Anisotropic Polygonal Remeshing

Pierre Alliez
INRIA Sophia-Antipolis

David Cohen-Steiner
INRIA Sophia-Antipolis

Olivier Devillers
INRIA Sophia-Antipolis

Bruno Lévy
INRIA Lorraine

Mathieu Desbrun
U. of So. California

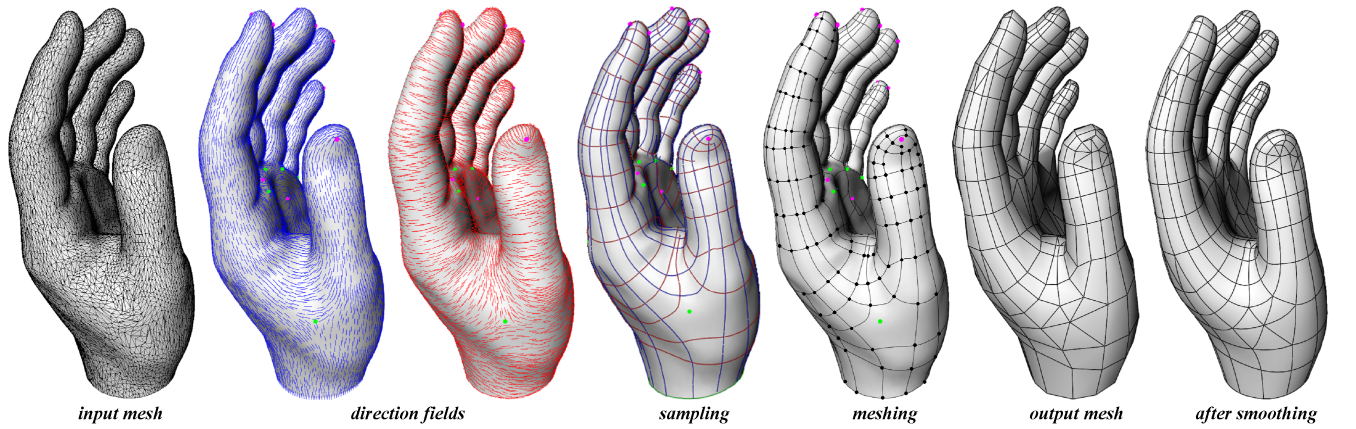


Figure 1: *From an input triangulated geometry, the curvature tensor field is estimated, then smoothed, and its umbilics are deduced (colored dots). Lines of curvatures (following the principal directions) are then traced on the surface, with a local density guided by the principal curvatures, while usual point-sampling is used near umbilic points (spherical regions). The final mesh is finally extracted by subsampling, and conforming-edge insertion. The result is an anisotropic mesh, with elongated quads aligned to the original principal directions, and triangles in isotropic regions. Such an anisotropy-based placement of the edges and cells makes for a very efficient and high-quality description of the geometry. A smooth surface can be obtained by quad/triangle subdivision of the newly generated model.*

Abstract

In this paper, we propose a novel polygonal remeshing technique that exploits a key aspect of surfaces: the intrinsic *anisotropy* of natural or man-made geometry. In particular, we use curvature directions to drive the remeshing process, mimicking the lines that artists themselves would use when creating 3D models from scratch. After extracting and smoothing the curvature tensor field of an input genus-0 surface patch, lines of minimum and maximum curvatures are used to determine appropriate edges for the remeshed version in anisotropic regions, while spherical regions are simply point-sampled since there is no natural direction of symmetry locally. As a result our technique generates polygon meshes mainly composed of quads in anisotropic regions, and of triangles in spherical regions. Our approach provides the flexibility to produce meshes ranging from isotropic to anisotropic, from coarse to dense, and from uniform to curvature adapted.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations.

Keywords: surface remeshing, anisotropic sampling, polygon meshes, lines of curvatures, tensor fields, approximation theory.

1 Introduction

Despite a recent effort to make digital geometry tools robust to arbitrarily irregular meshes, most scanned surfaces need to undergo complete remeshing (alteration of the sampling and of the connectivity; see [Turk 1992; Eck et al. 1995; Hoppe 1996; Lee et al. 1998; Kobbelt et al. 1999; Botsch and Kobbelt 2001; Alliez et al. 2002; Gu et al. 2002]) before any further processing: results of finite element computations, compression, or editing rely heavily on an good description of the original geometry. Several techniques have been proposed over the last decade, with a wide variety of target applications. In [Alliez et al. 2002], a thorough review shows that most existing methods combine mesh simplification and vertex optimization (see [Hoppe et al. 1993; Borouchaki 1998] for example); others start with a complete resampling of the surface [Turk 1992], mixed with connectivity optimization. However, even if this remeshing process has now been made both efficient and flexible, most techniques do not put any constraint on the local shape of the mesh elements: although vertex density is often required to depend on local curvatures, no condition is imposed on the resulting shape and orientation of the triangles or quads. Whenever we wish to align or stretch mesh elements with a certain direction field, we need *anisotropic remeshing*.

Such a specific remeshing is interesting for many reasons. While many elliptic partial differential equations ideally require meshes with quasi-equilateral triangles, elongated elements with large aspect ratio are often desired in the field of simulation, for fluid flow or anisotropic diffusion for instance. In these cases, a 2×2 matrix (referred to as a Riemannian metric tensor) traditionally indicates, for each point on the surface, the desired orientation and aspect ratio of the mesh element locally desired [Bossen and Heckbert 1996].

Additionally, several researchers in approximation theory have proven that the same anisotropic requirement naturally arises when an optimal mesh is sought after: for a given number of elements, a mesh will "best" approximate a smooth surface (for the L^p norms with $p \geq 1$) if the anisotropy of the mesh follows (in non-hyperbolic regions) the eigenvalues and eigenvectors of the curvature tensor of the smooth surface regions [Simpson 1994; D'Azevedo 2000]. This can be intuitively noticed by considering a canonical example, such as an infinite cylinder: planar quads infinitely stretched along the lines of minimal curvature provide the best piecewise linear description. This similarity between applications in simulation and approximation is not surprising if we interpret both these results in terms of optimal error control. In this paper, we will explore the problem of anisotropic remeshing, and present a novel, efficient, and flexible stroke-based remeshing technique whose lines continuously follow intrinsic geometric properties across a model.

1.1 Previous Work

Because of the theoretical ubiquity of anisotropic meshes, algorithms for anisotropic remeshing have been proposed in several geometry-related fields.

Anisotropic Triangle Remeshing Bossen and Heckbert [1996] proposed an anisotropic triangle meshing technique for flat, 2D regions on which a metric tensor is defined. They proceeded through successive vertex insertions, vertex removals, and iterative relaxations, that include edge flips to *align the edges* in accordance with the metric tensor. Shimada [1996] used ellipse packing to introduce anisotropy in the remeshing; although this type of methods generates high quality anisotropic meshes whose elements conform precisely to the given tensor field, this accuracy is obtained at the price of rather slow computations, and results in very limited ways for a user to guide the design of the mesh.

Heckbert and Garland [1999] made an interesting link between the quadric error metric used in their mesh simplification [Garland and Heckbert 1998] and its asymptotic behavior on finely tessellated surfaces. In particular, they demonstrated that the triangles resulting from their mesh simplification technique will be more elongated along minimal curvature directions. Such remeshing-through-simplification methods provide fast results, but again, leave very little flexibility in the process. Moreover, the anisotropic behavior is only proven for fine meshes: the results show, however, a limited (and uncontrollable) amount of anisotropy on coarse meshes. Finally, notice that work on feature remeshing [Botsch and Kobbelt 2001] has also pointed out the importance of using anisotropic triangles in feature regions and of aligning their edges to the principal directions, although no complete anisotropic remeshing technique using these principles was proposed.

Anisotropic Quad Remeshing Several works have also focused on using *quadrangles* for remeshing, due to their appealing tensor-product nature. Borouchaki and Frey [1998] described an anisotropic triangle mesh generation, and then transformed the resulting mesh into a quad-dominant mesh through a simple triangle-to-quad conversion. Shimada and Liao [1998], on the other hand, proposed to directly use rectangle packing, where the rectangles are stretched according to a specified vector field on the surface. This computational intensive packing leads to a quad-dominant anisotropic mesh, aligned with the given vector field.

In Computer Graphics, there have also been recent attempts at finding anisotropic parameterizations [Sander et al. 2002; Guskov 2002]. Gu *et al.* [2002] showed how this could be used to provide a perfectly regular remeshing of surface meshes. However, no control over the alignment of the edges with specific directions is provided.

Lines of Curvatures and Curvature-based Strokes Even if anisotropy is a relatively recent research theme in mesh processing, this particularity of almost all shapes has long been noticed and used by artists. A caricaturist, for instance, only needs a few select strokes to convey strong geometric information. Similarly, a digital artist creates or edits a 3D model in a top-down fashion, using the main axes of symmetries and a few sparse strokes to efficiently design the mesh, contrasting drastically with the local point-sampling approach of most automatic remeshing techniques (Figure 2). In the scientific community, studies and previous non-photorealistic rendering techniques have also shown how much lines of curvatures are essential in describing the geometry [Brady et al. 1985; Hertzmann and Zorin 2000]: since local directions of minimum and maximum curvatures indicate respectively the slowest and steepest variation of the surface normal, these anisotropic, intrinsic quantities govern most lighting effects. In particular, many hatching techniques use strokes that are aligned along the principal curvatures: this results in a perceptually convincing display of complex surfaces [Interrante et al. 1996; Interrante 1997; Rossl and Kobbelt 2000; Girshick et al. 2000; Hertzmann and Zorin 2000].

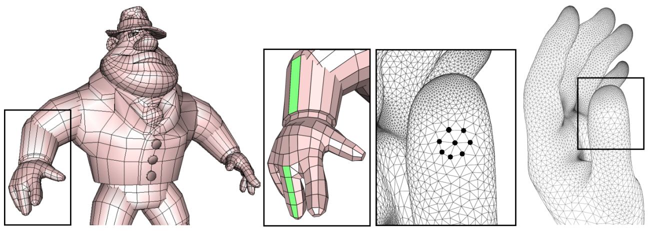


Figure 2: *Artist-designed models (left) often conform to the anisotropy of a surface, contrasting with the conventional curvature-adapted point sampling used in most remeshing engines (right).*

1.2 Contributions

Although illustration and sketching techniques have been using principal curvature strokes to represent geometry, graphics techniques rarely even exploit anisotropy of a surface to drive the remeshing process. Nevertheless, a straight edge on a coarse mesh naturally represents a zero-curvature line on the surface. It therefore seems appropriate (though non trivial!) to directly place edges parallel to the local principal directions in non-hyperbolic areas (see Figure 3, left), instead of first placing vertices to then slowly optimize their positions in order to align the induced edges.

In this paper, we propose a principal curvature stroke-based anisotropic remeshing method that is both efficient and flexible. Lines of minimum and maximum curvature are discretized into edges in regions with obvious anisotropy (Figure 3, left), while traditional point-sampling is used on isotropic regions and umbilic points where there is no favored direction (as typically done by artists; see Figure 3, right). This approach guarantees an efficient remeshing as it adapts to the natural anisotropy of a surface in order to reduce the number of necessary mesh elements. We also provide control over the mesh density, the adaptation to curvature, as well as over the amount of anisotropy desired in the final remeshed surface. Thus, our technique offers a unified framework to produce quad-dominant polygonal meshes ranging from isotropic to anisotropic, and from uniform to adapted sampling.

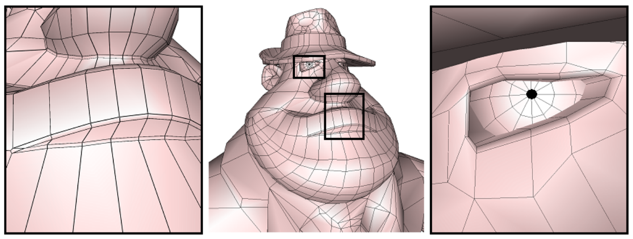


Figure 3: *Left: Skilled mesh designers tend to intuitively align edges with lines of minimum and maximum curvatures in anisotropic areas, as it provides a more compact representation of the local geometry. Right: Point sampling is, however, preferred in spherical areas where no particular direction is perceived.*

1.3 Overview

Figure 1 illustrates the main steps of our algorithm. We assume the original model to be a genus-0, non closed triangle mesh, possibly provided with tagged feature edges (non-zero genus input can be done on a per-chart basis). In a preliminary step, we build the *feature skeleton* [Botsch and Kobbelt 2001; Alliez et al. 2002], representing all the tagged features (creases and corners) in a graph of adjacency. The mesh is now ready to be remeshed:

- We first estimate the curvature tensor field of the surface at the vertices, and deduce the two **principal direction fields** stored as a 2D symmetric tensor field in a conformal parameter space. These fields are then smoothed, and the degenerate points (umbilics) are extracted (see Section 2).
- We then trace a *network of lines of curvature*, with a density guided by the local principal curvatures, in order to **sample the original geometry** appropriately along minimum and maximum curvatures, in agreement with asymptotic results from approximation theory. The isotropic regions (around the umbilic points, being either spherical or flat, are point-sampled since no obvious direction of symmetry is locally present (see Section 3).
- Finally, the vertices of the newly generated mesh are extracted from the *intersections* of lines of curvature on anisotropic areas, and a constrained Delaunay triangulation offers a convenient way to deduce the final edges from a subsampling of the lines of curvature (see Section 4). The output of our algorithm is a **quad-dominant anisotropic polygon mesh**, due to the natural orthogonality of the curvature lines.

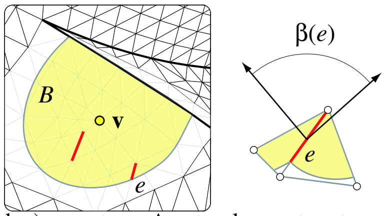
We discuss the various computational geometry and numerical tools we used to significantly ease the implementation, as well as our results in Section 5.

2 Principal Direction Fields

Since we will base our remeshing method on lines of curvature, we first need to extract the principal curvatures. In this section, we describe how the curvature tensor field of the input surface is extracted, smoothed, and analyzed. Most of these steps are performed directly in parameter space, to speed up the computations.

2.1 Robust 3D Curvature Tensor Estimation

Due to the piecewise-linear nature of the input mesh, the very notion of curvature tensor, well known in Differential Geometry [Gray 1998], becomes non trivial, and subject to various definitions [Taubin 1995; Meyer et al. 2002]. In order to have a *continuous* tensor field over the whole surface, we build a piecewise linear curvature tensor field by estimating the curvature tensor at each vertex and interpolating these values linearly across triangles. However, locally evaluating the surface curvature tensor at a vertex is not very natural. For every edge e of the mesh, on the other hand, there is an obvious minimum (i.e., along the edge) and maximum (i.e., across the edge) curvature. A natural curvature tensor can therefore be defined at each point along an edge, as noticed recently in [Cohen-Steiner and Morvan 2003]. This line density of tensors can now be integrated (*averaged*) over an arbitrary region B by summing the different contributions from B, leading to the simple expression:



$$\mathscr{T}(\mathbf{v}) = \frac{1}{|B|} \sum_{\text{edges } e} \beta(e)\ |e \cap B|\ \bar{e}\,\bar{e}^{\,t} \tag{1}$$

where $\mathbf{v}$ is an arbitrary vertex on the mesh, $|B|$ is the surface area around $\mathbf{v}$ over which the tensor is estimated, $\beta(e)$ is the signed angle between the normals to the two oriented triangles incident to edge e (positive if convex, negative if concave), $|e \cap B|$ is the length of $e \cap B$ (always between 0 and $|e|$), and $\bar{e}$ is a unit vector in the same direction as e. In our implementation, we evaluate the tensor at every vertex location $\mathbf{v}$, for a neighborhood B that approximates a geodesic disk around this vertex. This approximation is done by simply computing the disk around $\mathbf{v}$ that is within a sphere centered at $\mathbf{v}$. The sphere radius is specified by the user; a radius equal to $1/100^{th}$ of the bounding box diagonal is used by default. To remain consistent with our tensor field evaluation, the normal at each vertex can now be estimated by the eigenvector of $\mathscr{T}(\mathbf{v})$ associated with the eigenvalue of minimum magnitude. The two remaining eigenvalues κ_{min} and κ_{max} are estimates of the principal curvatures at $\mathbf{v}$. Notice that the associated directions are *switched*: the eigenvector associated with the minimum eigenvalue is the maximum curvature direction γ_{max}, and vice versa for γ_{min} (see Figure 4). This curvature tensor evaluation procedure, in addition to being intuitive and simple to implement, has solid theoretical foundations, as well as convergence properties [Cohen-Steiner and Morvan 2003].

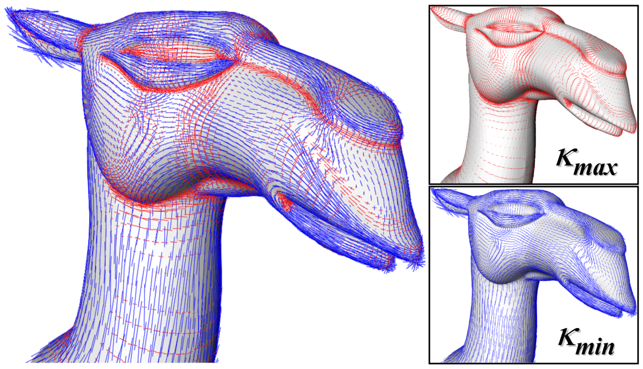


Figure 4: *Principal directions γ_{min} and γ_{max} estimated at mesh vertices, scaled by their respective curvatures.*

2.2 Flattening the Curvature Tensor Field

To allow for fast subsequent processing, we wish to 'flatten' the surface, along with its curvature tensor field. We use the discrete conformal parameterization recently presented in [Lévy et al. 2002; Desbrun et al. 2002] as the solution of choice for mapping the 3D surface to a 2D domain: based on a simple variational formulation, this parameterization automatically provides an angle-preserving mapping, without fixing any boundary positions, by simply solving a simple, sparse linear system. We also compute the induced area distortion as advocated in [Alliez et al. 2002].

On this parameterization, we can now simply store the 2D curvature tensor (the normal component is no longer needed). For every vertex in this 2D parameterization, we thus compute the 2D curvature tensor $\mathbf{T}$ such as:

$$\mathbf{T} = \mathbf{P}^t \begin{pmatrix} \kappa_{min} & 0 \\ 0 & \kappa_{max} \end{pmatrix} \mathbf{P} \tag{2}$$

We do not need to compute the matrix $\mathbf{P}$ in practice. The tensor can be found simply by picking an edge from the 1-ring, projecting it onto the tangent plane, and computing the signed angle α between this projection and the eigenvector of the maximum eigenvalue: the quasi-conformality of our parameterization allows us to now find the projected eigenvector by starting from the same edge in parameter space, and rotating it by α. The other eigenvector being orthogonal to the first one by definition, the symmetric matrix representing $\mathbf{T}$ can now be found explicitly.

Once we have $\mathbf{T}$ at each vertex, the 2D tensor field is then interpolated linearly, i.e., the matrix coefficients are linearly interpolated over each triangle (there are only three coefficients to interpolate, since the matrix is symmetric). Therefore, for any value (u,v) in the parameter space, we can return the value of the local tensor $\mathbf{T}(u,v)$.

2.3 Tensor Field Smoothing

Although the averaged nature of our tensor construction (Section 2.1) tends to remove local imperfections due to the piecewise-linear description of our input meshes, an additional pass of smoothing over the resulting 2D tensor field is often most needed. Indeed, if a coarse remeshing of the surface geometry is desired, we first have to smooth and simplify the tensor field in order to only capture the global geometry of the surface. However, if a very detailed remeshing is desired, no or little smoothing is needed.

A Gaussian filtering of the tensor (coefficient by coefficient) is performed directly in the parameter space. This is efficiently done by placing a small disk around each 2D vertex of our parameterization, with a radius inversely proportional to the local area distortion: the conformal nature of the parameterization will keep it a geodesic disk. We then convolve the field using this circular, isotropic support for the Gaussian function. Although this fast convolution is sufficient in most cases (see Figure 5), a more anisotropic smoothing of the three tensor coefficients can also be performed when higher geometric fidelity is required: the reader can refer to [Hertzmann and Zorin 2000] or [Meyer et al. 2002] for possible practical solutions. We finally get a smoothed, continuous curvature field that encodes the principal directions along with their associated curvatures as its eigenvectors and eigenvalues, respectively.

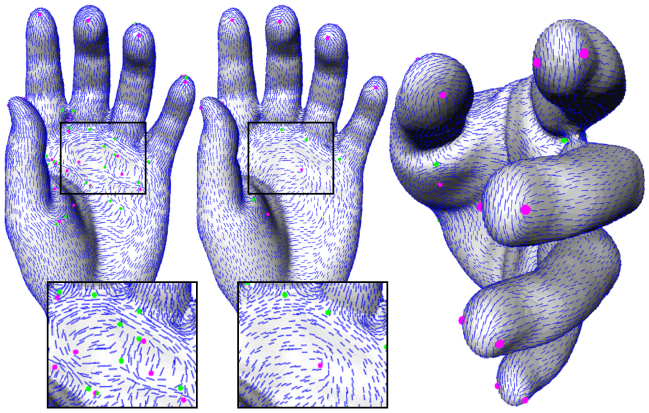


Figure 5: *Progressive smoothing of the principal direction fields. From left to the right: initial minimal curvature directions, the same region after 10 smoothing iterations, and another view of the smoothed field. Although the smoothing is computed in parameter space, the tensor field has been projected back onto the surface for illustration purposes. The color dots indicate umbilics.*

2.4 Tensor Field Umbilic Points

The topology of a tensor field is partially defined by its degenerate points, called *umbilic points*. Such degenerate points of a 2D symmetric tensor field are at locations (u_i, v_i) such as:

$$\mathbf{T}(u_i, v_i) = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}. \tag{3}$$

This corresponds to the regions of the mesh where the field is *isotropic*, *i.e.*, where the surface is locally spherical or flat. To find the umbilic points of our piecewise-linear tensor field, we follow Tricoche [2002]: we define the deviator part **D** of our tensor field **T**, obtained through:

$$\mathbf{D} = \mathbf{T} - \frac{1}{2} tr(\mathbf{T})\mathbf{I}_2 = \begin{pmatrix} \alpha & \beta \\ \beta & -\alpha \end{pmatrix}, \tag{4}$$

where the special case $\alpha = \beta = 0$ corresponds to an umbilic point. Due to the linear interpolation within each triangle, only one umbilic point can exist per triangle, and it locally corresponds to either a *wedge* type, or a *trisector* type [Tricoche 2002] as shown in Figure 6. All the umbilics can easily be found by going over each triangle and solving a 2×2 linear system. They are then classified using a third-order polynomial root-finding problem as described in [Delmarcelle and Hesselink 1994]. We keep a list of all the types and 2D positions of these umbilics for further treatment. Notice finally that the smoothing of the tensor field described in the previous section drastically reduces the number of umbilic points, as it also simplifies the topology of the extracted curvature tensor field.

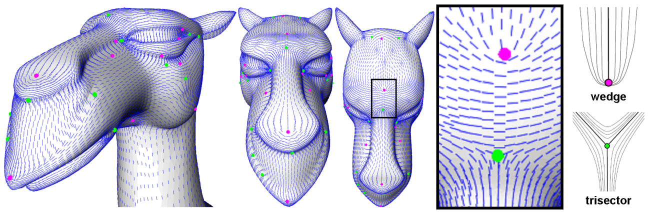


Figure 6: *Trisector and wedge umbilic points are the only possible singularities of a piecewise-linear tensor field.*

2.5 Taking Care of Features

When tagged features are present on the input mesh, special care must be used during extraction, smoothing, and umbilic analysis. First, the averaged regions over which we integrate the curvature tensors must be clipped if they intersect a feature. Indeed, feature lines often represent a significant discontinuity in the geometry (as between two adjacent faces of a cube for instance), and a one-sided evaluation is therefore recommended. Second, the smoothing step must also perform the same clipping (in the 2D plane this time) during the Gaussian smoothing of a vertex **v** near a feature also to avoid "contamination" between separate regions; after the clipping is done, the contribution due to a feature vertex located within the support is set to be the average of the values of its neighbors on the same side of the feature as **v**. These operations, simple to implement, are sufficient to deal correctly with features.

Once a smoothed tensor field is obtained, the next stage of our algorithm consists in resampling the original geometry stored as a 2D tensor field in parameter space, using both points and curvature-directed strokes.

3 Resampling

At this stage, we wish to anisotropically resample our geometry. Although a large majority of techniques perform resampling by spreading 0-elements (vertices, isotropic by nature) over the surface, this way of proceeding does not qualify as anisotropic. However, 1-elements (edges) are, by nature, anisotropic as they represent a segment of zero curvature locally. Therefore, we propose to resample the geometry by what is known as *lines of curvatures* [Gray 1998]: these lines are always along either the minimum, or the maximum curvatures. With a proper density in agreement with local curvatures, such a network of orthogonal curves will adequately discretize the object. The final edges will be found by subsampling these lines. Based on these observations, we show in this section how anisotropic areas are sampled with a set of curves aligned along principal directions, and how isotropic (*i.e.*, spherical) areas are simply discretized with points (see Figure 7).

3.1 Curve-based Sampling for Anisotropic Areas

Our goal is to trace a network of orthogonal lines of curvature in anisotropic areas. We present the numerical approach we used to successfully tracing lines, before giving details on *where* the lines are traced on the surface.

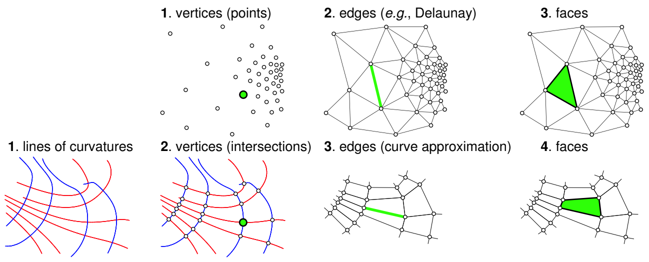

1. vertices (points)
2. edges (e.g., Delaunay)
3. faces
1. lines of curvatures
2. vertices (intersections)
3. edges (curve approximation)
4. faces

Figure 7: *Point-based sampling vs. curve-based sampling: while most techniques spread vertices first before deducing edges and faces, we use lines of curvatures on anisotropic areas to find vertex positions, before simplifying these lines to straight edges, and then deducing faces. Note that we use regular point-sampling on nearly-isotropic areas since principal directions are meaningless when the surface is almost spherical.*

3.1.1 Lines of Curvatures

By definition, a line of maximum (resp. minimum) curvature is a curve on a surface such as, at every point of the curve, the tangent vector of this curve is collinear with the principal direction of the surface that corresponds to the maximum (resp. minimum) curvature. Each line of curvature either starts from an umbilic point and ends at another one, or has a closed orbit, or can enter and exit from the domain bounds. One can trace such a curve $\mathbf{C} : t \mapsto u(t), v(t)$ in the parameter space (u, v) of the surface (see Section 2.2) by integrating the following ordinary differential equation:

$$\begin{bmatrix} u'(t) \\ v'(t) \end{bmatrix} = \gamma(t), \tag{5}$$

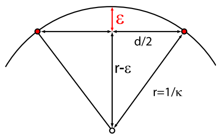
where γ is an eigenvector of $\mathbf{T}(u(t), v(t))$. More precisely, γ is the eigenvector associated with the smallest (resp. largest) eigenvalue of $\mathbf{T}$ when computing a line of maximum (resp. minimum) curvature.

3.1.2 Numerical Integration of a Line

Equation (5) can be numerically solved with an embedded fourth-order Runge-Kutta integration with adaptive step [Press et al. 1994] where the step length is weighted by the norm of the deviator (see Section 2.4), as recommended by Tricoche [2002]. If a starting point $(u^{(0)}, v^{(0)})$ is chosen, the local tensor is directly evaluated on the parameterization and its associated eigenvector γ is computed on the fly: the integration routine provides the next point along the line of curvature. By iterating this process, we find a series of locations $(u^{(k)}, v^{(k)})$ that defines a piecewise-linear approximation of a line of curvature. Notice that once the line ends (at an umbilic point, at a feature line, at the boundary, or close to another line of curvature), we start again at (u_0, v_0) but in the opposite direction this time, to complete the line. We now turn to the problem of finding the local density required for these lines of curvature.

3.1.3 Local Density of Lines

Two pivotal questions at this point of the algorithm are: how many lines should be traced on the surface, and where should we trace them? A partial answer is to first compute the desired density of lines needed at any given point on the surface, or, inversely, the spacing distance between two lines. To achieve this, first consider two lines of curvature very close to each other. A cross section of the surface, normal to these two lines, will show an approximate arc of circle (the local osculating circle of the surface) with two points on it corresponding to the trace of these two lines. A linear approximation between these two points will be away from the actual osculating circle (i.e., the surface) by a small distance. If we want to *guarantee* that this distance is less than ε in order to minimize the piecewise-linear reconstruction error, the distance d between the two points must be dependent on κ as follows:



$$d(\kappa) = 2\sqrt{\varepsilon\left(\frac{2}{|\kappa|} - \varepsilon\right)}. \tag{6}$$

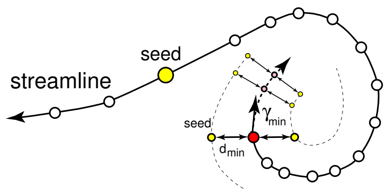
This means that for any point on a line of maximum (resp. minimum) curvature, an approximation of the optimal distance to the next line of same curvature is $d_{max} = d(\kappa_{min})$ (resp., $d_{min} = d(\kappa_{max})$). Notice that, in the limit (as element area goes to zero on a differentiable surface), Equation (6) leads to an aspect ratio of the rectangular elements equal to:

$$\frac{d_{max}}{d_{min}} \approx \sqrt{\frac{|\kappa_{max}|}{|\kappa_{min}|}}, \tag{7}$$

which coincides with the result obtained by [Simpson 1994] in approximation theory. The spacing between lines of curvature defined above thus provides, for fine meshes, optimal approximation of the underlying smooth surface. In our implementation, these theoretical distances are approximated quite well directly in parameter space: due to the conformal nature of the parameterization, multiplying such a distance by the local area stretching [Alliez et al. 2002] will provide the distance in the parameter space.

3.1.4 Curve-based Sampling

Now that we know both how to trace lines of curvature and how spaced they should be, we can start the curve-based sampling per se. High-quality placement of *streamlines* have already been studied in other applications, for visualization of vector fields for instance. Different approaches, using image guidance [Turk and Banks 1996], adapted seeding [Jobard and Lefer 1997], and more recently flow-guided seeding [Verma et al. 2000], have been proposed, but always for regularly sampled fields. It is however a trivial matter to adapt them to our context: the technique we describe next is therefore a hybrid version of [Jobard and Lefer 1997], and [Verma et al. 2000]. We will deal with the lines of minimum curvature and the lines of maximum curvature *independently*. We first put all the umbilic points into a list of *potential seeds* for lines of curvatures. We then begin by tracing lines of maximum (resp. minimum) curvature originated from the umbilic point with maximum absolute curvature, as proposed in [Verma et al. 2000]. One line gets started if the umbilic point is a wedge, while three get started if it is trisector, to respect the local topology of the vector field (see Figure 6). If no umbilics were present, we start the line at the point with the largest $|\kappa_{min}|$ (resp. $|\kappa_{max}|$). After each integration step needed to trace the line of curvature, a pair of seeds, placed orthogonally to the current line at the ideal distance (computed locally as in Section 3.1.3), is added to the list of potential seeds [Jobard and Lefer 1997].



The current line is traced until one of these cases happen:

- the line reaches another umbilic point;
- the line comes back close to its starting seed: in this case, a loop is created;
- the line crosses an edge of the feature graph or the domain boundary;
- or the line becomes too close to an existing line of maximum (resp. minimum) curvature.

The notion of closeness in the explanations above is relative to the local optimal distance d_{min} (resp., d_{max}) between lines. However, we artificially decrease the optimal distances near the umbilic points to allow for a higher-fidelity discretization. The set of potential seeds are put in a priority queue sorted by the difference between the local optimal distance at this seed and the actual distance to a streamline. The seed that best fits the local requirement is then used to start a new line, as described above. We perform this seed selection and the subsequent line tracing iteratively until a complete coverage is obtained. A final check is performed to make sure that no large areas are still uncovered. This is done by randomly sampling the parameterization space and evaluate desired distance vs. actual distances. Generally, only a handful of additional lines of curvatures get started this way.

Proximity Queries Since the algorithm described above makes heavy use of distance computations, we must handle all the proximity queries with care and efficiency. Due to the highly non-uniform distribution of samples used on the surface, a quad-tree data structure would not pay off. Instead, we opted for a conventional computational geometry tool, for which optimized implementations are readily available (such as in CGAL [Fabri et al. 2000], the library we use): a *constrained Delaunay triangulation* (CDT). Indeed, a CDT allows for fast proximity queries to constraints; furthermore, exploiting the coherence of requests (as we advance along the line of curvature) through face caching results in near-linear complexity in the number of samples. We proceed as follows: we first enter each feature segment in a CDT. Then, while we trace one line of curvature, we cache each of its samples and perform the proximity queries in the current CDT, providing distances to existing lines and features. When we are done with this line, we incorporate all its constituting segments into the CDT as constraints, and start a new line.

Control Parameters The sampling process is made flexible by providing the user with three types of control. First, the parameter ε indicating the geometric accuracy of the remeshing (see Equation 6) is an easy way to guide the number of lines of curvature. Second, the user can also apply a transfer function F (as in [Alliez et al. 2002]) to the curvatures, to tune the amount of curvature adaptation of the final mesh. Finally, the amount of isotropy vs. anisotropy is selected through a value $\rho \in [0;1]$. We turn the optimal distance definitions from Equation 6 into: $d_{max} = d(\rho/2|\ \kappa_{max}| + (1-\rho/2)\ |\kappa_{min}|)$ and $d_{min} = d(\rho/2\ |\kappa_{min}| + (1-\rho/2)\ |\kappa_{max}|)$.

3.2 Point-based Sampling in Spherical Areas

In spherical and flat areas, the surface has no special direction of symmetry; placing edges in this case does not make sense. We therefore use a more traditional point sampling technique in these regions. Although efficient [Alliez et al. 2002] or precise [Alliez et al. 2003] point-sampling methods could be used, it must be noted that these regions are extremely rare: except for canonical shapes such as a plane or a sphere, the tensor smoothing we initially perform tends to reduce the spherical regions to single umbilic point, for which sampling is straightforward.

When a region has several umbilic points, we only pick a subset of them to sample the region according to desired spacing (computed using Equation (6) again). A score for each umbilic point is computed as a function of its desired distance and the actual distance to another selected sample or to a feature line[1]. The best fit is selected, tagged as being an *isotropic sample*, and we iterate this process until we can no longer add samples. Notice that, occasionally, we use up all the umbilics without meeting the density requirement. This can only happen when large triangles in flat regions are present (since only one possible umbilic point was generated per triangle, a flat region may be undersampled). In these rare cases, we iteratively add more random samples in the triangles and proceed with the best-fit selection algorithm until saturation.

[1]This distance is computed through a proximity query to the CDT. Additionally, samples that are selected will be incorporated in the CDT in order to take them into account for future requests.

4 Meshing

The previous resampling stage has spread a series of lines of curvatures and isotropic samples over the surface. We now must deduce the final cells, edges and vertices of our remeshing process to complete our work. Principal curvatures being always orthogonal to one another, the network of lines of curvatures have created well-shaped quad regions all over the surface. We capitalize on this observation to extract a quad-dominant mesh as follows.

4.1 Vertex Creation

In anisotropic regions, we traced lines of curvature using polyline approximations while we used regular sample points for spherical and flat regions. The *vertices* will therefore be the intersections of curvature lines, and the isotropic samples that we spread. While the isotropic samples do not require any specific treatment, computing the line intersection has to be performed.

In order to perform these intersections quickly, as well as to prepare us for the next steps, we make use of a CDT again, in parameter space. We first enter all the features edges as constraints in a new CDT. We add all the little segments defining the lines of curvatures sequentially, as constraints as well. Finally, the isotropic samples are added as vertices in the CDT. The vertices, intersection of features or of the lines of curvatures, have *automatically* been added to the CDT since two intersecting edge constraints will generate a vertex insertion: the vertex creation phase is over.

Notice that the performance of this phase is, again, heavily affected by the order in which the constrained segments are added. We found, not surprisingly, that random insertion leads to slow performance. On the other hand, adding the segments sequentially along each line of curvature results in almost linear complexity, as the incremental CDT benefits from spatial coherence through caching. In our tests, the whole CDT process has been this way faster than any of the other algorithms dedicated to segment intersections we have tried without exploiting spatial coherence.

4.2 Edge Creation

The lines of curvatures must now be subsampled in order to extract the relevant edges. Although it could seem that simply joining the previously-extracted vertices would do, we must proceed with care to avoid folds on the mesh. We use a straightforward decimation process that safely removes all useless samples: going repeatedly over each vertex present in the CDT, we eliminate those which:

- are Runge-Kutta samples and have only one constraint segment attached (it will trim away all dangling curvature lines) (see Figure 8,A);
- have zero constrained segments attached and are not isotropic samples (vertices of this type appear during the decimation process, when a curvature line disappears totally for instance);
- have two constrained segments of same type attached (two minimum curvature line segments, two maximum curvature line segments, or two feature edges)—but only if removing these two segments and replacing them by a single constraint segment does *not* create any new intersections (see Figure 8,B). This last condition guarantees that our graph of region adjacencies stays planar: it will prevent *folding* in the final mesh.

This decimation is performed until we can no longer delete vertices. While this process has taken care of the anisotropic regions, we still do not have edges in isotropic regions. This is easily remedied by finally adding the CDT edges incident to the isotropic samples as constraints: it will provide a triangulation of each spherical or flat region (further edge-swaps can be performed later to reduce valence dispersion or approximation error; see [Alliez et al. 2002]).

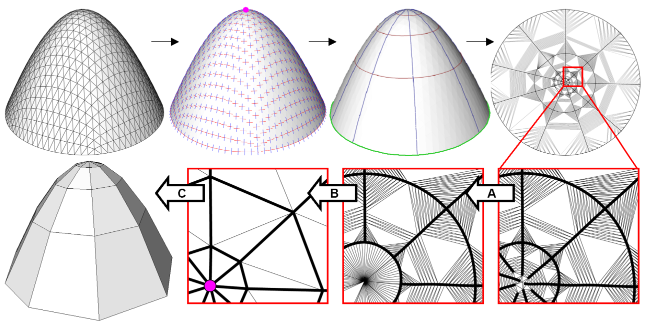


Figure 8: *Remeshing phase: a dome-like shape is sampled with lines of curvatures. All the curvature line segments (red/blue) and the feature edges (green) are added as constraints in a CDT in parameter space. The CDT creates a dense triangulation; a rapid vertex decimation (A,B) then suppresses most small edges, and leaves only few vertices, defining a coarse polygonal mesh. Adding constraint edges to the umbilic (center) point takes care of the near-spherical cap.*

4.3 Polygon Creation

The last stage of our remeshing phase extracts a final polygonal mesh from the CDT by finding all regions entirely surrounded by constrained edges: these will be our polygons. This can be done efficiently by simply visiting each CDT triangle once and recursively visit its neighbors until constraint edges are reached (see Figure 8,C). These extracted polygons being possibly concave we perform a convex decomposition using an implementation of Greene's dynamic programming algorithm [Greene 1983] (also included in CGAL). We provide an additional option to bound the highest degree of the polygons to easily allow for quad/triangle mesh generation. This task is achieved through a recursive polygon partitioning algorithm that uses simple rules for conforming-edge insertion, as indicated in Figure 9.

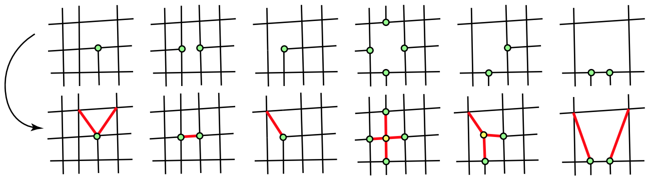


Figure 9: *A hybrid quad/triangle mesh is generated by adding conforming edges to T-junctions in a systematic manner (this table is not exhaustive).*

5 Results and Discussion

Different remeshing examples for relatively simple shapes are illustrated in Figure 10. A dome-like shape (first row) exhibits a spherical area at the top, and anisotropic areas elsewhere. The lines of maximum curvature converge towards the umbilic point at the top, and the lines of minimum curvature are concentric, closed circles. The vertices on the boundary have been deduced from intersections between feature graph and lines of curvatures. Notice how the area nearby the umbilic point has been triangulated, while other areas have been tessellated with elongated four-sided elements. For illustration purposes, a quad/triangle subdivision algorithm [Stam and Loop 2002; Levin 2003], designed to preserve the hybrid (quad/triangle) structure is applied to generate a smooth surface from the newly generated coarse mesh. Stretching the dome (second row) totally modifies the distribution of curvatures on the surface, generating rather elongated elements on highly anisotropic areas. Finally, a saddle-like shape exemplifies the various spacings happening as a function of curvatures.

The model of a pig entirely remeshed with our technique is illustrated in Figure 11. The curvature-based sampling of our lines of curvatures produces elongated quads in anisotropic areas. The edges tend to follow the local directions of symmetry, as expected. Conforming edges have been added to the output polygonal model in order to obtain a hybrid quad/triangle model. The second row shows a close-up of the ear, along with a surface obtained by quad/triangle subdivision.

Finally, three other anisotropically remeshed models are shown in Figure 12. The *octa-flower* (A) is chosen to illustrate piecewise smooth anisotropic remeshing (G,H). The direction fields are estimated, then piecewise smoothed as described in Section 2.5 (B–F). The closeup (C) illustrates how the direction fields are not influenced by the features, or by each other across the sharp creases. Remeshing the *bunny head* with three resolutions is illustrated by Figure 12(I); notice the placement of the elements on the ears. The eye and the ear of the Michelangelo's David model show the richness of the geometry: the lines of curvatures conform to all the details, creating a mesh adapted to the 'anatomy' of the original model. Note that we show the resulting polygonal mesh before insertion of conforming edges.

Timing Our current implementation allows us to process the hand model (Figure 1) in 0.4*s* for the tensor field computations, 60*s* for the sampling phase, and 1*s* for the final remeshing phase. These timings are typical of all other models, with the exception of the entire head of *Michelangelo's David* that required 8 minutes to resample. Given that no post-optimization process is required, we regard these numbers as very reasonable.

Implementation As indicated through this paper, we have tried to systematically use numerical techniques and computational geometry tools optimized and readily available to decrease the difficulty of implementation. We strongly advise *against* an implementation "from scratch" of our technique: it would result in weeks of coding, with slow and brittle results. The use of numerical techniques polished over time, and of an optimized and robust computational geometry library guarantees a much easier implementation, as well as fast and robust results. For instance, the remeshing part of our technique requires only 200 lines of code when interfaced with CGAL with an appropriate filtered kernel [Fabri et al. 2000], while earlier trials made for significant (ten times) larger code, and less robust and efficient results. For reference, the tensor field processing code requires 1000 lines, while the sampling process is 5000 lines. Notice also that being able to handle the David's head mesh is proof of numerical robustness: even very large area distortion due to flattening is accommodated for.

Limitations Due to the global parameterization used in this paper, the technique is limited to genus-0 patches. For closed or genus>0 objects, this requires to go through chart construction and surface cutting. Besides, the main bottleneck of our current approach is clearly the sampling stage. Although it is undeniably the most important stage, finding heuristics to improve it or to speed it up would be desirable. In addition, it would also be useful to develop a fast optimization phase, when higher quality bounds on the sampling density are needed. Finally, moving the remeshed vertices out of the original manifold could drastically improve the resulting error approximation, but this is not the focus of this work, and it will be explored at a later time.

6 Conclusions and Future Work

We have introduced a novel approach to remeshing, exploiting the natural anisotropy of most surfaces. Imitating artists' curvature strokes used in caricatures, we trace lines of curvatures onto the surface with a proper local curvature-dependent density before deducing a quad-dominant mesh, with elements naturally elongated along local minimum curvature directions. Resulting meshes are very efficient, in the sense that they capture the main geometric

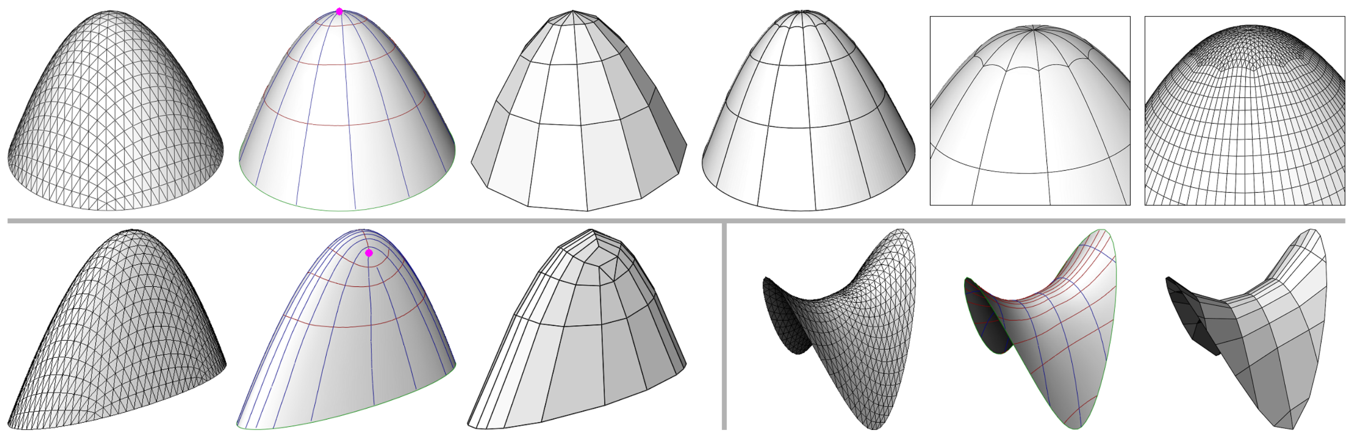


Figure 10: *Top: A dome-like shape, its lines of curvatures, the output of our remeshing process, its limit surface after quad/triangle subdivision, with two close-ups of the cap; Bottom: A squeezed dome and a saddle shape exhibit high anisotropy.*

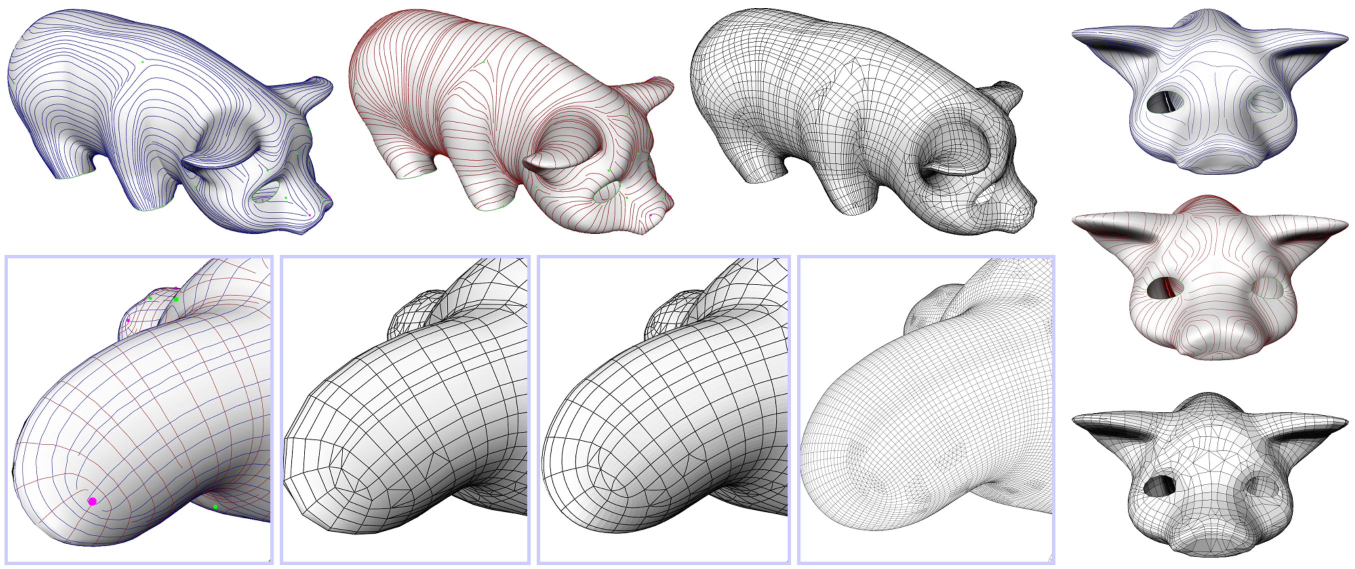


Figure 11: *Remeshing a pig. Row 1, and right column: lines of minimum (blue) and maximum (red) curvature, and the anisotropic polygon mesh generated. Row 2: close-up on an ear showing the lines of curvatures, the resulting polygon mesh with conforming edges, the surface after quad/triangle subdivision (edges of the coarse model are superimposed), and the mesh after two iterations of subdivision.*

features with a very low number of elements. This method also offers control over the mesh quality and density. Obvious extensions include a user-guided selection of the lines of curvatures.

As future work we wish to find a way to sample and remesh directly on the manifold embedded in a three-dimensional space, without using a parameterization. Finally, exploring other resampling solutions is of interest. In particular, following the direction of minimum absolute curvature would be in complete agreement with approximation theory [D'Azevedo 2000]. This approach leads to non-orthogonal edge intersections in hyperbolic regions, which is visually displeasing but optimal in terms of approximation error. We plan to investigate this alternate solution and evaluate its relevance to our community.

Acknowledgments Meshes are courtesy of the Michelangelo project, www.aranz.com, and Y. Ohtake. The authors wish to thank Xavier Tricoche for advice, Mariette Yvinec, Andreas Fabri and Lutz Kettner for their help with CGAL, Steven Schkolne, Nathan Litke and Peter Schröder for proof-reading, and Softimage for their modeler. This work was funded in part by the ARC Télégéo grant (INRIA), the ECG project of the EU No IST-2000-26473, and the NSF (CCR-0133983, DMS-0221666, DMS-0221669, EEC-9529152).

References

ALLIEZ, P., MEYER, M., AND DESBRUN, M. 2002. Interactive Geometry Remeshing. *ACM Transactions on Graphics 21(3)*, 347–354. ACM SIGGRAPH conference proceedings.

ALLIEZ, P., COLIN DE VERDIÈRE, É., DEVILLERS, O., AND ISENBURG, M. 2003. Isotropic Surface Remeshing. In *Shape Modeling International Conference Proceedings*. To appear.

BOROUCHAKI, H., AND FREY, P. 1998. Adaptive Triangular-Quadrilateral Mesh Generation. *Intl. J. Numer. Methods Eng. 41*, 915–934.

BOROUCHAKI, H. 1998. Geometric Surface Mesh. In *Int. Conf. on Integrated and Manufacturing in Mechanical Engineering*, 343–350.

BOSSEN, F., AND HECKBERT, P. 1996. A Pliant Method for Anisotropic Mesh Generation. In *5th Intl. Meshing Roundtable*, 63–76.

BOTSCH, M., AND KOBBELT, L. 2001. Resampling Feature and Blend Regions in Polygonal Meshes for Surface Anti-Aliasing. In *Eurographics proceedings*, 402–410.

BRADY, M., PONCE, J., YUILLE, A., AND ASADA, H. 1985. Describing Surfaces. *Journal of Computer Vision, Graphics, and Image Processing 32*, 1–28.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted delaunay triangulations and normal cycle. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*

D'AZEVEDO, E. F. 2000. Are Bilinear Quadrilaterals Better Than Linear Triangles? *SIAM Journal on Scientific Computing 22(1)*, 198–217.

DELMARCELLE, T., AND HESSELINK, L. 1994. The Topology of Symmetric, Second-Order Tensor Fields. In *IEEE Visualization Proceedings*, 140–145.

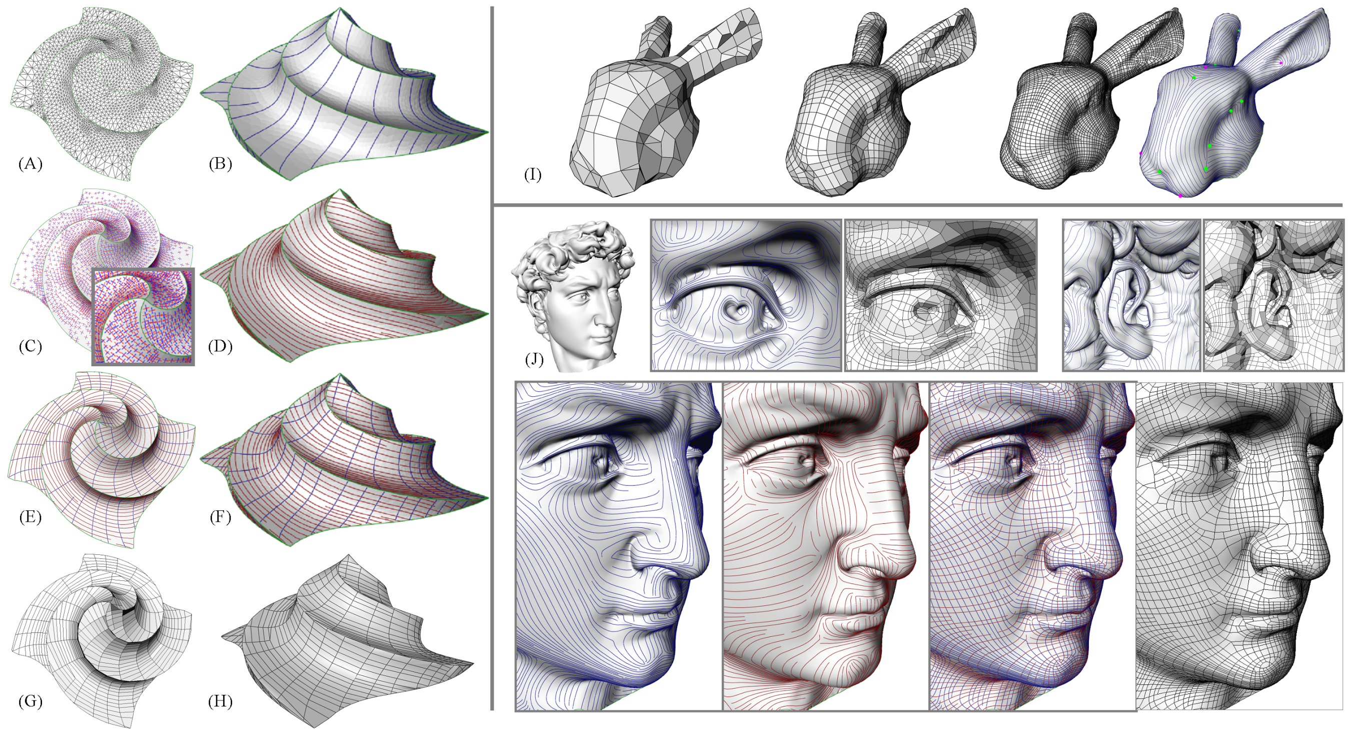


Figure 12: *A-H: the octa-flower geometry illustrates the behavior of our remeshing technique for piecewise smooth surfaces. Principal direction fields are estimated and piecewise smoothed (C) (see Section 2.5). I: The bunny's head is remeshed with different mesh densities. J: Finally, Michelangelo's David is remeshed; close-ups on the eye and the ear show the complexity of the model, and how the lines of curvatures match the local structures. Below is another closeup, on the whole face this time, with lines of curvatures and final polygonal mesh.*

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic Parameterizations of Surface Meshes. In *Proceedings of Eurographics*, 209–218.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution Analysis of Arbitrary Meshes. In *ACM SIGGRAPH Conference Proceedings*, 173–182.

FABRI, A., GIEZEMAN, G.-J., KETTNER, L., SCHIRRA, S., AND SCHÖNHERR, S. 2000. On the Design of CGAL, a Computational Geometry Algorithms Library. *Softw. – Pract. Exp. 30*, 11, 1167–1202. www.cgal.org.

GARLAND, M., AND HECKBERT, P. 1998. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. In *IEEE Visualization Conf. Proc.*, 263–269.

GIRSHICK, A., INTERRANTE, V., HAKER, S., AND LEMOINE, T. 2000. Line Direction Matters: an Argument for the use of Principal Directions in 3D Line Drawings. In *International Symposium on Non Photorealistic Animation and Rendering*.

GRAY, A., Ed. 1998. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press.

GREENE, D. H. 1983. The Decomposition of Polygons into Convex Parts. In *Computational Geometry*, F. P. Preparata, Ed., vol. 1 of *Adv. Comput. Res.* JAI Press, Greenwich, Conn., 235–259.

GU, X., GORTLER, S., AND HOPPE, H. 2002. Geometry Images. In *ACM SIGGRAPH Conference Proceedings*, 355–361.

GUSKOV, I. 2002. An Anisotropic Mesh Parameterization Scheme. In *Proceedings of 11th International Meshing Roundtable*, 325–332.

HECKBERT, P., AND GARLAND, M. 1999. Optimal Triangulation and Quadric-Based Surface Simplification. *Journal of Computational Geometry: Theory and Applications 14(1-3)* (nov), 49–65.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating Smooth Surfaces. In *ACM SIGGRAPH Conference Proceedings*, 517–526.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh Optimization. In *ACM SIGGRAPH Conference Proceedings*, 19–26.

HOPPE, H. 1996. Progressive Meshes. In *ACM SIGGRAPH Conference Proceedings*, 99–108.

INTERRANTE, V., FUCHS, H., AND PIZER, S. 1996. Illustrating Transparent Surfaces with Curvature-directed Strokes. In *IEEE Visualization*.

INTERRANTE, V. 1997. llustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution. In *ACM SIGGRAPH Conference Proceedings*, 109–116.

JOBARD, B., AND LEFER, W. 1997. Creating Evenly-Spaced Streamlines of Arbitrary Density. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*, 45–55.

KOBBELT, L., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. 1999. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum, Eurographics '99 issue 18*, 119–130.

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *ACM SIGGRAPH Conference Proceedings*, 95–104.

LEVIN, A. 2003. Polynomial generation and quasi-interpolation in stationary non-uniform subdivision. *Computer Aided Geometric Design 20(1)*, 41–60.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. In *ACM SIGGRAPH Conference Proceedings*, 362–371.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H., 2002. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. Proceedings of VisMath.

PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. 1994. *Numerical recipes in C – The art of scientific programming*, 2nd ed. Cambridge University Press, UK.

ROSSL, C., AND KOBBELT, L. 2000. Line-art Rendering of 3D Models. In *Proceedings of Pacific Graphics*.

SANDER, P., GORTLER, S., SNYDER, J., AND HOPPE, H. 2002. Signal-specialized parametrization. In *Eurographics Workshop on Rendering 2002*.

SHIMADA, K., AND LIAO, J. 1998. Quadrilateral Meshing with Directionality Control through the Packing of Square Cells. In *7th Intl. Meshing Roundtable*, 61–76.

SHIMADA, K. 1996. Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles. In *6th Intl. Meshing Roundtable*, 63–74.

SIMPSON, R. B. 1994. Anisotropic Mesh Transformations and Optimal Error Control. *Appl. Num. Math. 14(1-3)*, 183–198.

STAM, J., AND LOOP, C., 2002. Quad/triangle subdivision. Preprint.

TAUBIN, G. 1995. Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation. In *Proceedings of Fifth International Conference on Computer Vision*, 902–907.

TRICOCHE, X. 2002. *Vector and Tensor Field Topology Simplification, Tracking, and Visualization*. PhD thesis, Universität Kaiserslautern.

TURK, G., AND BANKS, D. 1996. Image-Guided Streamline Placement. In *ACM SIGGRAPH Conference Proceedings*, 453–460.

TURK, G. 1992. Re-Tiling Polygonal Surfaces. In *ACM SIGGRAPH Conference Proceedings*, 55–64.

VERMA, V., KAO, D. T., AND PANG, A. 2000. A Flow-guided Streamline Seeding Strategy. In *IEEE Visualization*, 163–170.