Point cloud representation

Lars Linsen*

Universität Karlsruhe, Germany†

Abstract

Reconstructing a surface out of a three-dimensional set of points, which is obtained by sampling an object's boundary, is done by generating an arbitrary triangular mesh. Our approach is to obviate the computation of such a mesh connectivity and to represent the object's surface only by the point cloud.

We discuss how such a point cloud representation can be visualized and present processing steps like coarsifying and smoothing, which are important for dealing with the objects. Further we apply a multiresolution method to point cloud representations and use this technique as well as others for modelling purposes.

1 Introduction

In Computer Graphics surfaces of three-dimensional objects are represented by triangular meshes in general. They are gained by evaluating a known mathematical description or by interpolating or approximating a given set of unorganized points in the sense of surface reconstruction.

Many applications in manufacturing, medicine, geography, design, etc. require the scanning of rather complex three-dimensional objects (e.g. prototypes) to (re-)incorporate them into a computer-based or computer-aided processing. Thus, measuring techniques were evolved to easily produce a large amount of points lying on the objects' surfaces. Such a point set representing the boundary of a three-dimensional object we call a point cloud.

In the last decade various methods have been developed for generating triangular meshes out of point clouds. In this paper we present an approach, that avoids such a time-consuming generation and uses point cloud representations instead. This reduces the time complexity as well as the storage complexity. We discuss how point clouds can be visualized and be treated. Important treatments are the reduction of the amount of data, the elimination of error distortion, and the modification and editing of the represented objects.

Therefore Section 3 is dedicated to visualization, Section 4 to smoothing, Section 5 to multiresolution methods including reduction and refinement, and Section 6 to some modelling techniques. In Section 7 we discuss our approach and compare it to the related work stated in Section 2.

*llinsen@ira.uka.de

†http://i33www.ira.uka.de/~llinsen

2 Related work

In the nineties in the field of surface reconstruction 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, 9, 11, 15, 25]), distance functions (e.g. [6, 15]), warping (e.g. [1]), and incremental surface-increase (e.g. [4, 5, 11, 19]). A survey including other papers is given in [20].

To obtain a high accuracy and resistance against error distortion the measuring techniques nowadays produce up to many millions of sampling points. Thus the point clouds are downsampled before a surface reconstruction algorithm is applied. For the data reduction some heuristics like grouping of points are used [10, 24, 29, 31].

The results of the mesh generations are arbitrary triangular meshes. For such meshes smoothing operators were developed in [7, 13, 16, 28] as well as multiresolution methods in [14, 17]. The smoothing operators will be discussed later, the multiresolution methods for arbitrary triangular meshes are similar to our approach for point clouds. In [8] the authors propose multiresolution analysis on arbitrary triangular meshes by remeshing the given meshes to obtain meshes with subdivision connectivity.

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

Our first main task is to find a pleasant visualization for point cloud representations. Drawing only all the single points as done for particle animations of fire, fog, water, etc. (cf. [22, 26]) does not lead to a plastic impression of the three-dimensional object as illustrated in Figure 2(a). Raycasting gives better results, but the point cloud has to be rather dense and for frame rates of 1-2 fps about one hour of preprocessing is required [12, 21].

Since a point cloud representation does not contain any faces or surfaces, we must substitute the connectivity of triangular meshes by environments of each point. To find an appropriate environment, which represents a small piece of the object's surface, we assume that the nearest neighbours of a point contribute to that environment. In fact we only assume that the sampling rate is high enough, and in practice it is indeed. A proper sampling rate has a spatial frequency greater than or equal to the frequencies of the sampled surface.

But defining the environment of a point $\mathbf{p}$ by its k-nearest neighbours is not an optimal choice. In Figure 1(a) we give an example, where the k-nearest neighbours cover only half the environment.

Therefore we select k neighbours, that are distributed all around $\mathbf{p}$ by introducing an angle criterion. We compute the least squares best fitting plane P of $\mathbf{p}$ and its neighbours, project the neighbours onto P, and sort the projected neighbours around the projection of $\mathbf{p}$ in the plane P. Then we consider every

neighbour $\mathbf{q}_i$ and its successor $\mathbf{q}_{i+1}$ with respect to the sorting and claim that the angle $\angle \mathbf{q}_i \mathbf{p} \mathbf{q}_{i+1}$ fulfils the criterion, i.e. it must not exceed a certain limit, e.g. $\frac{\pi}{2}$ (cf. Figure 1(b)).

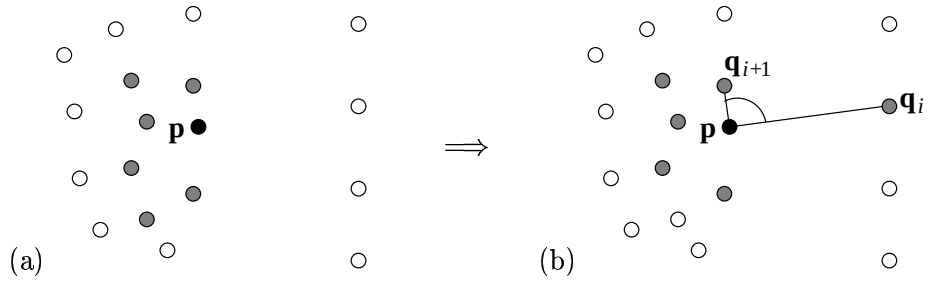

Figure 1: Necessity of the angle criterion for choosing the neighbours of a point $\mathbf{p}$.

Using the sorting of the selected neighbours such an environment can easily be visualized by a triangle fan. The whole point cloud is visualized by rendering all the triangle fans. Although the triangle fans do not form a coherent triangular mesh, it has been our experience, that there are almost no visually dangling parts.

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 [15]. A result is shown is Figure 2(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 2(b). In [23] Rusinkiewicz and Levoy develop their QSplat approach to overcome this problem. However, this approach needs a triangular mesh. Moreover for best results they resort to antialiasing techniques which further raise the time complexity of their method.

Compared to a triangular mesh our triangle fans consist overall of approximately $\frac{k}{2}$ times as many triangles. But eliminating duplicates of triangles leads to a comparable number of triangles for rendering point clouds and meshes, about $2.5n$ instead of $2n$ for $k = 6$.

4 Smoothing

Point clouds are gained by thoroughly scanning three-dimensional objects using various measuring techniques. Thus, before applying the visualization methods one has to overcome two major problems. On one hand the amount of data is too large to can be dealt with and on the other hand a number of measuring errors may occur leading to distortion. To eliminate such errors we apply a local smoothing operator to the point clouds.

For triangular nets several smoothing operators are known. Kobbelt [16] presents a discrete Laplacian smoothing operator that moves a single point $\mathbf{p}$ to the centroid $\frac{1}{k} \sum_{i=1}^{k} \mathbf{q}_i$ of its 1-ring formed by the points $\mathbf{q}_1, \ldots, \mathbf{q}_k$. Hence it is possible to apply the operator to arbitrary triangular meshes. Replacing the 1-rings by the neighbourhoods we can apply it to point clouds, too.

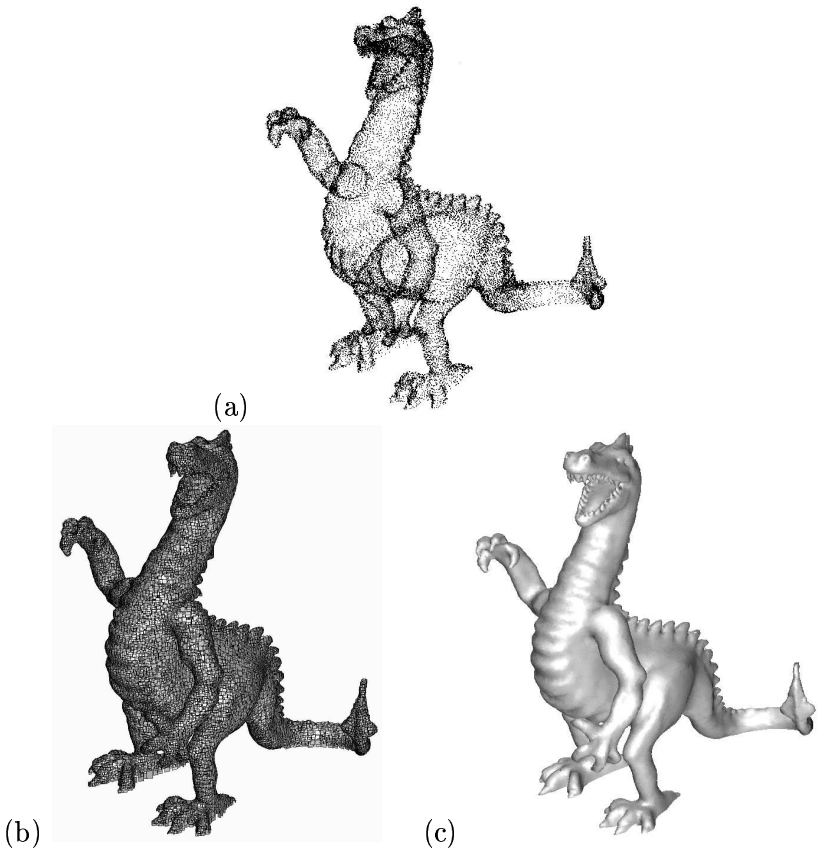

Figure 2: (a) Plotting the points of a point cloud. (b) Drawing an accumulation of pieces of the represented surface. (c) Our visualization of a point cloud representation.

Kobbelt makes further investigations, but Taubin [28] argues that these Laplacian smoothing operators cause shrinkage when applied to every vertex of a mesh. From a signal processing examination Taubin concludes that Laplacian smoothing without shrinkage is obtained by alternatively performing a smoothing step with positive scale factor λ and an un-shrinking step with negative scale factor μ. The factors fulfil the constraint $\lambda + \mu > 0$.

However, Guskov et al. [14] claim that the points $\mathbf{p}$ and $\mathbf{q}_1, \ldots, \mathbf{q}_k$ should be summed with coefficients, which do not only depend on the connectivity but also on the geometry. In Figure 3 we demonstrate, how the smoothing operators with connectivity-based coefficients try to make edge lengths as uniform as possible, whereas including geometrical aspects allows smoothing without affecting the triangle shapes. This is important for modelling purposes, especially when colour or texture information is used.

Their smoothing operator gives a mesh with a minimal sum

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

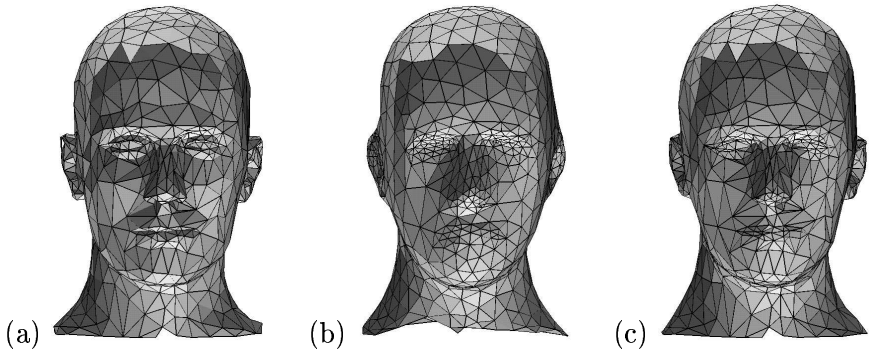

Figure 3: The Laplace-operator applied to a triangular mesh (a) changes the triangle shapes (b), in contrast to our smoothing operator (c).

of squared second differences

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

associated with an edge e and depending on the points of the adjacent triangles (cf. Figure 4). The coefficients for the second order difference are given by

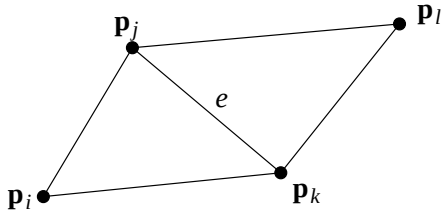

Figure 4: The support of $\mathcal{D}_e^2$.

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

where d_{jk} is the distance from $\mathbf{p}_j$ to $\mathbf{p}_k$ and A_{xyz} is the signed area of the triangle Δ_{xyz}. With a consistent orientation of the triangles, one does not have to take care about the sign of the areas and the coefficients sum up to zero. For computing the coefficients $c_{e,x}$, $\mathbf{p}_l$ is rotated using e as a hinge until the four points are coplanar. The associated smoothing operator is

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

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 4) 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 $\mathbf{p}_i$.

But the support of his operator includes points that have no connecting edge with the to be smoothed point. Hence the support is too big for our purposes, because it exceeds the environment covered by the neighbourhood.

So we developed a smoothing operator, that combines all the three important features, i.e. its support is limited to the neighbourhood, it is non-shrinking, and it includes geometrical aspects. By defining

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

with coefficients

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

and notations from above, we gain our smoothing operator

$$\Diamond\mathbf{p} := \sum_{i=1}^{k} \Diamond_{\mathbf{p}\mathbf{q}_i} - \mathbf{p} \ .$$

It does not yet fulfil the non-shrinkage criterion, but we can derive the equation

$$\Diamond\mathbf{p} = \sum_{i=1}^{k} \gamma_i\mathbf{q}_i - \mathbf{p} \ , \quad \sum_{i=1}^{k}\gamma_i = 1 \ .$$

Hence we can use the method of Taubin, which ensures non-shrinkage and convergence. For smoothing $\mathbf{p}$ we iterate the two steps

$$\begin{aligned} \mathbf{p} &:= \mathbf{p} + \lambda\Diamond\mathbf{p} \\ \mathbf{p} &:= \mathbf{p} + \mu\Diamond\mathbf{p} \ . \end{aligned}$$

These equations can be rewritten as

$$\mathbf{p} = \sum_{i=1}^{k} \omega_i\mathbf{q}_i \tag{1}$$

where the coefficients ω_i depend on the coefficients c_i and the factor λ (or μ). Note that in the uniform case the $\Diamond$-operator is equivalent to the Laplace-operator.

In Figure 5 we apply our smoothing operator to the distorted object (a) for $k = 8$ and gain the smooth point cloud representation (b).

5 Multiresolution

5.1 Reduction

After having eliminated error distortion the remaining major problem is to handle the large number of points. In general the sampling rate is high to obtain an oversampling, which is useful for error-estimation. Therefore the

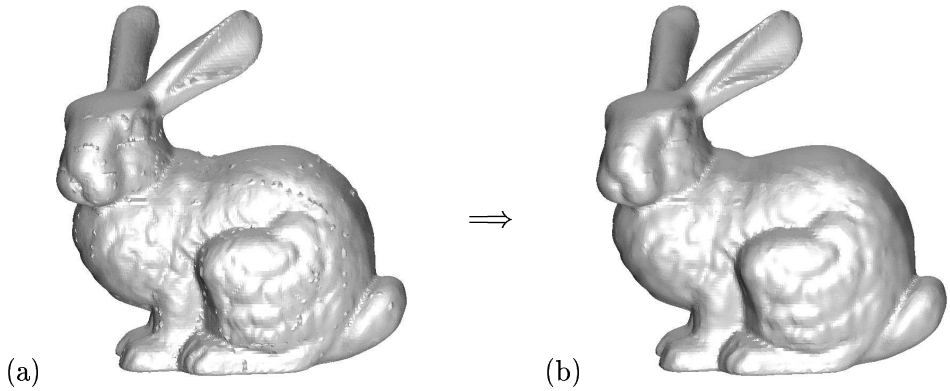

Figure 5: Error elimination by applying our smoothing operator.

sampled point cloud contains redundancy. This redundancy is eliminated by deleting points, which give no further information about the object's shape. So our aim is to provide every point $\mathbf{p}_i$ with an information content $M(\mathbf{p}_i)$ and to gradually delete the point with the lowest entropy.

Certainly the entropy depends on the distances from a point to its neighbours, because if two points are very close, one of them is supposed to be redundant. But the information content shall also take into account other features of the object's surface. For example for a planar region or a region of constant curvature less points are needed than for a region of heavy changes in curvature.

Following some ideas of Szeliski and Tonnesen [27] we define $M(\mathbf{p})$ so as to reflect the non-planarity M_p, change of the normals M_c and the non-uniformity M_u of this change. Let $\mathbf{p}_j = \mathbf{p} + \mathbf{d}_j$ for $j = 1, \ldots, k$ be the neighbours of $\mathbf{p}$ and let $\mathbf{n}_j$ and $\mathbf{n}$ be the corresponding normal directions. Then

$$M_d(\mathbf{p}) := \sum_{j=1}^{k} d_j^2 \ ,$$

where $d_j = \|\mathbf{d}_j\|_2$. Further, $M_p(\mathbf{p})$ is given by the distances of the neighbours from the tangent plane at $\mathbf{p}$

$$M_p(\mathbf{p}) := \sum_{j=1}^{k} (\mathbf{n}^t \ \mathbf{d}_j)^2 \ ,$$

$M_c(\mathbf{p})$ is given by

$$M_c(\mathbf{p}) := \sum_{j=1}^{k} \|\mathbf{n} - \mathbf{n}_j\|_2^2 \cdot d \ ,$$

where $d = \frac{1}{k} \sum_{j=1}^{k} d_j^2$, and $M_u(\mathbf{p})$ is defined as

$$M_u(\mathbf{p}) := \sum_{j=1}^{k} ((\mathbf{n} + \mathbf{n}_j)^t \ \mathbf{d}_j)^2 \ .$$

Figure 6 illustrates the meaning. Note that different from [27] these information

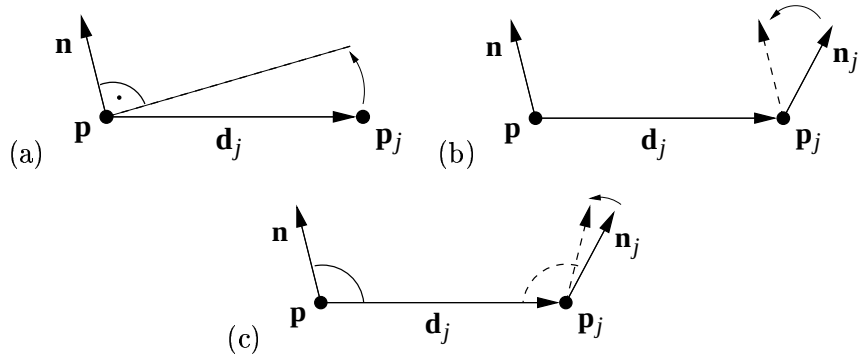

Figure 6: Illustration of non-planarity (a), change of the normals (b), and non-uniformity of this change (c).

contents are invariant under Euclidean transformations and scale uniformly for all $\mathbf{p}$ under scaling.

Another important feature for an object's representation is the colour information. If the RGB-values of $\mathbf{p}$ and its neighbours $\mathbf{p}_j$ are stored in $\mathbf{c}$ and $\mathbf{c}_j$ respectively, we define

$$M_{\text{colour}}(\mathbf{p}) := \sum_{j=1}^{k} \|\mathbf{c} - \mathbf{c}_j\|_2^2 \; d \; .$$

The information content $M(\mathbf{p})$ is a weighted sum of its components,

$$\begin{aligned} M(\mathbf{p}) \quad := \quad & \alpha_d M_d(\mathbf{p}) + \alpha_p M_p(\mathbf{p}) + \alpha_c M_c(\mathbf{p}) + \alpha_u M_u(\mathbf{p}) \\ & + \alpha_{\text{colour}} M_{\text{colour}}(\mathbf{p}) \; . \end{aligned}$$

Figure 7 shows an example, where the object shown in (a) is reduced to 42% by removing in (b) the points with the least values M_d and in (c) with the least information content M. Note that fine details are lost in (b) but not in (c), in particular look at the teeth.

The reduction of a point cloud containing colour information is shown in Figure 8. A flat shading is used to emphasize the changes in colour distribution, when M_{colour} is not used.

5.2 Detail information

In the last subsection we discussed how to eliminate redundancy by deleting certain points. But for rendering an object in a scene, where the viewer is far from the object, a lower resolution suffices and even less points are needed. So we delete further points, but need to store some detail information, because as the viewer comes closer to the object, we need the higher resolutions again.

Such a level-of-detail control is also helpful for progressive transmission over a network or for progressive uploading from a storage medium. A complex object is displayed beginning with a low resolution and then progressively improving the display as more detail is obtained.

For the level-of-detail control it suffices to store global informations about the deleted points. But applying multiresolution editing (cf. Section 6.1) the

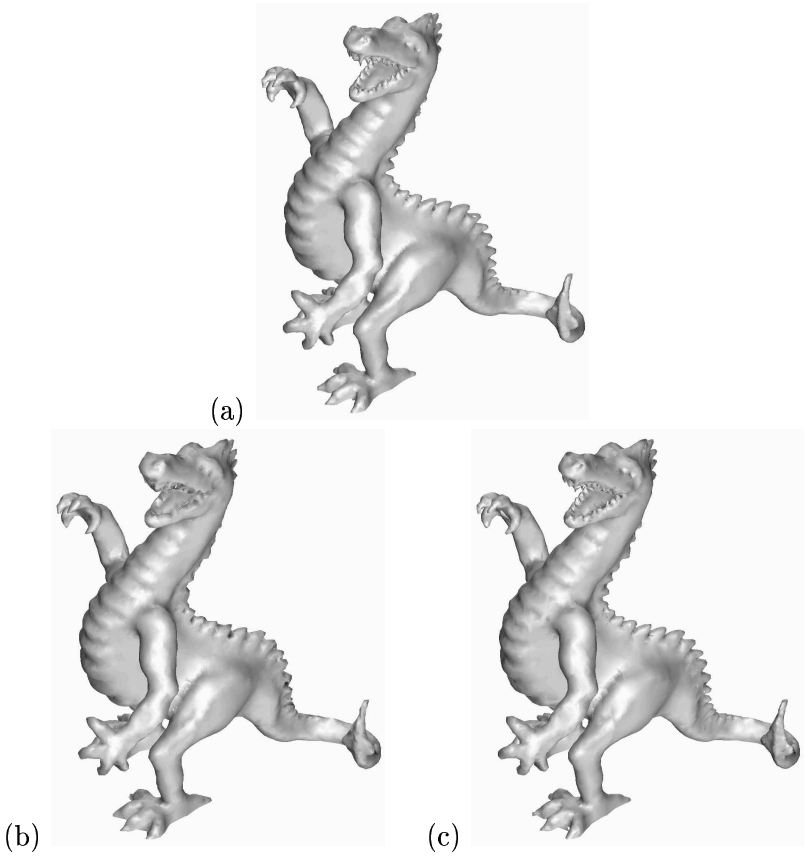

Figure 7: Reducing a point cloud (a) only by considering distances (b) or by considering surface features as well (c).

object's shape is changed on a low level of detail and the reinserted points shall fit into this new shape. This requires a storage of local informations of a point due to its neighbourhood.

When a point $\mathbf{p}$ is chosen to be deleted, we compute its optimal location $\mathbf{p}' := \mathbf{p} + \Diamond\mathbf{p}$ due to the smoothing operator $\Diamond$ (cf. Section 4). $\mathbf{p}'$ depends only on the neighbours $\mathbf{q}_1, \ldots, \mathbf{q}_k$ and the coefficients $\omega_1, \ldots, \omega_k$ of Equation 1 ($\lambda = 1$). Then we store the distance vector $\mathbf{d} := \mathbf{p} - \mathbf{p}'$ from the optimal location $\mathbf{p}'$ to the actual location $\mathbf{p}$ (cf. Figure 9). $\mathbf{d}$ contains the detail information of the deleted point $\mathbf{p}$.

For reinserting the point $\mathbf{p}$ we again compute the optimal location $\mathbf{p}'$ by applying our smoothing operator to the neighbours $\mathbf{q}_1, \ldots, \mathbf{q}_k$. By adding the detail vector $\mathbf{d}$ to $\mathbf{p}'$ we get the original location of $\mathbf{p}$, if the locations of the neighbours remained unchanged.

If the neighbours were moved, the locations of $\mathbf{p}'$ and finally of $\mathbf{p}$ are affected by this modification. But if the whole neighbourhood was transformed, e.g. by a rotation, the detail vector $\mathbf{d}$ shall be transformed, too (cf. Figure 10). Therefore the detail vector has to be stored in a local frame. We do this by storing $\mathbf{d}$ in relation to the least squares best fitting plane of the neighbours. When the neighbours are transformed uniformly, the least squares best fitting plane is

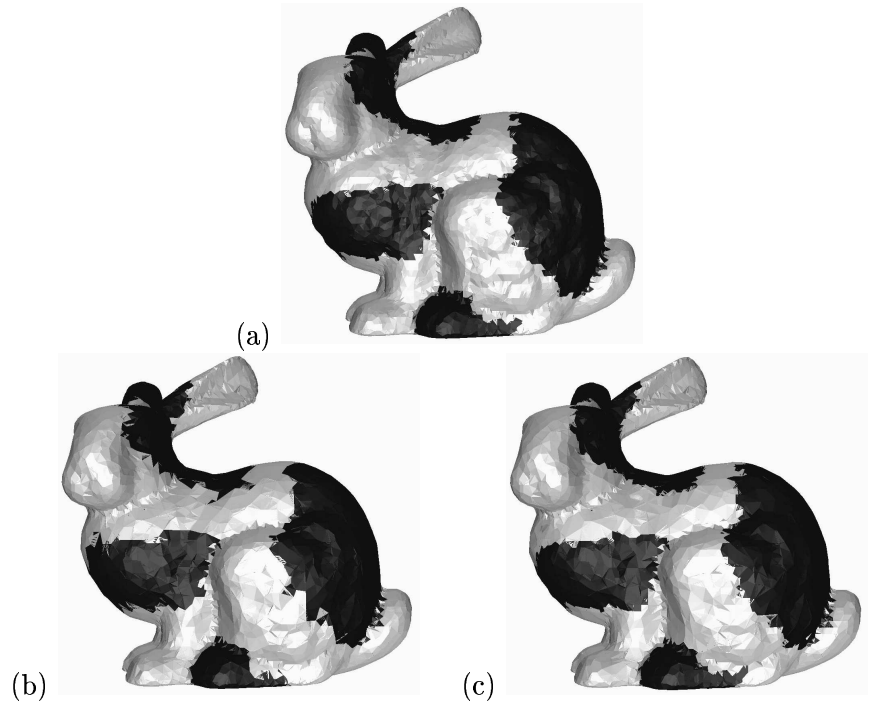

Figure 8: Reducing a point cloud (a) to 50% with (c) and without (b) taking care about the colour information.

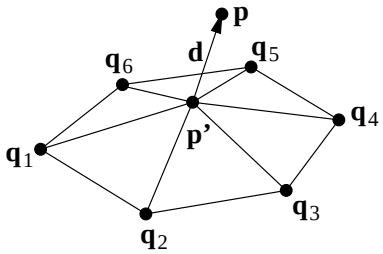

Figure 9: Detail vector **d**.

modified in the same way.

5.3 Refinement

Even if there is no detail information it is possible to refine a point cloud. In our experiments we used the information content of Subsection 5.1. To insert a point we determine the point $\mathbf{p}$ with highest information content $M(\mathbf{p})$, its neighbour $\mathbf{q}$ with highest information content and take the predecessor or successor $\mathbf{r}$ of $\mathbf{q}$ in the neighbour sorting of $\mathbf{p}$ again due to highest information content. The point we insert is

$$\mathbf{s} = \frac{M(\mathbf{p})\mathbf{p} + M(\mathbf{q})\mathbf{q} + M(\mathbf{r})\mathbf{r}}{M(\mathbf{p}) + M(\mathbf{q}) + M(\mathbf{r})} \ .$$

Finally, $\mathbf{s}$ is moved along the surface normal by applying the smoothing operator $\triangleq$. Figure 11 shows an example.

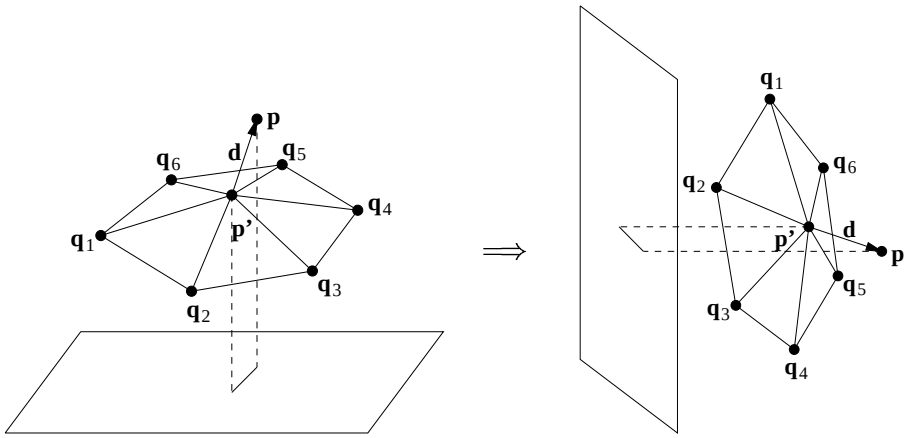

Figure 10: Storing the detail vector in a local frame.

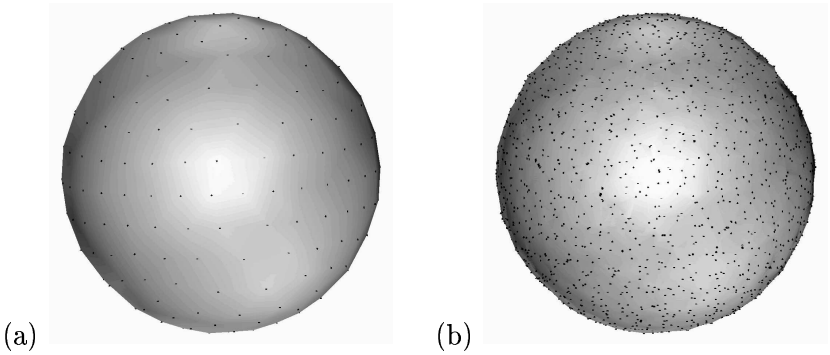

Figure 11: Refining a point cloud, thus raising the number of points to 1000% of the original amount.

6 Modelling

6.1 Interactive multiresolution modelling

After having developed a powerful tool for treating point clouds, namely the multiresolution method, we apply this technique to interactive modelling. In Figure 12 is illustrated how we proceed.

The original point cloud is visualized in Figure 12(a). The number of points is reduced to 3% of the original amount. The reduced point cloud is shown in Figure 12(b). Then we interactively edit the shape of the object. For the figured bunny we rotate the head by $\frac{\pi}{4}$ and stretch the snout and the tail, while the ears are being shortened. As a result we gain the kangaroo-like animal in Figure 12(c). Finally we refine the modified point cloud by reinserting the deleted points and adding the details up to the original level of detail. In Figure 12(d) the result is shown.

So what we do is to edit the global shape of the represented object, while leaving the small details unchanged. This is because during the reduction the small details are stored in the detail vectors. Hence they remain unaffected by the interactive editing and are reinserted during the refinement.

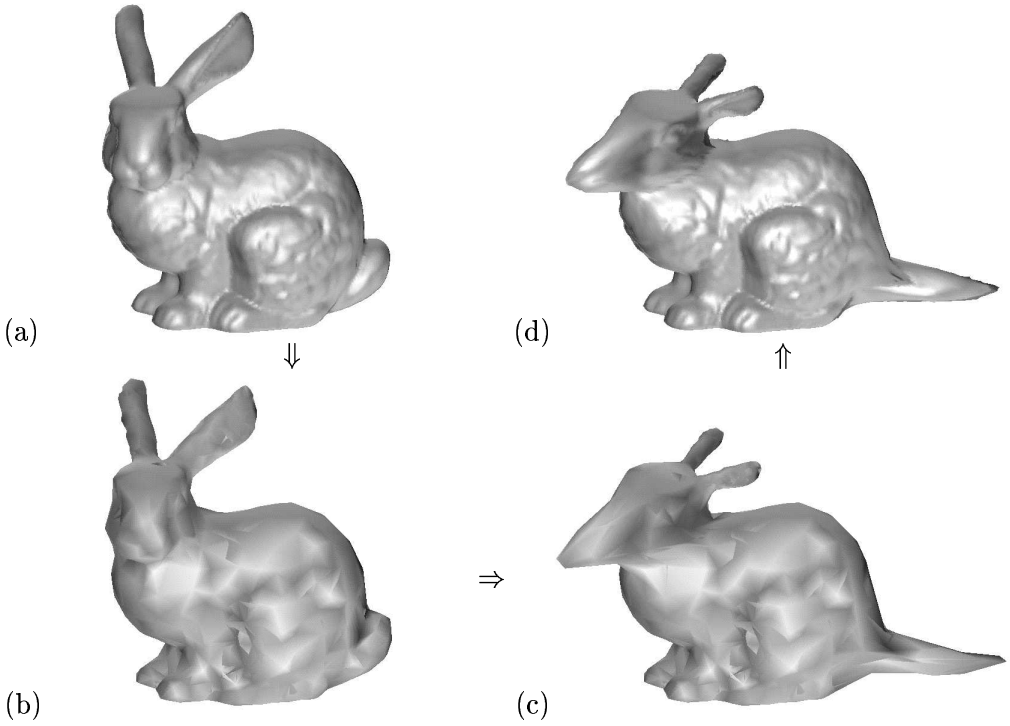

Figure 12: Multiresolution editing.

6.2 Detail frequency spectrum

The detail vectors of the multiresolution methods introduced in Section 5 can be used in another way. Since the spatial extent of the detail information is given by the neighbourhood of the point, the extent increases while deleting the points. Thus by taking this order, we get a frequency spectrum of the detail information. The detail vectors of early deleted points are related to small distances and high frequencies, whereas the detail vectors of lately deleted points are related to large distances and low frequencies.

To this frequency spectrum we apply filtering. A low pass filter leaves all the details unchanged up to a certain frequency and from then on sets all the details to zero. A high pass filter, which annihilates the coarsest details, does not make sense, because it removes the basic informations about the object's shape. More useful are stopband filters, which annihilate details in a certain range of the frequency spectrum.

In Figure 13 we give an example for low pass and stopband filtering. Note that using the low pass filter the high-frequency details like the mountain ridges and peaks vanish, whereas using the stopband filter these fine details remain but some coarser characteristics get smoothed, which leads to the global flattening of the mountain landscape.

Low pass filters provide a second facility for smoothing a point cloud. But this signal processing approach offers even more modelling possibilities by filtering certain frequencies. Nevertheless for smoothing huge amounts of error-distorted data the smoothing operator defined in Section 4 should be preferred, because it quickly eliminates the high-frequency distortion.

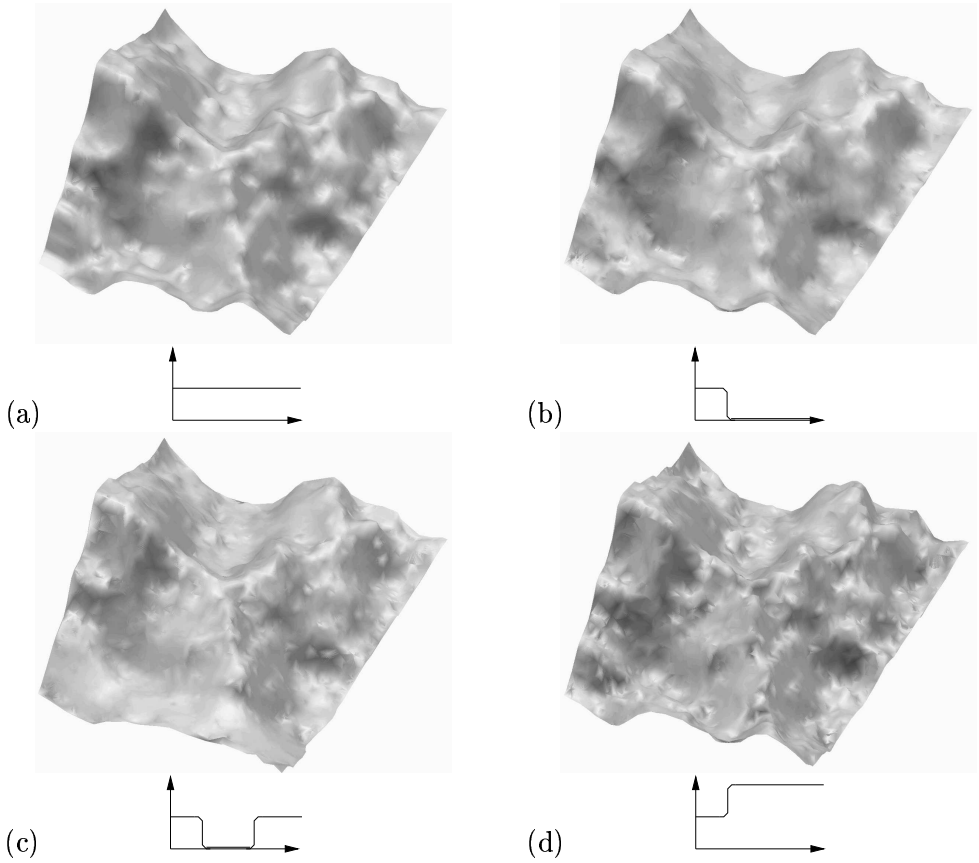

Figure 13: Modelling a point cloud (a) by using a low pass filter (b), a stopband filter (c), and enhancement (d).

Rather than annihilating the detail vectors they can also be multiplied by a constant factor. Therefore instead of deleting some details they are enhanced. In Figure 13(d), note how the features of the mountain landscape are intensified.

6.3 Extended CSG

The idea of Constructive Solid Geometry is to use basic simple solids and some modelling operations like union, intersection, and difference to construct more complex solids. This approach is extended to objects of arbitrary shape in [18]. The surfaces of objects are represented by meshes, on which the modelling operations are carried out. This requires cutting off and connecting meshes.

We transfer this idea to point clouds. We traverse the minimum spanning tree from Section 3 of the first point cloud and consider every line **pq** from a point **p** to its son **q**. For the line **pq** we look for intersections with the neighbourhoods of all points of the second point cloud. Due to the existence of such intersections, we propagate from **p** to its son **q**, whether the point **q** should contribute to the resulting point cloud or not. We proceed analogously for the second point cloud.

We assume that the sampling rate is that high, that the line **pq** does not intersect twice the surface of the second object. This is necessary, because

neighbourhoods may overlap and therefore we can not distinguish between another intersection of the line with the second object's surface and a duplicate of the already known intersection. Note that this assumption can always be fulfilled by refining the point clouds.

The major problem using meshes is to connect the remaining parts and to construct one single mesh, especially when topological constraints are to be fulfilled. Using point clouds we have nothing to do but simply collecting the desired points.

In Figure 14 is shown, how the object of a dragon (left) is united with another head of a dragon (middle). Note that there are no points in the interior

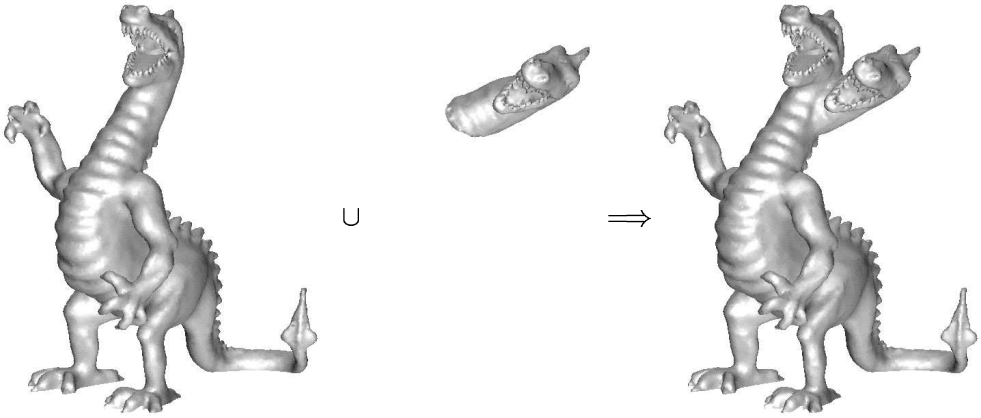

Figure 14: Union of two solids in point cloud representation.

of the resulting solid (right).

7 Discussion

Arbitrary surfaces are traditionally represented by piecewise polynomial functions like splines. With the upcoming of subdivision algorithms a simpler representation based on meshes became more important. We perform another step of simplification by introducing the point cloud representation. In this section we discuss the advantages and disadvantages of point clouds in comparison with triangular meshes.

Using triangular meshes for surface reconstruction, a mesh needs to be generated before the required operations like smoothing can be executed. Because the original data contain too many points for a mesh generation, some heuristics must be found to select some of them. Using point clouds allows to smooth the data immediately. Thus the whole sample information is available. Afterwards a reduction can be executed due to surface and colour features.

Various methods were developed for the triangular mesh generation. Most 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 look at the overall running times in table 1. For the chosen examples they are reduced from

minutes to seconds.

method	# points	computer	time
Delaunay tetrahedrization [19]	1248	SGI Indigo2 Extreme	45 s
	2600		approx. 1 min
	48921		approx. 133 min
Algorri, Schmitt [1]	45233	Sun Sparc Station 40 MHz	18 min 19 s
Bajaj et al. [3]	9223	SGI Indigo2	approx. 10 min
Edelsbrunner [9]	9600	SGI 50 MHz, MIPS R4000	approx. 39 min
	10000		approx. 16 min
	10088		approx. 26 min
	15000		approx. 27 min
Hoppe et al. [15]	18224	20 MIPS Workstation	31 min 15 s
Mencl, Müller [19]	1248	SGI Indigo2 Extreme	approx. 2 min
	2600		approx. 5 min
	48921		approx. 193 min
Shimada [25]	1000	IBM RS/6000	approx. 40 s
point clouds visualization	47109	SGI Indigo2 Extreme	59 s
		Sun Ultra30	20 s
		PC with Athlon K7 800MHz	7 s
	100001		24 s
	160940		59 s

Table 1: Comparing the running times of triangular mesh generations to the visualization of point clouds.

Newer approaches [4, 11] try to overcome the drawback of high costs for mesh generation by making extra assumptions on the sampling rate. They achieve lower running times at the expense of generality.

Another advantage in complexity occurs while storing an object in point cloud representation, because no connectivity information has to be saved. Nevertheless if an object needs to be visualized several times, it is reasonable to store the neighbourhoods rather than recomputing them every time, which leads to similar storing complexities for triangular meshes and point clouds.

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

If the measuring techniques are not sufficiently reliable or the sampling rate is too low, multiple sampling may help, i.e. the object is scanned more than once and the samples have to be merged. For triangular meshes this causes a complete remeshing, whereas for point clouds nothing has to be done but only collecting the various samples in one point cloud.

A similar problem occurs, when measuring techniques such as range images are used, which scan the object from different viewpoints. Many merging steps like the one shown in Figure 15 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 (cf. Subsection 6.3). Numerical problems may occur. For point clouds the points of the multiple range images are simply stuck into one single point cloud. Due to calibration errors we apply our smoothing operator to the regions, where the samples overlap. If desired the overlapping region can be reduced. A result for point clouds is shown on the right of Figure 15.

We conclude that having found a nice visualization method for point clouds, whose quality is comparable with the one for triangular meshes, other techniques of geometric modelling and design can be used and work very well. So

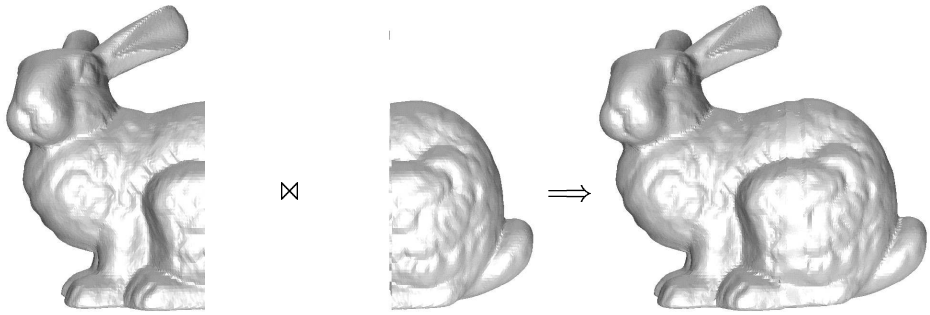

Figure 15: Merging two partly overlapping samples to build one single point cloud.

we spare the time-consuming triangular mesh generation and can apply our smoothing operator and our reduction method directly to the numerous points of sampled objects. We even recognise that some things become easier, when topological constraints can be neglected.

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 Transactions 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 Schrder, Alan H. Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow.* Proceedings of SIGGRAPH 1999, 1999.

[8] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, Werner Stuetzle: *Multiresolution Analysis of Arbitrary Meshes.* Computer Graphics (SIGGRAPH '95 Proceedings), 173-182, 1995.

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

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

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

[12] J. P. Grossman, William J. Dally: *Point Sample Rendering.* Rendering Techniques '98, Springer, Wien, 181 - 192, 1998.

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

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

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

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

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

[18] Lars Linsen: *Netbased Modelling.* SCCG 2000 proceedings, Budmerice (Slovakia), May 2000.

[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.