EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (*2001*), *Number 3*

Resampling Feature and Blend Regions in Polygonal Meshes for Surface Anti-Aliasing

Mario Botsch   Leif Kobbelt

Computer Graphics Group
RWTH-Aachen

Abstract

Efficient surface reconstruction and reverse engineering techniques are usually based on a polygonal mesh representation of the geometry: the resulting models emerge from piecewise linear interpolation of a set of sample points. The quality of the reconstruction not only depends on the number and density of the sample points but also on their alignment to sharp and rounded features of the original geometry. Bad alignment can lead to severe alias artifacts. In this paper we present a sampling pattern for feature and blend regions which minimizes these alias errors. We show how to improve the quality of a given polygonal mesh model by resampling its feature and blend regions within an interactive framework. We further demonstrate sophisticated modeling operations that can be implemented based on this resampling technique.

1. Introduction

Surface reconstruction usually refers to the process of deriving manifold surface information from measured point data. The input data typically comes as a dense (in general unstructured) cloud of sample points in 3–space and the output is the mathematical description of a surface that interpolates or approximates all or some of the samples.

The classical *scattered data interpolation techniques* are mostly based on fitting spline surfaces [11] or surfaces spanned by radial basis functions [15] to the discrete data. The variety of shapes that can be reconstructed with these techniques is limited since a global parameterization of the data over some parameter domain $\Omega \subset \mathbb{R}^2$ is required.

With the wider availability and improving performance of 3D scanning technology, the complexity of geometric data sets has increased significantly. Data sets with several million sample points are routinely generated from scanning highly complex shapes. Since classical approximation techniques can no longer be applied without involved pre–processing (e.g. segmentation [21, 22]), many surface reconstruction schemes based on polygonal meshes have been proposed over the last years.

There are different techniques to generate interpolating or approximating triangle meshes for a given cloud of sample points. One approach is to connect the given samples directly by estimating the neighborhood relation between the points from their spatial constellation [7, 1]. Some of these algorithms combine the neighbor–finding with a subsampling mechanism to control the complexity of the resulting mesh [2]. Other approaches derive a volumetric signed distance field for the space around the cloud of samples and generate an approximating triangle mesh by extracting the zero–level iso–surface from that volumetric scalar field [9, 5, 23].

All the above techniques lead to highly detailed triangle meshes. Although most algorithms allow the user to control the output complexity by globally adjusting the resolution, it is often necessary to set the resolution high enough to avoid topological ambiguities. An additional drawback is due to the fact that the resolution can only be changed globally and hence we either lose relevant geometric detail (if we set the resolution too low) or we extremely oversample flat surface regions (if we set the resolution to high). Locally adapting the resolution is difficult since this requires to detect the presence of fine detail (= estimate the surface curvature) *before* the surface is actually generated. In coarse-to-fine approaches like [23] the mesh resolution is adapted by selective refinement based on an *a posteriori* estimator.

The standard procedure to avoid these difficulties is therefore to first reconstruct highly complex meshes and then apply some mesh decimation technique [10, 18, 13] which effectively reduces the number of triangles while keeping a prescribed approximation tolerance and optimizing the visual quality. Out–of–core decimation algorithms are able to process data sets of virtually any size [16, 4]. As a result we obtain a triangle mesh that approximates the original geometry up to a prescribed tolerance with a minimum number of triangles.

Since most geometry-based decimation schemes are greedy algorithms that only consider the *local* shape to decide about which vertex to remove in the next step, one usually has no direct influence on the distribution and global alignment of the mesh vertices on the surface. The only guarantee is that the vertex distribution correlates with the surface curvature, i.e. we obtain a low vertex density (large triangles) in flat regions and a high density (small triangles) in curved regions. For the better decimation algorithms we can further observe that in cylindrical regions with high curvature in one and low curvature in the orthogonal direction, the vertex distribution is anisotropic, leading to long thin triangles along the cylinder axis direction (cf. Fig. 1). This is an added value to those decimation schemes since such triangulations approximate cylindrical regions much better for a fixed triangle budget.

While such decimated meshes are well–suited for displaying, they turn out to be inappropriate for more sophisticated downstream applications like numerical simulation (e.g. CFD). The reason for this is that the (weighted) random distribution of vertices (= surface samples) leads to severe alias errors which become visible as flat shading artifacts (cf. Fig. 1) and which can lead to erroneous simulation results. Those alias errors are caused by the fact that although the decimated triangle mesh stays pointwise within some tolerance to the original data, the normal vectors can deviate significantly.

This so called normal noise becomes particularly evident in the vicinity of feature lines on the original shape. Here the two principal curvatures differ very strongly – in the extreme case of sharp features, the curvature across the feature even diverges.

The only way to solve this geometric alias problem in surface reconstruction is to choose the “right” sampling pattern, i.e. to globally adjust the distribution and alignment of mesh vertices such that the normal vectors of the triangles approximate the normal vectors of the original surface.

In this paper we propose a solution to this sampling problem. We present a technique to resample the feature regions of a *given* triangle mesh such that the alias artifacts are strongly reduced. We advocate for an integration of this technique into a semi–automatic set–up since we consider the problem of *detecting* feature regions to be independent from the actual (re–)sampling problem: If we would have a reliable technique for feature detection, we could combine it with our resampling technique to implement a fully automatic resampling scheme. For industrial surface design applications, manual feature detection is acceptable and even preferred by most designers.

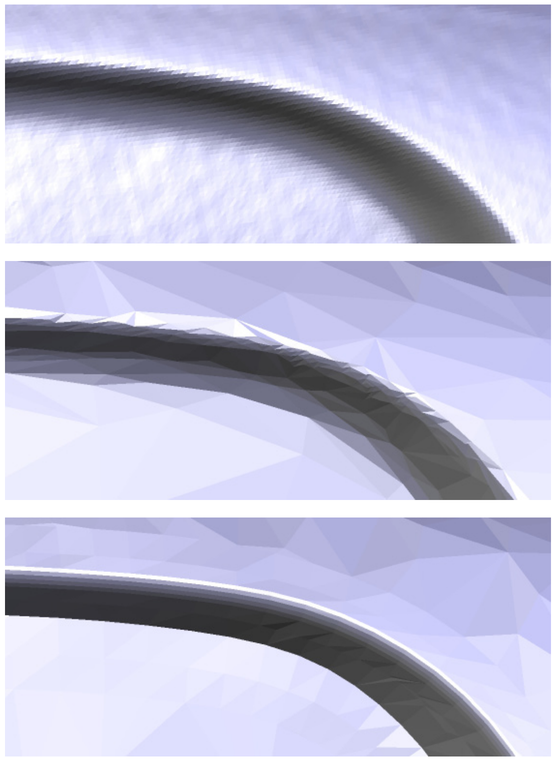
Figure 1: *Geometric alias effects such as normal noise become clearly visible under specular shading. The top image shows an original 3D–scan of a feature region. Although the point positions have been sampled at high precision, the normals of the resulting mesh deviate strongly from the normals of the original surface. Applying mesh decimation (center) improves the situation slightly since the triangles are stretched along the feature but the normal noise is still disturbing. In the bottom image we applied our alias–reducing feature resampling. Although the mesh resolution has not changed, the quality has improved due to effective normal noise elimination.*

The main contribution of this paper is the definition, justification, and application of a sampling pattern for geometric feature regions which provably satisfies the mesh quality requirements for numerical simulation applications. We further present an efficient and effective technique to reverse–engineer these sampling patterns with only little user input for the features of a given geometric model. We finally demonstrate that the additional structure of the resampled mesh models provides the basis for a number of high level modeling operations.

2. Feature regions

In the boundary representation of geometric (solid) models we can distinguish three types of surface regions: there are *geometric primitives* (parts of spheres, cylinders, or tori), *freeform surfaces* (smooth surface patches of general shape), and *blends*. Blends are usually constructed to join the other surface parts in order to obtain a consistent representation of a closed solid.

A blend surface can be thought of as either generated by rolling a ball with varying or constant radius over the gap between two surface segments or by sweeping a profile curve along the two opposite boundaries (cf. Fig. 2). In the extreme case, a blend can degenerate to a feature curve where two surface segments meet with discontinuous tangent planes. We call such feature curves *sharp features*. They correspond to rolling ball blends with zero radius of the ball.

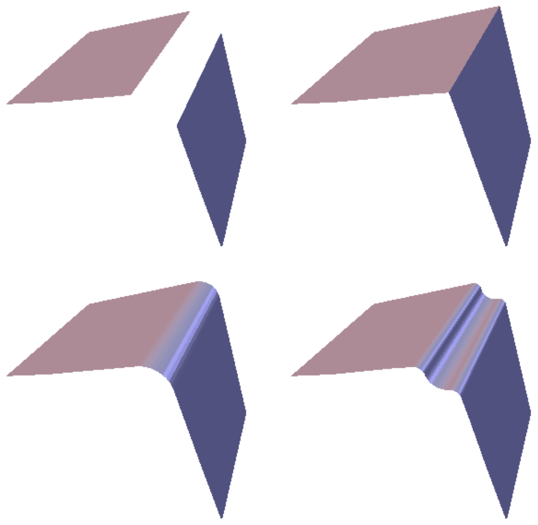
Figure 2: *Feature regions on a complex surface usually emerge from blending two separate patches along the corresponding boundary. The two patches on the top left can be joined by computing their intersection. This leads to a sharp feature line (top right). Alternatively we can roll a ball of prescribed radius (bottom left) or sweep a more complicated profile (bottom right).*

When sampling a surface, we have to adapt the sampling density to the local curvature distribution in order to capture all (and only) relevant geometric details. Obviously, in highly curved regions where the principal curvatures κ_1 and κ_2 are both large, we have to sample more densely than in regions where both principal curvatures are small. If the magnitude of the curvatures does not differ too much then an isotropic sampling pattern is fine. However, since *two* principal curvatures characterize the local curvature, an optimal sampling pattern has different densities in the corresponding principal directions. Hence, in *feature regions* – characterized by $\kappa_1 \ll \kappa_2$ – we have to use an anisotropic sampling pattern and this pattern should be aligned to the principal directions.

In terms of the above classification into *primitives*, *freeform patches*, and *blends* the feature regions are usually the blend areas where, e.g., a sphere of radius $1/\kappa_2$ rolls along a curve with curvature $\kappa \approx \kappa_1 \ll \kappa_2$.

3. Surface sampling

In the introduction, we pointed out that the weighted random distribution of surface samples as it emerges from mesh decimation does not yield satisfactory results in feature regions due to *normal noise*. We now give a more precise definition of normal noise and then present a simple sampling pattern for feature regions that reduces normal noise to a minimum.

The roughness of a triangle mesh can be measured by some discrete analogon to the concept of curvature [6, 20, 17]. A simple discretization is, e.g., to rate the curvature (= non–planarity) across an edge of the mesh by the angle between the normal vectors of the adjacent triangles. We call this angle the *normal jump*. If the triangle mesh is an orientable manifold then we can distinguish *convex* normal jumps (positive sign) and *concave* normal jumps (negative sign). By adding the normal jumps with respect to its three direct neighbors, we obtain a single curvature value per triangle which can be interpreted as *discrete mean curvature*.

High quality ("class A") surfaces in Geometric Modeling and CAD are usually characterized by low variation of curvature. Most surface fairing techniques improve a given shape by reducing the surface's curvature or its variation in an optimization process [19, 6, 12]. Transferring this notion of *fairness* to triangle meshes implies that we consider meshes to be of high quality if the variation of the normal jumps is low.

For a low quality mesh with strongly varying normal jumps the individual triangle normals are more or less randomly tilted away from the normal cone corresponding to the underlying surface patch (cf. Fig. 1). We call this effect which becomes clearly visible under specular shading of the surface *normal noise*, because it behaves very similar to *surface noise* which refers to a high frequency perturbation of the vertex positions away from the underlying surface. The process of reducing or even removing normal noise by choosing an appropriate sampling pattern is called *surface anti–aliasing*.

It is easy to see that random sampling generally leads to significant normal noise. Consider, e.g., the simple example of an orthogonal cylinder. Placing the samples randomly on the surface causes an uncontrollable tilt of the triangle normals away from the original surface normals which are all orthogonal to the cylinder's axis.

If we reduce the sampling density in the direction of the cylinder axis we obviously reduce the normal noise since the resulting long and thin triangles become more and more parallel to the cylinder axis and hence their normals become approximately orthogonal to the axis. However in general the normal noise will never disappear completely and, in fact, we trade the triangle's aspect ratio (another mesh quality criterion) for the reduced normal noise.

The generic configuration of a triangle's normal vector being orthogonal to the cylinder axis occurs if the triangle's embedding plane intersects the cylinder in two parallel lines. Since the triangle is spanned by three surface samples, these samples also have to lie on those two lines. This implies that a triangle is free of normal noise if and only if one of its edges is parallel to the cylinder axis.

Now consider a sampling pattern where all samples lie on a set of lines which are parallel to the cylinder axis and distributed equally around the cylinder. Each strip between two of those lines can be tesselated by a planar triangulation. As a consequence the normal jumps between triangles are either zero (within the same strip) or a constant angle that only depends on the number of strips. Hence the normal noise is minimal. The two different normal jump values correspond to the two principal curvatures on the cylinder surface.

We now generalize this idea to derive a sampling pattern for surfaces that are part of an envelope generated by moving a sphere of constant radius along a space curve. In Section 5 we will apply the same sampling pattern to even more general profile sweep surfaces to empirically demonstrate that we still obtain superior quality meshes compared to random sampling although we can no longer guarantee zero normal noise in this generalized setting.

The envelope of a moving sphere can be defined alternatively by a *center curve* $\mathbf{g}(t)$ along which a planar circle profile is moved. The orientation of the circle's embedding plane at a time step t_0 is defined by the tangent $\mathbf{g}'(t_0)$ of the center curve. The sweep surface itself is the collection of all profiles at different time steps $t \in [a,b]$. We assume that the minimum curvature radius of the center curve $\mathbf{g}$ is larger than the radius of the circle profile to avoid the discussion of surface degeneracies.

According to the above definition we can distinguish two natural directions on such a sweep surface S: one *along* the center line and one *around* the center line. We can use these directions for a natural parameterization $S(t,u)$ with t varying along and u around the center curve. In this parameterization, the iso–curves with constant parameter t_0 are circles around the center $\mathbf{g}(t_0)$. Iso–curves with constant parameter u_0 are the *trajectories* along which a specific point on the circle profile moves. Obviously, the trajectories are offset–curves to the center curve and consequently the iso–curves with respect to the parameter t and u intersect perpendicularly. In fact, it can be shown that the iso–curves are the principal curvature lines of the sweep surface [3]. Another important property of the trajectories which will be used later on, is that they have constant Euclidean distance as well as constant geodesic distance.

The sampling pattern for the sweep surface has to discretize the parameter domain in t and u direction. We start by discretizing the moving profile itself. This means we approximate the circle $S(0,u)$ by a closed polygon $P = [\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}]$ with $\mathbf{p}_i = S(0, i/n)$. When sweeping this closed polygon instead of the circle, we obtain a surface that consists of n ruled surfaces

$$R_i(t,u) \;=\; (1-u)\; S(t, \frac{i}{n}) \;+\; u\; S(t, \frac{i+1}{n}).$$

Because the trajectories along which the points $\mathbf{p}_i$ move are perpendicular to the profiles, we immediately see that if we choose the polygon P to be a regular n–gon then the normal jump between neighboring ruled patches R_i is exactly $2\pi/n$ everywhere (cf. Fig. 3). Hence we have a constant normal jump (= zero variation).

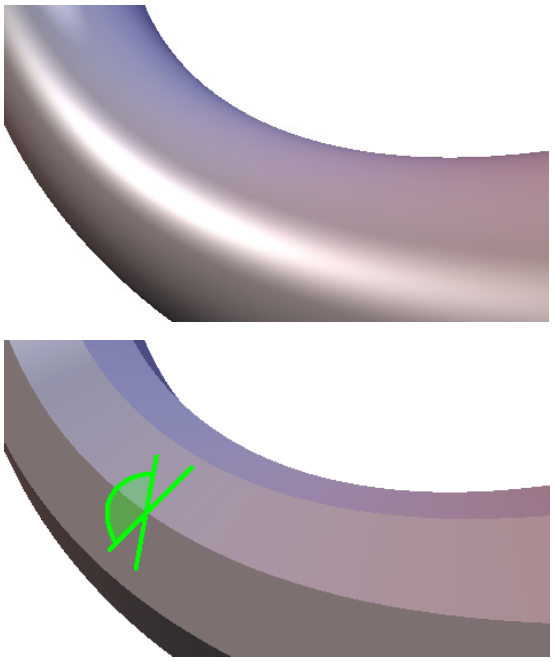

Figure 3: *By discretizing the sweep profile we approximate the original envelope surface S by a collection of n ruled surfaces R_i. Since we replace the circle profile by a regular n–gon which moves orthogonally to its embedding plane, all normal jumps between neighboring strips are constantly equal to $2\pi/n$.*

Next we have to find a triangulation for each ruled patch R_i which corresponds to a discretization in t direction. Since the trajectories are lines of minimal curvature, we do not expect large normal jumps between triangles within the same strip. However, we have to make sure that the constant normal jump property of the n–gon sweep is preserved as good as possible.

If we approximate the segment $\mathbf{g}([t_0,t_1])$ by a straight line or a circular arc then the resulting patch $R_i([t_0,t_1],[0,1])$ is a cylindric or conic surface patch. Notice that the approximation error of a circular arc to the curve segment $\mathbf{g}([t_0,t_1])$

decreases like $O(|t_1 - t_0|^3)$. Hence, even for general center curves $\mathbf{g}$ we can locally approximate each surface patch $R_i([t_0,t_1],[0,1])$ by a cylindric or conic surface patch. Moreover R_i is linear in u direction and the two iso–curves $R_i(t,0)$ and $R_i(t,1)$ are offsets of each other.

For a conic patch $R_i([t_0,t_1],[0,1])$, any two generator lines (e.g. $R_i([t_0],[0,1])$ and $R_i([t_1],[0,1])$) intersect in a common point, the apex. Hence the quadrilateral $[R_i(t_0,0),R_i(t_0,1),R_i(t_1,0),R_i(t_1,1)]$ is planar. If $R_i([t_0,t_1],[0,1])$ slightly deviates from a conic patch, the corresponding quadrilateral will still be very flat. Consequently, the quadrilateral spanned by $R_i(t_0,0)$, $R_i(t_0,1)$, $R_i(t_1,0)$, and $R_i(t_1,1)$ will be very close to planar and no matter how we split it into two triangles, we do not introduce a significant normal jump between the two resulting triangles. In addition, if we look at neighboring quadrilaterals $[R_i(t_0,0),R_i(t_0,1),R_i(t_1,0),R_i(t_1,1)]$ and $[R_{i+1}(t_0,0),R_{i+1}(t_0,1),R_{i+1}(t_1,0),R_{i+1}(t_1,1)]$, the normal jump between them is approximately $2\pi/n$ since each quad is spanned by a pair of generating lines from the neighboring strips R_i and R_{i+1} for the same parameter value t_0 and t_1 respectively.

Hence, it turns out that the regular triangulation for each strip which uses the sample pairs $R_i(t_j,0)$ and $R_i(t_j,1)$ for any sequence of parameter values t_j does not introduce significant normal noise. Moreover, we can even show that any modification of the triangulation only increases the normal noise.

Consider, e.g., the four samples $A = S(t_0,i/n)$, $B = S(t_1,i/n)$, $C = S(t_2,(i-1)/n)$, and $D = S(t_3,(i+1)/n)$ which define two triangles $T_1 = [A,B,C]$ and $T_2 = [D,B,A]$ in neighboring strips. If the trajectories $S(t,(i-1)/n)$, $S(t,i/n)$, and $S(t,(i+1)/n)$ are no straight lines then the normal angle between T_1 and T_2 can differ significantly from the optimal value $2\pi/n$ when we choose the parameter values t_2 and t_3 from the interior of the interval $[t_0,t_1]$ (cf. Fig. 4). This local deviation of the normal jump from the average $2\pi/n$ propagates across the mesh because the sum of the normal jumps along a planar contour around the center line is constantly equal to 2π.

In conclusion of this section we find that the key to an anti–aliased sampling pattern on spherical sweeps is to arrange the surface samples $\mathbf{p}_{i,j} = S(t_i,u_j)$ such that the the points $[\mathbf{p}_{i,j}]_i$ lie on a common circle around the center curve and the samples $[\mathbf{p}_{i,j}]_j$ lie on trajectories.

4. Interactive surface reconstruction

In the last section we showed that a rolling–ball blend should be triangulated based on a sampling pattern that is aligned to the principal curvature directions (trajectories and circle profiles) in order to minimize normal noise. However, sampling an existing feature with unknown center curve $\mathbf{g}$ from a given triangle mesh is a different situation: unless we are dealing with a sharp feature (i.e. its radius equals zero) the center curve does not lie on the given surface, neither do we know the radius of the profile that was swept along it. Instead, the surface data only provides us the resulting blend. In order to reverse–engineer the feature and to resample it in an anti–aliased manner we have to generate the sampling pattern without explicitly knowing the principal direction based parameterization $S(t,u)$.

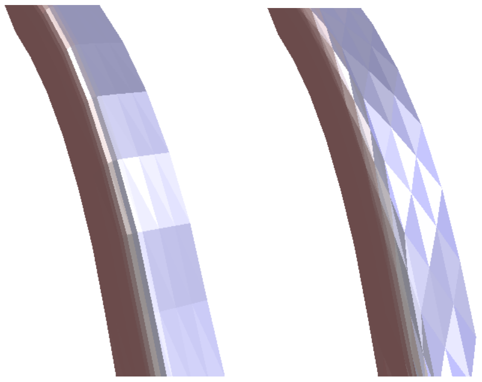
Figure 4: *In both images, the feature region is reconstructed by placing the samples along the trajectories. One the left, the samples are "synchronized" in the orthogonal direction (i.e. along the contours) as well, leading a noise free appearance. One the right, we shifted the phase on every other trajectory thus provoking extreme normal noise. Notice that the geometric approximation error is the same in both examples.*

Our goal is to generate the sampling grid using a *fishbone*–type of grid structure: we first construct a *backbone curve* T_0 that is approximately aligned *along* the feature and then we trace *rib curves* C_i that branch off perpendicularly from it (and hence are aligned *around* the feature). In terms of the last section, the backbone corresponds to a trajectory on the sweeping profile, while the ribs represent the contours at different time steps. On each rib C_i we take a set of uniformly spaced samples $\mathbf{p}_{i,j}$ (with respect to arc–length parameterization). If we connect the jth sample from every rib, we obtain a curve T_j with constant geodesic distance from the backbone. This implies that the curve T_j is another trajectory and it follows that the set of samples $[\mathbf{p}_{i,j}]_{i,j}$ has the properties derived in the previous section.

Consequently, the resampled triangle mesh patch is an anti–aliased approximation of the feature region that we can insert into a target mesh by using a mesh–stitching method similar to the one described in [14]. This target mesh does not need to be the same mesh as the one we sampled from. In the context of surface reconstruction from range data, e.g., we may generate the target model using standard methods [5, 2]. This mesh could afterwards be enhanced by stitching in alias–reduced patches that were resampled from the best available geometry, i.e. from the original non–decimated range scans.

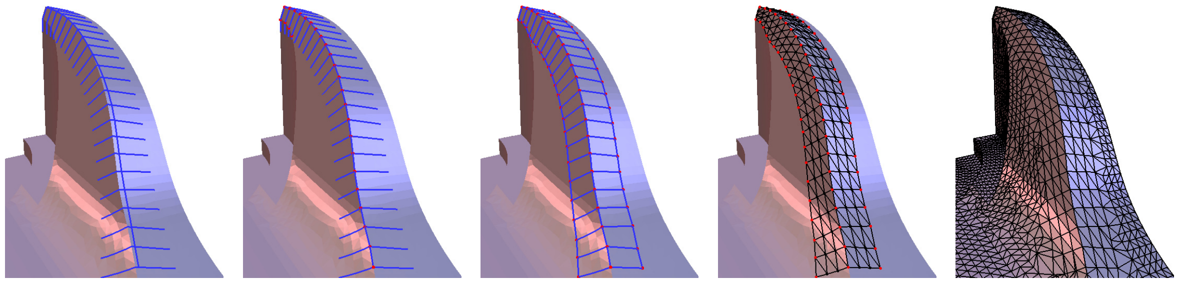

Figure 5: *This sequence of images gives a rough overview of our resampling procedure. The backbone is generated by interpolating user selected surface points and a set of ribs is created by intersecting the surface with a set of planes being orthogonal to the backbone (left). Feature snapping allows to re–position the backbone exactly on a sharp feature line (center left). Additional trajectories can be defined by marching from rib to rib according to some selection criterion (center). By uniformly sampling the rib curves, we can generate a regular triangulation of the feature region. The alignment of the sampling grid to the ribs in one direction and to the trajectories in the orthogonal direction guarantees an anti–aliased reconstruction (center, right). If necessary, the resampled mesh can be smoothed by applying univariate filters that preserve the feature characteristics. Finally the resampled mesh is stitched into the original (right).*

In the following we explain the basic steps of the resampling procedure in more detail:

The initial backbone is constructed interactively: the designer sketches the feature by picking a few positions on an estimated trajectory. The backbone curve is then generated automatically by smoothly interpolating or approximating these points. Since we do not require the backbone curve to lie exactly on the surface, this procedure allows us to obtain a smooth backbone curve even if the underlying surface data is noisy. The only (soft) requirement for the resulting backbone curve is that it should be an approximate offset of a trajectory.

To genererate the rib curves, the backbone is sampled either uniformly or with a curvature–dependent step width. For each of the sample points $\mathbf{v}_i$ we create a rib curve by intersecting the given surface with the plane positioned at $\mathbf{v}_i$ and orthogonal to the backbone's tangent (cf. Fig. 5, left). This special rib generation is the reason why the backbone does not have to lie exactly on the surface. Nevertheless, each rib is a *planar* polygon (a fact we will exploit later on) that *exactly* lies on the surface. If the given surface is a polygonal mesh, the plane intersection can be implemented by a local tracing scheme such that the computation costs do not depend on the overall complexity of the mesh.

To create a new trajectory the user selects one or more interpolation points on different ribs. Starting from such a point, the new trajectory is constructed by marching from rib to rib. The corresponding points on the neighboring ribs can be identified according to several different criteria:

- We can choose that point which has the same geodesic distance to an already existing trajectory (to mimic the offset curve property of trajectories).
- We can proceed in orthogonal direction to the current rib (to mimic the principal direction property of trajectories).

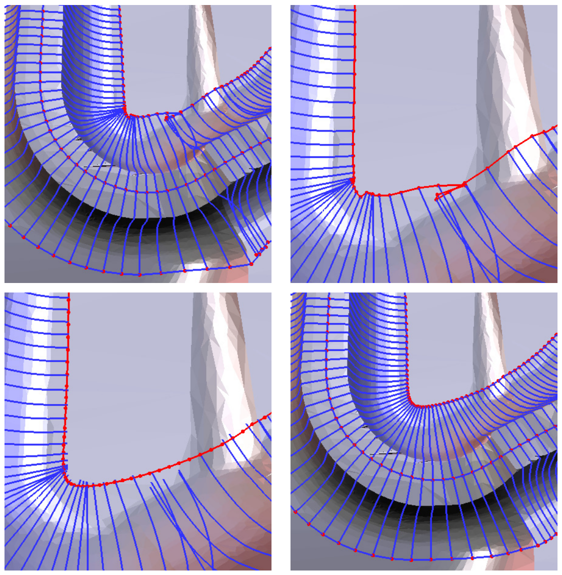

Figure 6: *At strongly curved features, the ribs may intersect each other. If we give up the requirement that the ribs have to be orthogonal to the trajectories, we can still find a decent triangulation with reduced normal noise.*

- We can choose the local curvature maximum to trace a sharp feature line. This is a convenient method to snap the backbone to a feature line if the initial backbone did not fit (cf. Fig 5, center left). Notice that the snapping only requires *univariate* feature detection within each rib curve.
- We can simply interpolate a given set of points by a smooth curve. This provides full manual control to the designer.

In case we use the backbone snapping we might have to recompute the ribs by intersecting the surface with a new set of planes being orthogonal to the new backbone (cf. Fig. 5, center left). We found this technique particularly useful in practice since it allows the designer to select sharp feature lines on a CAD model with a high precision without having to pick points exactly on the feature by himself.

When the feature is strongly curved we may run into the problem of overlapping rib curves. To generate an anti–aliased triangulation in this case, we have to give up the requirement that ribs have to be orthogonal to trajectories. The following technique yields very satisfactory results: For a given fishbone (= backbone samples $\mathbf{v}_i$ plus ribs) we generate two additional trajectories $[\mathbf{w}_i]$ and $[\mathbf{w}'_i]$ with constant distance along the ribs. In the presence of overlapping ribs, these trajectories will have kinks and loops. By applying a simple low–pass filter operator to the outer trajectories we can straighten out these degeneracies.

After the filtering, the three points $\mathbf{v}_i$, $\mathbf{w}_i$, and $\mathbf{w}'_i$ which are associated with the ith rib, define a tilted plane. By intersecting the given surface with these new planes, we obtain new ribs which no longer overlap (cf. Fig. 6).

The last step of the resampling procedure is to compute equidistant samples on each rib with respect to the arc–length parameterization. Those samples have the property that they are (trivially) aligned to the ribs (= contours) but also aligned to the trajectories since the jth sample on each ribs has the same geodesic distance to any other trajectory. Hence the sampling grid matches the requirements from Section 3 and therefore we can expect an anti–aliased surface reconstruction. The resampling procedure is concluded by stitching the new patch into the original mesh.

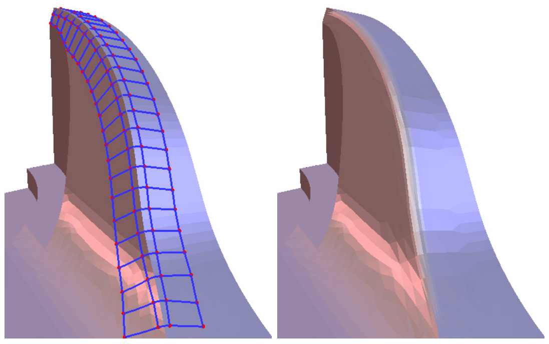
Figure 7: *Rounding a sharp feature: The feature region has been resampled by our new technique. The image on the left shows four trajectories and several ribs. The area between the inner trajectories is modified by replacing each rib segment with a Hermite interpolating profile having a prescribed blend radius (right image).*

5. Feature modeling

The fishbone metaphor described in the last section not only enables us to resample geometry in a way that strongly reduces normal noise, the additional structural information can also be used for high level feature modeling.

Changing the characteristics of a feature is a very frequent operation in product design. For example in CFD simulations it is often necessary to vary the sharpness of a feature (i.e., the radius of a rolling ball blend) to verify the impact on the overall aero dynamics. *Rounding* and *sharpening* are the operations which increase or decrease the blend radius along a feature.

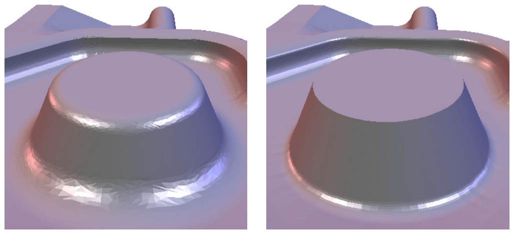
Figure 8: *The rounded feature is sharpend by setting the blend radius for the rib profiles to zero for the upper ring and to some small but non-zero value in the lower ring.*

On a fishbone–wise resampled feature region such modeling operations are very easy to implement. The reason for this is that we have a perfect alignment of the sampling grid to trajectories and ribs. Since the ribs are generated by plane intersections we additionally know that they are planar which reduces the feature modeling operations to 2D operations acting on the rib curves.

The generic formulation of a feature modeling operation is to select two trajectories, remove the part of the fishbone that lies between them and replace it by another mesh that fits to the boundary conditions imposed by the remaining parts. Since each rib can be processed independently we only have to implement the modeling operation for a planar curve and then apply it to each rib separately.

The corresponding 2D operation to which the feature modeling reduces is in fact a Hermite interpolation problem where two points and tangents are given and a C^1 interpolating curve is sought. These Hermite conditions are imposed by the remaing parts of each rib.

Let us first consider the rounding or sharpening operations. As stated above, both operations simply change the radius of the blend and hence they can be treated analoguously. The generic profile by which we want to solve the Hermite interpolation problem is depicted in Fig. 9. It consists of two straight segments on both sides of a circular arc. This generic profile is flexible enough to solve the Hermite problem for any convex configuration and has one additional degree of freedom: the circle radius. Hence we can prescribe

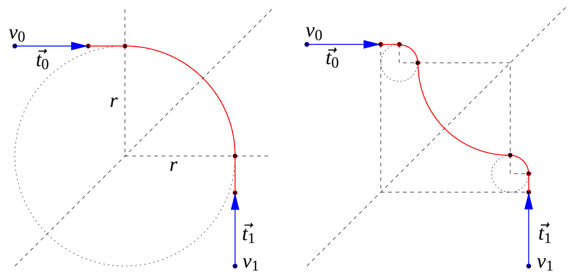

Figure 9: *We can use any generic profile for the feature modeling as long as it is flexible enough to solve the Hermite interpolation problem and provides at least one additional degree of freedom.*

the radius and find the resulting C^1 interpolant by a straightforward construction. Fig. 7 shows an example where the blend radius is increased and Fig. 8 shows the perfect reconstruction of a sharp feature by setting the blend radius to zero.

If we want to model more complicated profile sweeps, we simply have to replace the generic Hermite interpolant. Fig. 9 shows a parameterized profile with two straight segments and three circular arcs. The independent degrees of freedom are the circle radii and the opening angle of the center arc. Fig. 10 shows the application of this profile to the same feature that was modified in Fig. 7.

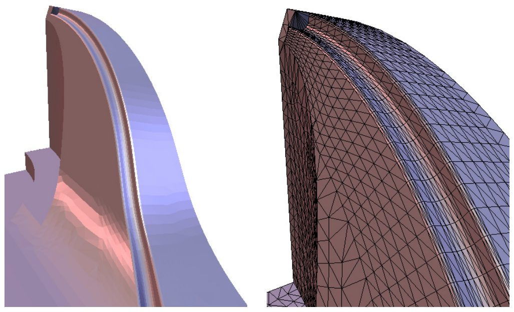

Figure 10: *Quite sophisticated modeling operations are possible if we use more complicated profiles for the 2D Hermite interpolation on each planar rib curve.*

6. Results

We applied the surface anti–alias technique in the context of CFD simulation for conceptual car design to a highly detailed mesh model of the BMW Z8 car. The normal noise contained in the models after the triangulation and decimation phase could effectively be removed. Some results are shown in Fig. 11∗. To re–model the complicated structure of features around the driver's window took a one–hour interactive session. Once the fishbones have been created for each feature, the blend radii can be changed interactively.

7. Conclusions

We demonstrated a new resampling and anti–aliasing scheme for the feature regions of a given surface. The major idea is to align the sampling grid to the (estimated) principal curvature directions of a sweep surface in order to minimize the normal noise. We presented a geometrical justification for the intuitive placement of the sample points along trajectories and contours of a rounded feature: it turns out that this special sampling pattern minimizes the normal noise which is measured in terms of the variation of normal jumps across mesh edges. This discrete fairness measure can be interpreted as a third order derivative (curvature variation) which is often used in CAGD to measure the fairness of a surface.

8. Acknowledgements

We would like to thank the BMW AG, Munich and Dr. Wolf Bartelheimer for cooperating and supporting this research project, and for providing us the Z8 data.

References

1. N. Amenta, M. Bern, M. Kamvysselis. *A new Voronoi-based surface reconstruction algoritm.* Proc. SIGGRAPH '98, 1998. 1

2. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin. *The Ball-Pivoting Algorithm for Surface Reconstruction.* to appear in IEEE Trans. on Vis. and Comp. Graph. 1, 5

3. M. P. do Carmo. *Differential Geometry of Curves and Surfaces* Prentice–Hall, Inc Englewood Cliffs, New Jersey, 1993. 4

4. P. Cignoni, C. Montani, C. Rocchini, R. Scopigno *External Memory Management and Simplification of Huge Meshes.* Preprint. 2

5. B. Curless, M. Levoy. *A volumetric method for building complex models from range images.* Proc. SIGGRAPH '96, 1996, 303–312. 1, 5

6. M. Desbrun, M. Meyer, P. Schröder, A. H. Barr. *Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow.* Proc. SIGGRAPH '99, 1999, 317–324. 3

7. H. Edelsbrunner, E. P. Mücke. *Three-dimensional alpha shapes.* ACM Trans. Graphics 13, 1994, 43–72. 1

8. M. Garland, P. S. Heckbert. *Surface Simplification using quadric error metrics.* Proc. SIGGRAPH '92, 1992, 71–78.

9. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. *Surface Reconstruction from Unorganized Points.* Proc. SIGGRAPH '92, 1992, 71–78. 1

10. H. Hoppe. *Progressive Meshes.* Proc. SIGGRAPH '96, 1996, 99–108. 1

11. J. Hoschek, D. Lasser. *Fundamentals of Computer Aided Geometric Design* AK Peters, Ltd., 1993. 1

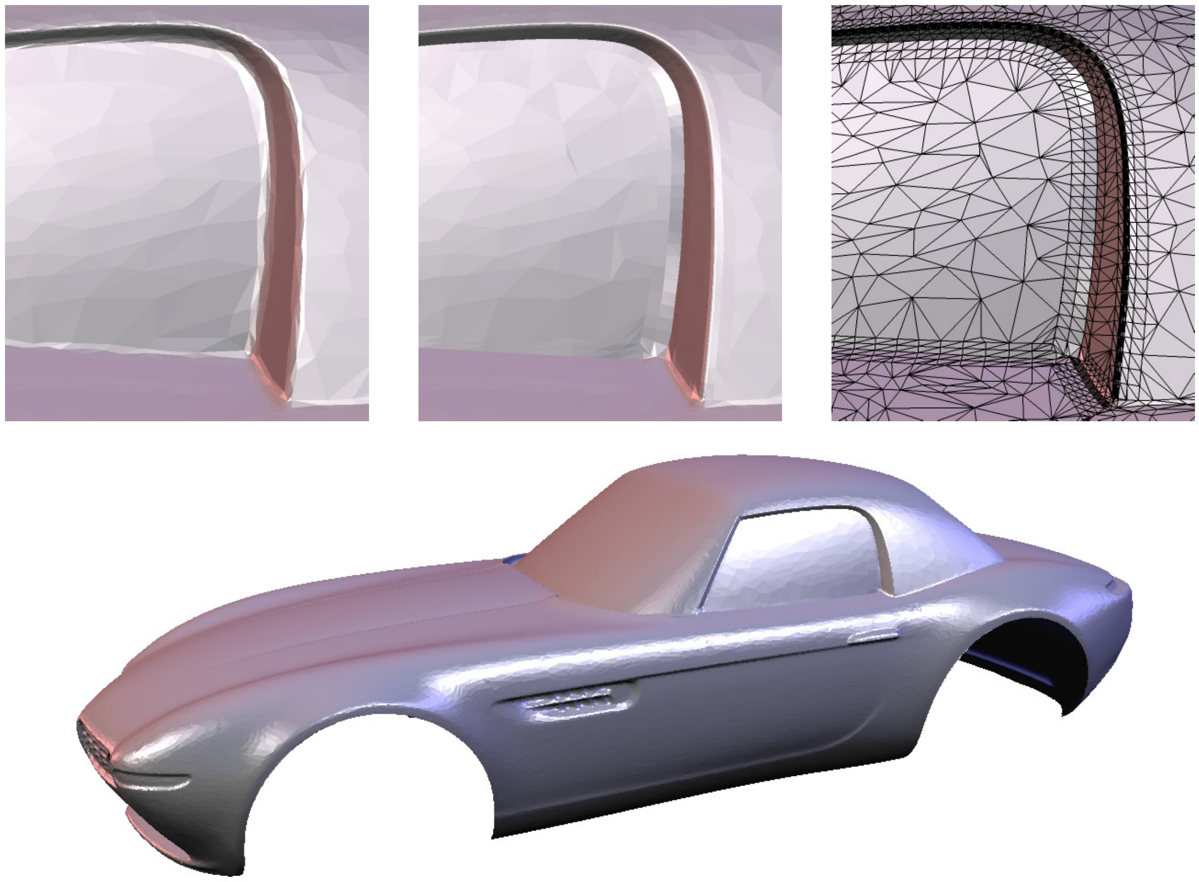

Figure 11: *We applied the resampling and surface anti–aliasing technique to a detailed BMW Z8 model which is supposed to be used for CFD simulation. The normal noise in the vicinity of the feature regions of the decimated model can cause severe numerical instabilities. In the remeshed feature and blend regions around the driver's window the normal noise is almost completely removed. The other parts of the model have not been resampled.*

12. L. Kobbelt, S. Campagna, J. Vorsatz, H.–P. Seidel. *Interactive Multi–Resolution Modeling on Arbitrary Meshes.* Proc. SIGGRAPH '98, 1998, 105–114. 3

13. L. Kobbelt, S. Campagna, H.–P. Seidel. *A General Framework for Mesh Decimation.* Graphics Interface, 1998, 43–50. 1

14. L. P. Kobbelt, M. Botsch. *An Interactive Approach to Point Cloud Triangulation* Proc. Eurographics '00, 2000, 479–487. 5

15. W. Light, *Advances in Numerical Analysis – Vol. II: Wavelets, Subdivision Algorithms, and Radial Basis Functions.* Oxford University Press, 1992. 1

16. P. Lindstrom. *Out-of-core simplification of large polygonal models.* Proc. SIGGRAPH '00, 2000, 259–262. 2

17. C. Rössl, L. P. Kobbelt. *Approximation and Visualization of Discrete Curvature on Triangulated Surfaces.* Proc. of 4th Conf. on Vision, Modeling, and Visualization (VMV-99), 1999, 339–346. 3

18. Remi Ronfard, Jarek Rossignac. *Full–range approximations of triangulated polyhedra* Proc. Eurographics '96, 1996, 67–76. 1

19. G. Taubin. *A signal processing approach to fair surface design.* Proc. SIGGRAPH '95, 1995, 351–358. 3

20. G. Taubin. *Geometric Signal Processing on Polygonal Meshes.* Proc. Eurographics '00, State of the Art Reports, 2000. 3

21. T. Várady, R. R. Martin, J. Cox. *Reverse engineering of geometric models — an introduction.* CAD **29** (1997), 255–268. 1

22. T. Várady, P. Benkő. *Reverse Engineering B-rep Models from Multiple Point Clouds.* Geometric Modeling and Processing 2000, 3–12. 1

23. Z. Wood, M. Desbrun, P. Schröder, D. Breen. *Semi-Regular Mesh Extraction from Volumes,* Proc. of the 11th Ann. IEEE Visualization Conference (Vis) 2000 1