Feature Sensitive Sampling for Interactive Remeshing

Mario Botsch Christian Rössl Leif Kobbelt

Max-Planck Institut für Informatik, Computer Graphics Group
Im Stadtwald, 66123 Saarbrücken, Germany
Email: {botsch,roessl,kobbelt}@mpi-sb.mpg.de

Abstract

We present a technique for remeshing irregular triangles meshes where the distribution and alignment can be adapted to the underlying geometry. Following the interactive *virtual range scanner* approach we overcome aliasing problems by introducing a special sampling technique. A sampling grid that can be aligned to the local features of the mesh is constructed interactively in an intuitive way and without adding reasonable overhead to the virtual scanning process.

1 Introduction

Unstructured triangles meshes can represent surfaces of arbitrary shape and topology. That is why they are the most versatile surface representation in computer graphics. Triangulated surfaces can be generated from many different sources ranging from CAD models that consist of polygons and NURBS-patches to point clouds that are obtained from range scanners.

There are different applications that require different properties on the meshes. So one may use various triangulations of the same surface each of which is suited for a specific application. As a consequence there is a demand for techniques that convert or remesh "bad" meshes to "good" meshes for the purpose of such applications.

In this paper we present an interactive remeshing technique. Our aim is to generate highly regular meshes, i.e. most vertices have valence six. The triangles should also be as "round" or equilateral as possible. Consider an input mesh as the one in figure 6 (top). The mesh is highly irregular, the vertex valences mostly differ from six, and there are ill shaped triangles. Such meshes are a typical output of a decimation algorithm. We want to convert this mesh to an almost regular mesh. While the irregular but compact mesh may be preferred for fast rendering, regular meshes are much more appropriate e.g. for simulations using FEM and CFD.

We use an interactive approach for remeshing. This allows great flexibility and a maximum of user control over the remeshing process. The *virtual range scanner* approach from [3] has proven to be highly intuitive and efficient. E.g. [4, 5] describe hierarchical algorithms for producing meshes with subdivision connectivity while preserving distinct features. In contrast our algorithm does not generate a subdivision connectivity but an arbitrary mesh. It does not need global information (parameterization), and it can be applied locally to parts of a mesh.

The main contribution of this paper is a technique for avoiding aliasing artifacts in virtual range scans. [3] use a regular sampling grid that allows exploiting the graphics hardware by depth-buffer sampling. We extend this work by defining a special sampling grid that can be aligned to the underlying geometry, especially to curved feature lines. As a consequence we obtain samples that can

be triangulated regularly with greatly suppressed aliasing effects. A special fishbone [8] like structure is used for defining feature regions in an intuitive way. These "fishbones" are defined by lines of curvature on the surface if available or they may be constructed manually. Only few overhead is added to the process of virtual range scanning while high quality and very regular triangle meshes are obtained.

The following section of the paper gives a brief overview over the virtual range scanner technique. Then we describe our new approach of feature sensitive sampling. In the last sections we present some results and conclusions.

2 Virtual range scanning

During a scanning session the input mesh is rendered in an OpenGL window so that the user can look at it from arbitrary positions. The outcome of the drawing process is used to generate a scan that gets sewed into previous scans afterwards [3]. Taking one virtual range scan consists of the following steps:

1. Choosing the viewing (scanning) position and direction by rotating, translating and zooming the input mesh.
2. Scanning this view, leading to a mesh that approximates the input mesh as seen from the current position.
3. Masking the new mesh.
4. Merging the new mesh with the mesh derived from previous scans.

2.1 Scanning

When rendering the geometry to be scanned a depth value $z := z(x, y)$ for each pixel $(x, y)^\top$ is automatically stored in the z-buffer of the graphics hardware. Therefore we get the screen space coordinates $(x, y, z)^\top$ of the pixel without any extra computation. These coordinates are then unprojected to world coordinates $(x', y', z')^\top$ by the inverse of the rendering transformations. These are the modelview matrix $\mathbf{M}$, the projection matrix $\mathbf{P}$ and finally the window-to-viewport mapping $\mathbf{V}$:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} := (\mathbf{VPM})^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

There is an implicit rounding in the window-to-viewport mapping when assigning real valued coordinates to integer pixel coordinates. This rounding can cause a global offset for the generated mesh. To minimize this effect we unproject the center $(x+\frac{1}{2}, y+\frac{1}{2}, z)^\top$ of the pixel instead of its lower-left corner $(x, y, z)^\top$ by using the inverse $(\mathbf{V}')^{-1}$ of a modified $\mathbf{V}$:

$$(\mathbf{V}')^{-1} := \begin{pmatrix} \frac{2}{w} & 0 & 0 & \frac{1}{w} - 1 \\ 0 & \frac{2}{h} & 0 & \frac{1}{h} - 1 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Using this technique we generate a new vertex for each pixel that was drawn onto the screen. These new vertices can now trivially be triangulated because of the underlying grid structure of the frame- and z-buffer.

With this flexible and intuitive interface the user can control the size and the alignment of the output triangles. Zooming in on the input mesh results in a finer sampling. Therefore we get a hardware based sub-sampling algorithm for free.

If feature lines of the input geometry are aligned either horizontally or vertically on the screen the resulting scanned triangles will have their edges aligned to these features (cf. figure 1). While this alignment works perfectly for straight feature lines it must fail for curved ones. This is where our feature based sampling will help.

2.2 Masking

When remeshing complicated geometry it is often useful not to scan the whole input mesh but to restrict the scanning to a certain area. This is done by using an intuitive "lasso" function to select the *region of interest*. All new vertices falling outside of this

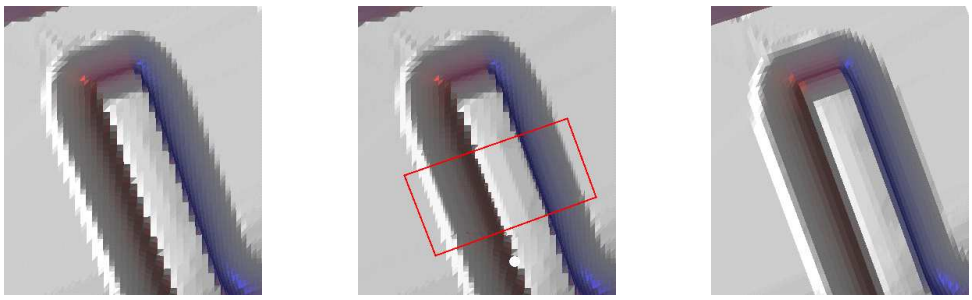

Figure 1: At sharp feature lines, any sampling algorithm tends to generate alias artifacts (left) which cannot be removed by refining the sampling density (center). However, aligning the sampling grid to the feature improves the visual quality.

user-defined area will be discarded. This can be used for including a small high-resolution patch into a coarser mesh. We just take a coarse scan of the whole object, select the region-of-interest and re-scan a zoomed version of this area (cf. figure 2).

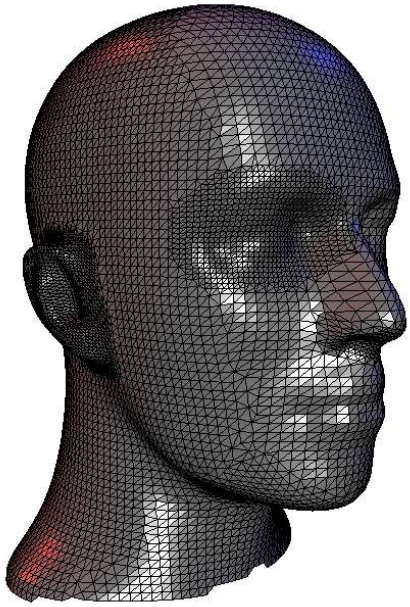

Figure 2: The interactive approach provides an intuitive interface to locally adapt the mesh resolution. Here the ear and the eye have been z-buffer scanned with a higher density.

Additionally an automatic *sampling quality* mask is applied to the remaining vertices. In the overlapping parts of the meshes this masking ensures that the better sampling of this area will replace the other one. To check this, every new vertex gets projected onto the old mesh. If a new vertex hits the old mesh, its quality is compared to the quality of the triangle it projects to, and the better one is kept.

The quality of a vertex or a triangle is defined in terms of edge length and the sampling angle. This is a very natural way to define sampling quality because the smaller the sampling angle and the shorter the distance between sampling points gets, the more reliable the sampling will be.

The projection of the new mesh onto the old one can be quite expensive, since for every new vertex the old triangle that lies closest to it has to be found. This results in a time-consuming quadratic search. The common way to cope with this problem is to use appropriate space partitioning structures like e.g. octrees or BSP-trees. If we want to handle huge input data we have to take care of memory consumption. Therefore we again utilize the graphics hardware to accelerate this projection by so-called *ID-rendering*. Every triangle in the old mesh is uniquely indexed by its ID that can be encoded as a RGB-color. Using this technique we can render up to $2^{redBits+greenBits+blueBits}$ distinguishable triangles. If this amount is not sufficient we can increase it by using the front- and back-buffer or do multi-pass rendering.

The vertex to be projected is now transformed to screen coordinates by the graphics hardware and we can identify the correspond-

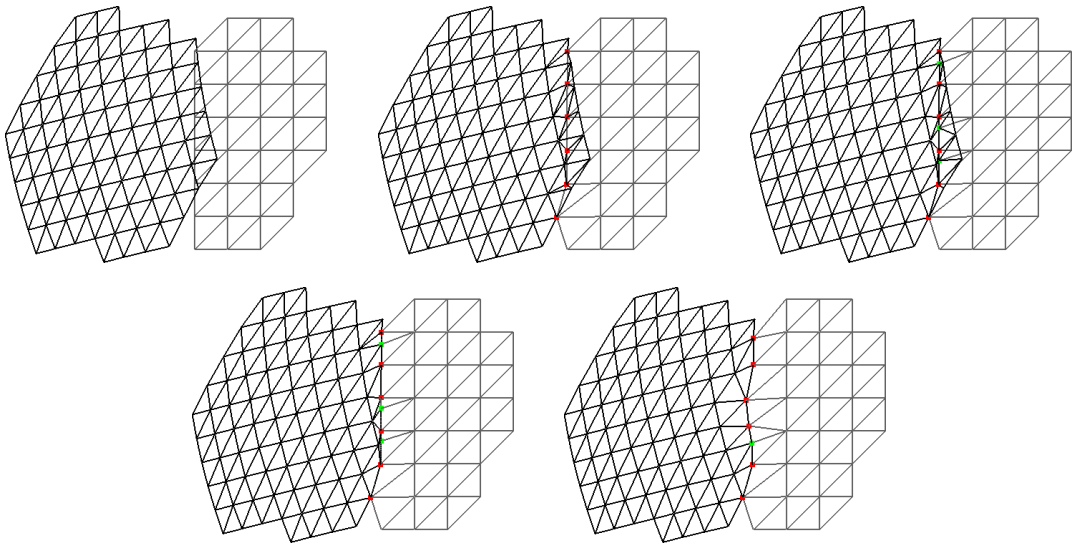

Figure 3: The stitching algorithm first inserts the new vertices into the old mesh then splits or flips edges of the old mesh to interpolate the new mesh's boundary edges and finally removes the redundant part of the old mesh. Collapsing short edges in the resulting mesh removes badly shaped triangles (from left to right).

ing triangle be just reading out the RGB-color at this pixel of the frame-buffer. This may not exactly be the nearest triangle, since especially near the contour of the data many triangles get rendered into a small portion of the screen. However the found triangle is a very good first-guess for a local search on the mesh. So we have reduced the projection problem from a quadratic search to one rendering step, this corresponds to bucket-sorting.

After this two masking operations only those pixels remain that lie within the user-defined *region-of-interest* and pass the *quality test* in overlapping regions. To avoid degenerated triangles and gaps after the stitching, we conclude the masking phase by applying morphological *erosion* and specialized *dilation* operators on the set of remaining pixels (cf. [1]). This can be done quite efficiently since we still have the pixel-grid parameterization.

2.3 Stitching

In the final step the acquired scan has to be merged with the existing mesh that was derived by previous scans. This is done by a slightly modified mesh zippering algorithm (cf. [9]).

We first insert the boundary vertices of the new mesh into the corresponding triangles of the old mesh. We then insert the boundary edges of the new mesh by intersecting or flipping edges of the old mesh to generate a common boundary polygon on both meshes. After this we remove the part of the old mesh that is bounded by this polygon and insert the new mesh. In order to improve the mesh quality we post-optimize the seam area by collapsing and flipping bad edges (cf. [2]). The whole process is depicted in figure 3.

3 Feature based sampling

The original virtual range scan algorithm in [3] samples the geometry over a regular grid. This allows the exploitation of the graphics hardware by using depth-buffer sampling. Aliasing artifacts are reduced by aligning the view to feature lines. This works well for straight features but still produces noticeable alias effects in curved feature regions. Multiple iterations of sequent aligning, masking,

scanning and stitching is not an appropriate option to cope with aliasing.

We extend this technique by introducing an alternative sampling grid that can be aligned to curved feature regions and thus suppresses aliasing. This grid can be set up from streamlines following the principle curvature directions or completely manually. Both ways only few curves must be provided. The actual grid is constructed from interpolating between these curves. In this section we use the *fishbone* metaphor from [8] for this construction process.

According to needs the user can choose either the standard regular or the alternative curved grid. The first option may be appropriate for most parts of the mesh and provides excellent performance due to hardware support. In contrast, our fishbone grid provides additional flexibility in feature regions but a ray casting scheme will be used for sampling.

3.1 Setting up a sampling grid

Feature regions are selected, resampled and triangulated one after another. The stitching algorithm merges a resampled submesh with the original mesh replacing "bad" mesh regions. The resulting mesh is the new input when fixing the next feature region.

Let us assume that we can calculate curvature information in every vertex of the input mesh, e.g. the method in [7] can be used. Then a continuous vector field can be defined on the surface by using barycentric interpolation on one of the principle directions. (We choose the direction towards the maximum curvature.) We then project the direction field to the current viewing plane – one direction per pixel – allowing efficient streamline tracing in 2D.

This interpolation and projection step can be done very elegantly by the underlying rasterization engine which maps the 3D mesh to the frame buffer: the vector valued attributes can be encoded as RGB colors and assigned to the mesh vertices. Rendering the mesh without any shading then generates a frame-buffer pixel matrix which stores the interpolated values. Arbitrary precision can be obtained by appropriate scaling and multi-pass rendering. Finally, a normalization step is done and we end up with a direction value in every pixel.

Now feature lines are defined to follow the principal directions. As a consequence an ideal sampling grid is spanned by lines of curvature. As mentioned, we can trace these lines entirely in 2D.

We build this grid in a *fishbone* fashion by first picking a *backbone* and then arranging the *ribs* in orthogonal direction.

To define the backbone, the user can pick an arbitrary point in the interior of the current segment. Starting at that point we compute the backbone curve by integrating along the *minimum* curvature direction field. On the backbone curve we distribute samples with constant arc-length distance. The orthogonal ribs are then computed by starting at those samples and integrating along the *maximum* curvature direction. Depending on the geometry minimum and maximum curvature can be exchanged.

In some cases, however, this simple rib generation technique fails to produce useful grid lines. Due to noise and the discretization of the direction field, neighboring lines can merge which destroys the global fishbone structure. In order to avoid this effect, the user can manually place a few sample points on the backbone from where the *key-ribs* are traced along the maximum curvature directions. Inbetween these key-ribs we uniformly distribute additional *blended ribs*. The blended ribs result from interpolating between the key-ribs. By properly choosing the starting points for the key-ribs, we generate a high quality set of pseudo-parallel grid lines (figure 4) in a few seconds.

Each grid line is represented by a polygon with constant edge length h (arc-length parameterization). To uniquely describe the shape of a rib grid line we need the first polygon edge E_0 and a sequence of angles α_i between successive edges E_i and E_{i+1}. The complete grid line can then be reconstructed by

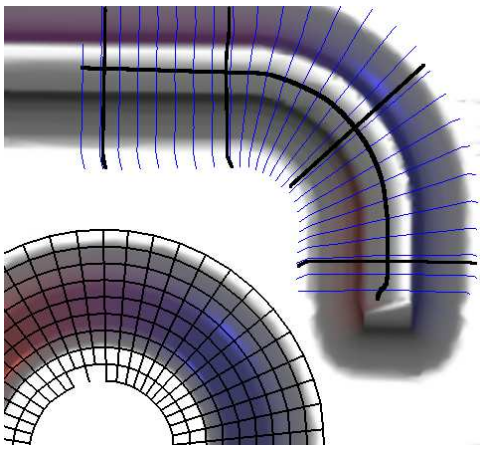

Figure 4: Example for the generation of grid lines. The fish bone in the top part is defined by the dark streamlines: the backbone along the principal direction towards the maximum curvature and the key-ribs towards the minimum curvature. (The middle key-rib was defined by a straight line.) The blended ribs are drawn in a light color. The bottom part shows the sampling grid obtained from a fishbone. Here, the backbone was defined by a streamline towards the maximum curvature while the key-ribs are lines into the minimum curvature direction at the joints to the backbone.

starting with the first edge E_0 and adding more edges E_{i+1} with the same length in the direction determined by the angles α_i. In the grid line representation $[E_0, \{\alpha_i\}]$, the edge E_0 determines the *orientation* and the sequence $\{\alpha_i\}$ determines the characteristic shape (figure 5).

Assume we are given two key-ribs $[E_0, \{\alpha_i\}]$ and $[F_0, \{\beta_i\}]$ which start on the same backbone. Then for every value $t \in [0, 1]$ we find a new starting point on the backbone arc between the two key-ribs and the corresponding blended rib is given by $[G_0, \{\gamma_i\} = \{(1-t)\,\alpha_i + t\,\beta_i\}]$ where the orientation G_0 is given by an weighted average of E_0 and F_0 and the characteristic shape is a weighted blend of the two key-ribs. Using this blending technique we can generate very good fishbone type grid lines by prescribing only rather few key-ribs.

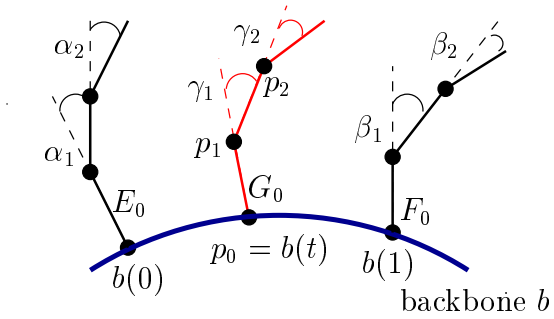

Figure 5: The blending of strokes is based on a decomposition into *orientation* (E_0,F_0) and *characteristic shape* ($\{\alpha_i\}$,$\{\beta_i\}$) and controlled by a parameter $t \in [0, 1]$ with the backbone curve b locally parameterized as shown.

After the rib generation, the fishbone structure is given by a set of polygons with unit edge length h. The kth vertex $\mathbf{p}_k^{(l)}$ of the lth rib $R_l = [G_0, \{\gamma_i\}]$ can be computed by

$$\mathbf{p}_k^{(l)} = \mathbf{p}_0 + G_0 + h \sum_{i=1}^{k-1} \begin{pmatrix} \cos(\sum_{j=1}^{i} \gamma_j) \\ \sin(\sum_{j=1}^{i} \gamma_j) \end{pmatrix},$$

or

$$\mathbf{p}_k^{(l)} = \mathbf{p}_0 - G_0 - h \sum_{i=1}^{-k-1} \begin{pmatrix} \cos(\sum_{j=1}^{i} \gamma_{-j}) \\ \sin(\sum_{j=1}^{i} \gamma_{-j}) \end{pmatrix}$$

for negative k.

Note that the user must take care that the blended ribs do not intersect, i.e. a minimum of key-ribs must be specified depending on the curvature of the backbone. Intersecting ribs will lead to back facing triangles that are ignored in the triangulation step.

In practice it may be infeasible to strictly use stream lines towards the minimum (backbone) or the maximum (key-ribs) curvature. Reasons could be noise and/or occlusion artifacts on the projected direction field, or a direction field that does not allow to construct a long backbone that follows the feature. Then the backbone and/or single key-ribs may be specified manually. This can be done by drawing curves (e.g. interpolating spline curves) in 2D. In fact, information from the frame buffer may be utilized to make this step easier and somewhat smarter by using standard image processing techniques like [6].

3.2 Sampling

The organization of the rib vertices $\mathbf{p}_k^{(l)}$ to a sequence of sequences $[[\mathbf{p}_k^{(l)}]_k]_l$ corresponds to a rectilinear matrix type structure where the vertices of one rib form a row $[\mathbf{p}_k^{(l)}]_k$ and the kth vertex for all ribs $[\mathbf{p}_k^{(l)}]_l$ form a column.

We now use this structure for sampling and triangulation. By casting rays from the eye point through the rib vertices onto the original mesh we get high quality samples. An initial ID-rendering step efficiently provides a first guess on what triangle will be hit by the ray. Only an inexpensive local search is necessary for finding intersections. Alternatively, advanced data structures like e.g. a BSP-trees of the original mesh can be used.

4 Results

Our remeshing method generates regular submeshes that are stitched into the input mesh replacing irregular regions. For demonstration we have remeshed two feature regions of a mesh (figure 6) where a naive aligning technique would not help. One clearly recognizes aliasing in the upper left part of the mesh compared to the feature sensitive remeshed upper right part. The remaining flat parts have been remeshed with the standard virtual range scanning approach. A total of three scans – two of them feature sensitive – have been used to create the bottom mesh.

5 Conclusions

We presented a new method for remeshing arbitrary meshes to highly regular ones. By extending the interactive *virtual range scanner* [3] approach our new algorithm inherits all its intuitiveness and flexibility while aliasing problems are reduced to a minimum. This is achieved by constructing a sampling grid that adapts to feature lines on the input mesh rather than using the standard pixel grid. As a consequence our technique produces high quality meshes also for regions with curved feature lines.

The construction of the feature sensitive sampling grid introduces only low overhead to the virtual scanning process. The presented fishbone approach may take advantage of curvature information on the input mesh that indicates feature regions. So grid lines (key-ribs) are generated automatically. Due to blending, only a few of these key-ribs are necessary to define a sampling grid.

Acknowledgements

This work was partially supported by the BMW AG, Munich.

References

[1] R. M. Haralick, S. R. Sternberg, X. Zhuang. *Image Analysis Using Mathematical Morphology.* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol PAMI-9, No. 4, 1987, pp. 532–552

[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. *Mesh Optimization.* Proc. SIGGRAPH'94, 1994, 19–26.

[3] L. Kobbelt, M. Botsch. *An Interactive Approach to Point Cloud Triangulation.* to appear in Computer Graphics Forum (Proc. EUROGRAPHICS'2000). 2000

[4] L. Kobbelt, J. Vorsatz, U. Labsik, H-P. Seidel. *A Shrink Warpping Approach to Remeshing Polygonal Surfaces.* Computer Graphics Forum (Proc. EUROGRAPHICS'99), 1999, pp. 119–129

[5] A.W.F. Lee, W. Sweldens, P. Schröder,L. Cowsar, D. Dobkin. *MAPS: Multiresolution Adaptive Parameterization of Surfaces.* Computer Graphics (Proc. SIGGRPAH'98), 1998, pp. 95–104

[6] E.N. Mortensen, W.A. Barrett. *Intelligent Scissors for Image Composition.* Computer Graphics (Proc. SIGGRPAH'95), 1995, pp. 19-1-198

[7] C. Rössl, L. Kobbelt. *Approximation and Visualization of Discrete Curvature on Triangulated Surfaces.* Proc. Vision, Modeling and Visualization'99, 1999, pp. 339–346

[8] C. Rössl, L. Kobbelt. *Line-Art Rendering of 3D-Models.* submitted, 2000

[9] G. Turk, M. Levoy. *Zippered Polygon Meshes from Range Images.* Computer Graphics (Proc. SIGGRAPH'94), 1994.

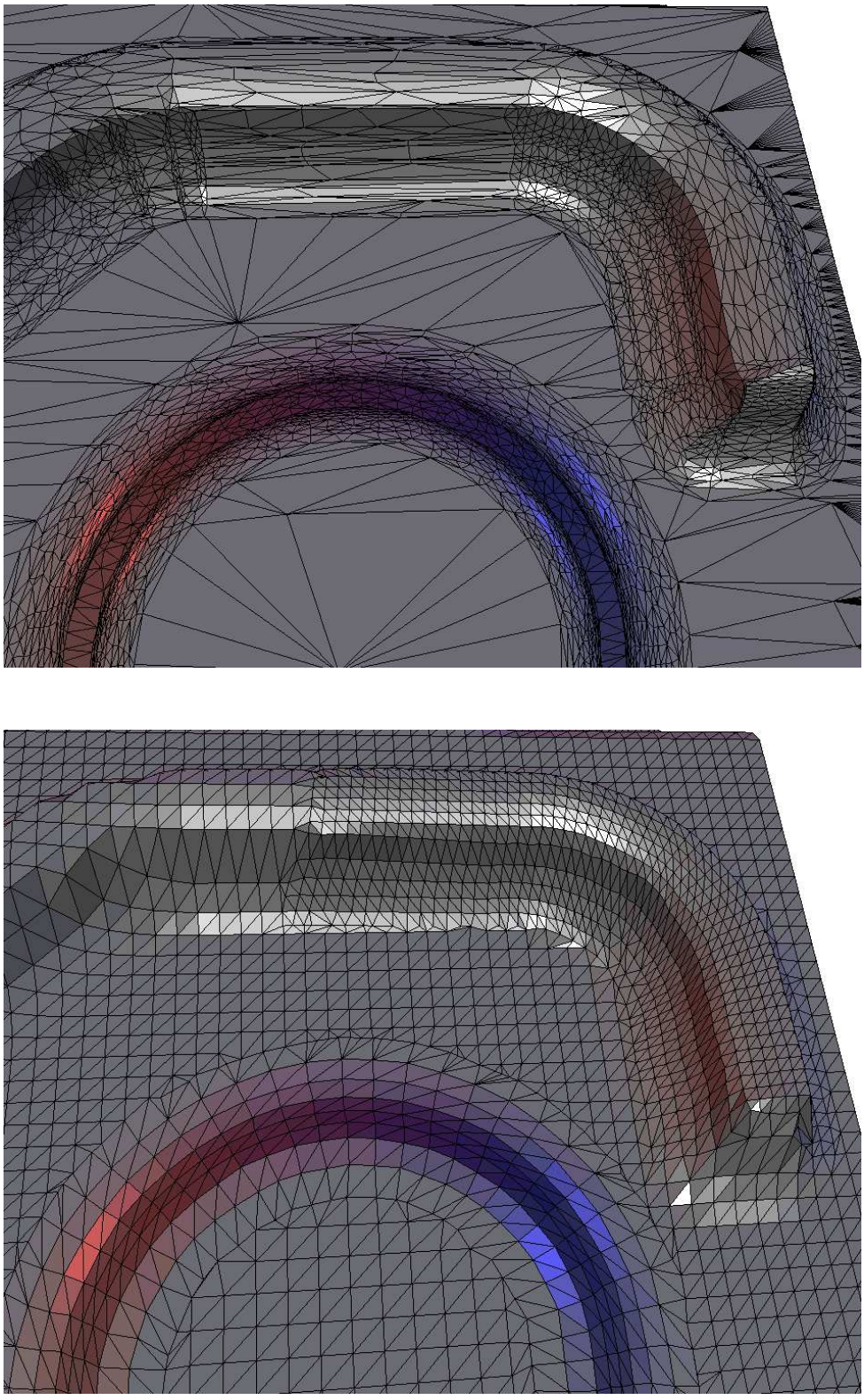

Figure 6: Top: "bad", irregular mesh. Bottom: Two feature regions (in the upper right and bottom part of the mesh) have been resampled with our feature sensitive method. Notice the difference between naive aligning of a regular grid (upper left corner) to the use of a curved sampling grid (upper right corner). The remaining flat parts have been processed with standard virtual range scans.