk}}{A_{ijk}A_{lkj}} A_{kil} \\ c_{e,k} &= -\frac{d_{jk}}{A_{ijk}A_{lkj}} A_{jli} & \qquad c_{e,l} &= \frac{d_{jk}}{A_{ijk}A_{lkj}} A_{ijk} \end{aligned}$$

$d_{jk} = ||\mathbf{p}_j - \mathbf{p}_k||_2$ and A_{xyz} denotes the signed area of the triangle $\mathbf{p}_x\mathbf{p}_y\mathbf{p}_z$.

The associated smoothing operator is

$$\mathbf{p}_i := \sum_j \omega_{ij}\mathbf{p}_j$$

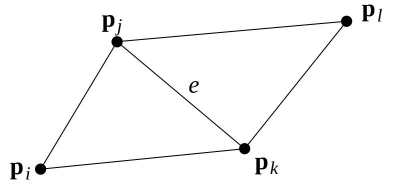


Figure 7: *The support of* D_e^2.

with

$$\omega_{ij} = -\frac{\sum_e c_{e,i} c_{e,j}}{\sum_e c_{e,i}^2} \,,$$

where the numerator is summed over all edges e, whose associated rhombus (cf. Figure 7) contains $\mathbf{p}_i$ and $\mathbf{p}_j$, and the denominator is summed over all edges e, that contribute to the neighbourhood (triangle fan) of $\mathbf{p}_i$.

The support of the smoothing operator has the form of a star and is larger than a k-neighbourhood. Therefore we came up with the modified smoothing operator

$$\mathbf{p} := \mathbf{p} + \lambda \Diamond \mathbf{p} \,,$$

where

$$\Diamond \mathbf{p} = \sum_{i=1}^{k} -\frac{\omega_i}{\omega_0} \mathbf{q}_i - \mathbf{p} \qquad (1)$$

and the values ω_i are determined by

$$\sum_{i=1}^{k} \Diamond_{\mathbf{p}\mathbf{q}_i} = \sum_{i=1}^{k} \omega_i \mathbf{q}_i + \omega_0 \mathbf{p}$$

with

$$\Diamond_{\mathbf{p}_j \mathbf{p}_k} = \sum_{x \in \{i,j,k,l\}} c_x \mathbf{p}_x$$

and the coefficients

$$\begin{aligned} c_i &= \frac{d_{jk}}{d_{jl}} A_{lkj} & \qquad c_j &= -\frac{d_{jk}}{d_{il}} A_{kil} \\ c_k &= -\frac{d_{jk}}{d_{il}} A_{jli} & \qquad c_l &= \frac{d_{jk}}{d_{il}} A_{ijk} \end{aligned}$$

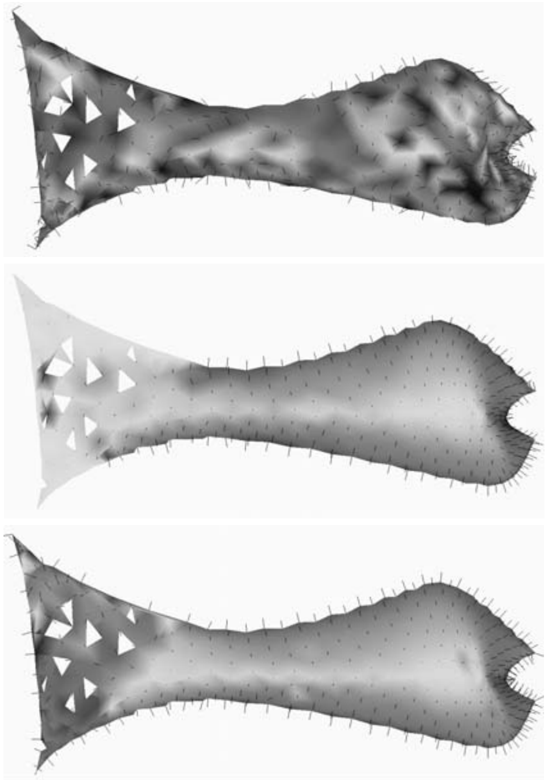
Note that $\Diamond$ is Kobbelt's discrete Laplace operator if all triangles of the k-neighbourhood are congruent.

Using Taubin's idea we alternately use a positive and a negative λ to avoid a shrinkage of the object.

Figure 8 shows a point cloud before and after smoothing it with our operator for $k = 8$.

Figure 9 illustrates the effects of the Laplacian and our smoothing operator applied to a triangular mesh. While the Laplacian smoothing operator changes the triangle shapes very obviously, our smoothing does not do so.

The reason for the behaviour is that the normalized vector $\Diamond \mathbf{p}$ is a very good approximation to the surface normal. Although we found this operator by trial and error it gives a much better approximation of the surface normal at $\mathbf{p}$ than the average normal of the triangle fan at $\mathbf{p}$ or the normal of the best fitting plane, as illustrated in Figure 10.



(a) (b) (c)

Figure 10: *Surface with average normals of triangle fans (a), normals of the best fitting planes (b), and the normalized vectors of our smoothing operator (c).*

6. Discussion

In this section we discuss the advantages and disadvantages of local triangulations in comparison with triangular meshes.

If triangular meshes are used for surface reconstruction, a mesh needs to be generated before operations like smoothing can be executed. Usually large point clouds are reduced before a mesh generation [9, 24, 29, 31]. In contrast to this approach we can smooth point clouds immediately. Thus the whole sample information is used and a reduction can be based on the smoothed surface geometry and colour distribution.

Various methods were developed for the triangular mesh generation. Many of them use Delaunay tetrahedrizations, which have a time-complexity of $O(n^2)$ for n points. In [30] it is shown that the computation of the k-nearest neighbours can be done in $O(n \log n)$ by using a preprocessing step. Thus point clouds can be visualized in $O(n \log n)$.

The profits in time-complexity become clearer when we

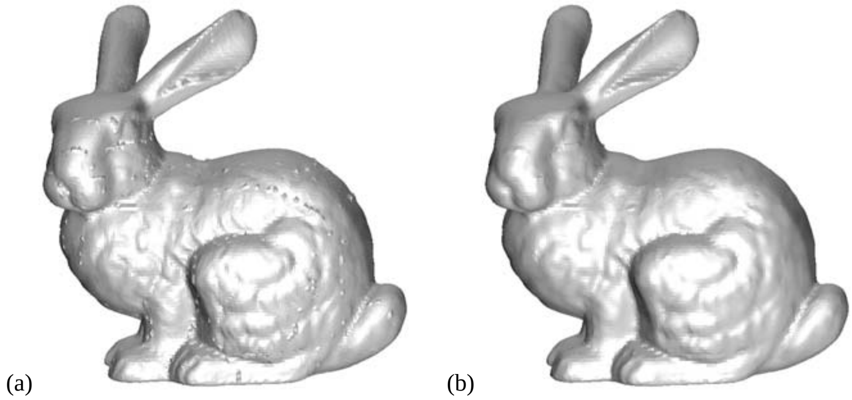


Figure 8: *Error elimination by applying our smoothing operator.*

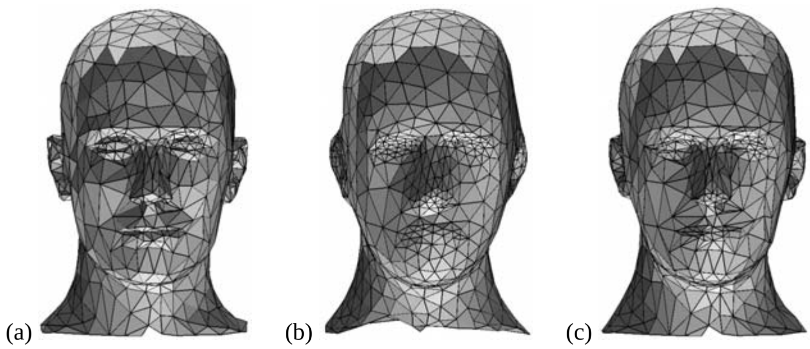


Figure 9: *The Laplace-operator applied to a triangular mesh (a) changes the triangle shapes (b), in contrast to our smoothing operator (c).*

look at the overall running time, see Table 1. These examples show that with our method the running times are reduced from minutes to seconds.

Newer approaches [4, 10] try to cut down on the high costs for a mesh generation by making extra assumptions on the sampling rate. They achieve lower running times at the expense of generality.

In our experience visualizing a triangular mesh versus a point cloud with our neighbourhoods leads to results with equal quality.

If measuring techniques scan the object from different viewpoints, many merging steps like the one shown in Figure 11 have to be executed. Again for triangular mesh representations sophisticated analyses of the object's shape are required to know exactly, where and how the samples can be merged. Numerical problems may occur. For point clouds the points of the multiple range images are simply stuck into one single point cloud. Because of calibration errors we apply our smoothing operator to the regions, where the samples overlap. If desired the overlapping regions can be reduced. A result for point clouds is shown in Figure 11(c).

Furthermore, our local triangulation is also very well suited for many other geometric operations, for example reduction, multiresolution modelling, refinement, and others, see [18].

References

1. Maria-Elena Algorri, Francis Schmitt: *Surface reconstruction from unstructured 3d data.* Computer Graphics Forum, Vol. 15 (1), 47 - 60, 1996.
2. Marco Attene, Michela Spagnuolo: *Automatic surface reconstruction from point sets in space.* Computer Graphics Forum, Vol. 19 (3), 457 - 466, 2000.
3. Chandrajit Bajaj, Fausto Bernardini, Guoliang Xu: *Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans.* SIGGRAPH '95 Proceedings, 109 - 118, 1995.
4. Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Clàudio Silva, Gabriel Taubin: *The Ball-Pivoting Algorithm for Surface Reconstruction.* IEEE Transac-

(a) (b) (c)

Figure 11: *Merging two partly overlapping samples to build one single point cloud.*

method	# points	computer	time
Delaunay [19]	1248 2600 48921	SGI Indigo2 Extreme	45 s 1 min 133 min
Algorri, Schmitt [1]	45233	Sun Sparc Station 40 MHz	18 min 19 s
Bajaj et al. [3]	9223	SGI Indigo2	10 min
Edelsbrunner, Mücke [8]	9600 10000 10088 15000	SGI 50 MHz, MIPS R4000	39 min 16 min 26 min 27 min
Hoppe et al. [14]	18224	20 MIPS Workstation	31 min 15 s
Mencl, Müller [19]	1248 2600 48921	SGI Indigo2 Extreme	2 min 5 min 193 min
Shimada [25]	1000	IBM RS/6000	40 s
Bernardini et al. [4]	11000 361000	PC with Pentium II Xeon 450MHz	3 s 7 s
Gopi et al. [10]	34834 83034	SGI Onyx	19 s 44 s
Local triangulation	47109 100001 160940	SGI Indigo2 Extreme Sun Ultra30 PC with Athlon K7 800MHz	59 s 20 s 7 s 24 s 59 s
Global triangulation via fan cloud	20021 35948	PC with Athlon K7 800MHz	6s 16s

Table 1: *Comparing the running time of triangular mesh generations with the visualization of point clouds.*

tions on Visualization and Computer Graphics, Vol. 5 (4), 349 - 359, 1999.

5. Jean-Daniel Boissonat: *Geometric Structures for Three-Dimensional Shape Representation.* ACM Transactions on Graphics, 266 - 286, 1984.

6. Brian Curless, Marc Levoy: *A Volumetric Method for Building Complex Models from Range Images.* SIGGRAPH ’96, New Orleans, LA, 4-9 August 1996.

7. Mathieu Desbrun, Mark Meyer, Peter Schröder, Alan H. Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow.* Proceedings of SIGGRAPH 1999, 1999.

8. H. Edelsbrunner, E.P. Mücke: *Threedimensional alpha shapes.* ACM Transactions on Computer Graphics, Vol. 13 (1), 43 - 72, 1994.

9. Michael S. Floater, A. Iske: *Thinning algorithms for scattered data interpolation.* BIT Numerical Mathematics, Vol. 38 (4), 705 - 720, 1998.

10. M. Gopi, S. Krishnan, C.T. Silva: *Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation.* Computer Graphics Forum, Vol. 19 (3), 2000.

11. J. P. Grossman, William J. Dally: *Point Sample Rendering.* Rendering Techniques ’98, Springer, Wien, 181 - 192, 1998.

12. Igor Guskov: *Multivariate Subdivision Schemes and Divided Differences.* Preprint, Princeton University, 1998.

13. Igor Guskov, Wim Sweldens, Peter Schröder: *Multiresolution Signal Processing for Meshes.* Proceedings of SIGGRAPH 99, 1999.

14. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle: *Surface Reconstruction from Unorganized Points.* Computer Graphics, Vol. 26, 71 - 78, 1992.

15. Leif Kobbelt: *Iterative Erzeugung glatter Interpolanten.* Ph.D. thesis, Universität Karlsruhe, Verlag Shaker, Aachen, 1995.

16. Leif Kobbelt, Swen Campagna, Jens Vorsatz, Hans-Peter Seidel: *Interactive Multi-Resolution Modeling on Arbitrary Meshes.* SIGGRAPH 98 proceedings, 1998.

17. Lars Linsen: *Point cloud representation.* Technical report No. 2001-3, Fakultät für Informatik, Universität Karlsruhe, 2001.

18. Lars Linsen: *Oberflächenrepräsentation durch Punktwolken.* Dissertation, Universität Karlsruhe, 2001.

19. Robert Mencl, Heinrich Müller: *Graph-Based Surface Reconstruction Using Structures in Scattered Point Sets.* Proceedings of Computer Graphics International '98, Hannover, 1998.

20. Robert Mencl, Heinrich Müller: *Interpolation and Approximation of Surfaces from Three-Dimensional Scattered Data Points.* State of the Art Report for EUROGRAPHICS '98, Lisbon, 1998.

21. Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, Markus Gross: *Surfels: Surface Elements as Rendering Primitives.* Proceedings of SIGGRAPH 2000, 2000.

22. William T. Reeves: *Particle Systems - A Technique for Modelling a Class of Fuzzy Objects.* Computer Graphics, Vol. 17 (3), 359 - 376, 1983.

23. Szymon Rusinkiewicz, Marc Levoy: *QSplat: A Multiresolution Point Rendering System for Large Meshes.* Proceedings of SIGGRAPH '00, 2000.

24. Thomas Schreiber: *Clustering for data reduction and approximation.* International Conference on Computer Graphics and Visualization '93, St. Petersburg, Russia, 1993.

25. Kenji Shimada: *Bubble Mesh - Physically-based Triangulation Method.* IBM Research, 1996.

26. Karl Sims: *Particle Animation and Rendering Using Data Parallel Computation.* Computer Graphics, Vol. 24 (4), 405 - 413, 1990.

27. Richard Szeliski, David Tonnesen: *Surface Modeling with Oriented Particle Systems.* Computer Graphics, Vol. 26 (2), 185 - 194, 1992.

28. Gabriel Taubin: *A Signal Processing Approach To Fair Surface Design.* Computer Graphics Proceedings, Annual Conference Series, 351 - 358, 1995.

29. Peter Uray: *From 3D point clouds to surfaces and volumes.* Schriftenreihe der Österreichischen Computer Gesellschaft, Oldenbourg, München-Wien, 1997.

30. Marek Vanco, Guido Brunnett, Thomas Schreiber: *A hashing strategy for efficient k-nearest neighbors computation.* CGI'99, Canmore, Juni 1999.

31. Stefan Vogt: *Reduktion optischer 3D-Meßdaten mittels Multi-Resolution.* Ph.D. thesis, Universität Karlsruhe, Verlag Mainz, Wissenschaftsverlag, Aachen, 2000.

32. Andrew P. Witkin, Paul S. Heckbert: *Using Particle Systems to Sample and Control Implicite Surfaces.* Proceedings of SIGGRAPH '94, 1994.