Discrete Fairing and Variational Subdivision for Freeform Surface Design

Leif P. Kobbelt[*]

Max-Planck-Institute for Computer Sciences

Abstract

The representation of freeform surfaces by sufficiently refined polygonal meshes has become common in many geometric modeling applications where complicated objects have to be handled. While working with triangle meshes is flexible and efficient, there are difficulties arising prominently from the lack of infinitesimal smoothness and the prohibitive complexity of highly detailed 3D-models. In this paper we discuss the generation of fair triangle meshes which are optimal with respect to some discretized curvature energy functional. The key issues are the proper definition of discrete curvature, the smoothing of high resolution meshes by filter operators, and the efficient generation of optimal meshes by solving a sparse linear system that characterizes the global minimum of an energy functional. Results and techniques from differential geometry, variational surface design (fairing), and numerical analysis are combined to find efficient and robust algorithms that generate smooth meshes of arbitrary topology which interpolate or approximate a given set of data points.

1 Introduction

There are several standard representations for surface geometry each of which is appropriate for specific tasks and operations. The most widespread representation in geometric modeling applications are *parameteric surfaces* $F : \Omega \in \mathbb{R}^2 \to \mathcal{S} \in \mathbb{R}^3$ which enable efficient point sampling by evaluating the function F at arbitrary locations $(u,v) \in \Omega$. *Spline representations* are particularly useful due to their intuitive shape control mechanism based on control vertices which locally attract the surface and hence enable the designer to generate and modify geometric models by roughly sketching the intended surface [18, 8].

Point location and ray intersection computations are more easily performed if a surface is given as a level set

$$\mathcal{S}_c \;=\; \left\{[x,y,z] \in \mathbb{R}^3 \,|\, f(x,y,z) = c\right\}$$

of some spatial scalar field $f : \mathbb{R}^3 \to \mathbb{R}$ (*implicit surfaces*). However in this representation, shape control and evaluation is rather difficult since slightly changing the function f can have drastic effects on the shape of $\mathcal{S}_c$ — including changes in the topology.

In engineering applications it is often desirable to represent surfaces as the outer skin of a solid object. Such objects are typically constructed by combinations of simple basic shapes like spheres, cones and boxes (*Constructive Solid Geometry*). However, the lacking shape flexibility of CSG objects makes this technique inappropriate for sophisticated freeform modeling applications.

Fig. 1 shows typical examples for the different surface representations. What is common to all of them is the fact that — at some stage of their processing — they are all converted into a triangle mesh representation: the samples on parameteric surfaces are connected by edges and triangles [19], implicit surfaces are extracted by the marching cubes algorithm [27], and CSG objects are handled by uniformly tesselating the basic objects up to a prescribed resolution.

The reason why this *discretization* is done, is the efficient way how triangle meshes can be displayed by today's computer graphics hardware. Moreover for the sake of efficiency and robustness many algorithms like milling path generation and FE simulation are preferrably performed on polygonal meshes. Hence, polygonal meshes or *explicit surfaces*[1] can be identified as the most versatile general purpose surface representation. Meshes provide maximum flexibility since arbitrarily complex objects can be constructed by simply putting together the triangles without having to observe complicated mathematical continuity conditions.

The two major difficulties with discrete and explicit surface representations are the lacking infinitesimal smoothness and the high complexity. Polygonal models with several millions of triangles have become commonplace since moderately priced 3D scanning devices are available. In order to be able to handle the complexity of such meshes on standard PCs and workstations we have to apply mesh decimation algorithms which reduce the number of triangles in a given model while minimally changing its geometric shape [4, 11, 17, 23, 33, 35]. As a byproduct such algorithms also generate *hierarchical representations* for highly detailed meshes and thus enable to dynamically adapt the *level of detail* to the available hardware resources and the application-dependent quality requirements.

In this paper we are concerned with the other central problem, namely the *smoothness* of triangle meshes. Although triangle meshes can never be smooth in the narrower sense of C^1 continuity, there is an intuitive notion of approximate smoothness or *discrete smoothness*. Fig. 2 shows two meshes which are both C^0 but the right one obviously is smoother than the left one.

In the next section we will present a more precise definition of *discrete curvature*. Smooth meshes are then characterized by low discrete curvature. The operators by which curvatures can be computed on triangle meshes lead to simple filter algorithms that improve the smoothness of an existing mesh by moving the vertices in order to minimize the discrete curvature or its variation. Applying

[*]Max-Planck-Institute for Computer Sciences, Im Stadtwald, 66123 Saarbrücken, Germany, `kobbelt@mpi-sb.mpg.de`

[1]We use the term *explicit surfaces* since all vertices of a triangle mesh have to be enumerated explicitly and there is in general no global rule how to compute their position otherwise.

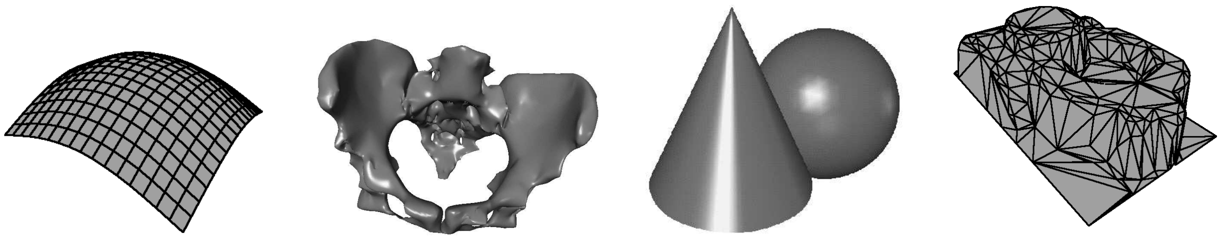

Figure 1: Different representations for surface geometry. From left to right: Parameteric surfaces map a planer domain into space (patches), implicit surfaces are iso-surfaces of scalar valued volume data and CSG objects are defined by joining or intersecting basic shapes. Finally triangle meshes provide maximum flexibility and maximum efficiency. In fact, all models shown here are actually represented by triangle meshes since they have been converted for display.

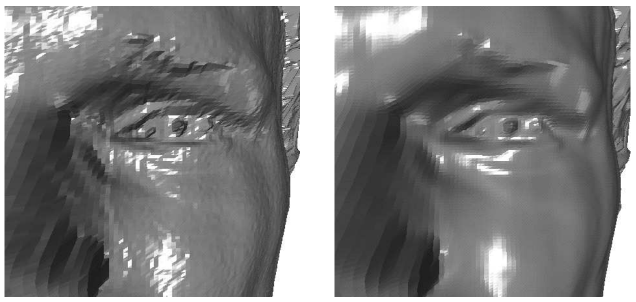

Figure 2: A mesh model obtained by scanning a human head. On the left the raw data is shown where noise artifacts are clearly visible. On the right a smoother version of the same model is shown. Although both models are C^0 surfaces, the right one apparently has an improved distribution of (discrete) curvature.

such *discrete fairing* algorithms on different levels of a hierarchical mesh representation significantly accelerates their convergence.

By combining variational methods from CAGD with the classical subdivision paradigm where surfaces are computed by iteratively generating a sequence of finer and finer meshes, we define the so-called *variational subdivision schemes* which enable the construction of high quality meshes with minimum discrete curvature and arbitrary topology. Finally we discuss how to integrate boundary conditions into the optimization problem and enumerate some application areas.

2 Discrete smoothness

Triangle meshes are piecewise linear surfaces and hence cannot be infinitesimally smooth unless they are planar. However, if the vertices of a triangle mesh are samples from a smooth continuous surface then we can approximate that surface increaslingly tight by refining the mesh. The sequence of meshes generated during this process converges to the smooth limit surface and hence it is called *asymptotically smooth*. This definition is used to rate the smoothness of stationary subdivision schemes and the expected quality of meshes generated by their iterative application [7, 32, 41].

However, in most applications we are not dealing with infinitely refined meshes and hence we have to derive a discrete approximation of the curvature which mimics the properties of the corresponding definitions in differential geometry. Namely, we are interested in finding discrete versions of the first and second fundamental form because this enables us to transfer the known concepts from the continuous to the discrete setting.

In numerical analysis, discrete approximations of *derivatives* are usually obtained by applying *divided difference operators*. If the values of a function are only known at discrete sample points then derivatives are estimated by computing a low-degree polynomial interpolant to the data points and evaluating its derivatives. The idea behind this construction is that the polynomial interpolant certainly is a good approximation to any smooth surface passing through the data points. Hence the leading coefficients of their local Taylor expansions won't differ very much.

In geometric modeling we are not dealing with functions but more generally with surfaces. A triangle mesh typically comes without canonical parameterization and hence in order to derive divided difference operators we first have to find parameter values for the mesh vertices. In general, this is a hard problem if certain quality requirements for the parameterization have to be satisfied [28]. In our case, however, the problem is not as difficult since we only need *local* parameterizations for the construction of divided difference operators.

Assume we want to compute *second* order partial derivatives then we have to compute a *quadratic* local interpolant. The interpolation problem is well-defined if we have six independent interpolation conditions. This means that in order to compute the second order derivatives at some vertex $\mathbf{p}_0$ in the mesh, we have to assign parameter values to it and to at least five other vertices in the vicinity. For symmetry reasons we usually take all neighboring vertices which are directly connected to $\mathbf{p}_0$. If the valence of $\mathbf{p}_0$ is higher than five, we compute an optimal quadratic approximant in the least squares sense. If the valence is below five, we can either compute a least norm solution or we can use a larger neighborhood.

Let $\mathbf{p}_1,\ldots,\mathbf{p}_n$ be the neighbors of $\mathbf{p}_0$ and we assign the parameter values (u_i,v_i) to $\mathbf{p}_i$ (cf. Fig. 3). Without loss of generality we assume $(u_0,v_0)=(0,0)$. The linear Vandermonde system for the (least squares) interpolating polynomial

$$F(u,v) \;=\; F + uF_u + vF_v + \frac{1}{2}u^2F_{uu} + uvF_{uv} + \frac{1}{2}v^2F_{vv}$$

is given by

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ u_i & v_i & \frac{1}{2}u_i^2 & u_iv_i & \frac{1}{2}v_i^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} F_u \\ F_v \\ F_{uu} \\ F_{uv} \\ F_{vv} \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{p}_i - \mathbf{p}_0 \\ \vdots \end{pmatrix}. \tag{1}$$

where we exploit that $F=\mathbf{p}_0$. If we denote the matrix in (1) by $\mathbf{V}$

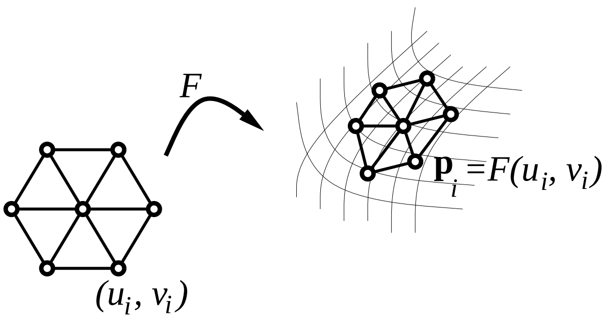

Figure 3: For the construction of the local divided difference operators at a specific vertex $\mathbf{p}_0$, we have to find parameter values (u_i, v_i) for the direct neighbors $\mathbf{p}_i$ in order to solve the local interpolation problem.

then the least squares solution is given by

$$\begin{pmatrix} F_u \\ F_v \\ F_{uu} \\ F_{uv} \\ F_{vv} \end{pmatrix} = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T \begin{pmatrix} \vdots \\ \mathbf{p}_i - \mathbf{p}_0 \\ \vdots \end{pmatrix}.$$

The rows of the matrix $\mathbf{D} = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T$ are the coefficients of the divided difference operators.

Obviously there are many degrees of freedom in the choice for the parameter values (u_i, v_i) and the remaining question is whether there are specific choices which are optimal in some sense.

The most simple way to define the values (u_i, v_i) is by uniform parameterization

$$(u_i, v_i) = \Big(\cos(2\pi i/n),\ \sin(2\pi i/n)\Big).$$

The advantage of this choice is that the matrix $\mathbf{D}$ depends only on the valence n of $\mathbf{p}_0$. Hence, the coefficients for the divided difference operators can be computed in advance for each possible valence $n = 3, 4, \ldots$

However from scattered data interpolation techniques we know that the uniform parameterization often leads to rather bad interpolating polynomials. The approximation of the data can be improved by adapting the parameterization to the spatial configuration of the vertices $\mathbf{p}_i$. Moreover, when rating the smoothness or fairness of a triangle mesh we are not so much interested in partial derivatives with respect to some arbitrary parameterization but our eventual goal is to find approximations of the principal curvatures κ_1 and κ_2 or their combinations $T = \kappa_1^2 + \kappa_2^2$ (total curvature), $H = \kappa_1 + \kappa_2$ (mean curvature), or $K = \kappa_1\,\kappa_2$ (Gaussian curvature).

It is a well-known theorem in differential geometry that T, H, and K are simple (linear or quadratic) combinations of the second order partial derivatives if the parameterization is *isometric*. This means that both tangent vectors F_u and F_v have unit length and are perpendicular to each other. In general there is no isometric global parameterization for surfaces unless $K \equiv 0$ everywhere.

For the derivation of the divided difference operators, however, we need a consistent parameterization only in a small neighborhood of each vertex. We can choose these local parameterizations independent from each other because we construct a custom tailored set of divided difference operators for each vertex. This means that we can assign different parameter values to the same vertex if local parameterizations are overlapping. Since the Taylor coefficients F_{uu}, F_{uv}, and F_{vv} are evaluated at the center point $F(0,0) = \mathbf{p}_0$ we tune each local parameterization such that it becomes (approximately) isometric at $(u, v) = (0, 0)$.

There are several heuristics to find such a parameterization. The most simple one is to estimate a normal vector $\mathbf{n}_0$ at $\mathbf{p}_0$ and then project the neighboring vertices $\mathbf{p}_i$ into the corresponding tangent plane (cf. Fig. 4). The projected points $\mathbf{p}_i'$ are represented with respect to an orthonormal basis at the origin $\mathbf{p}_0' = \mathbf{p}_0$. Usually this basis is found by normalizing $\mathbf{u}_0 = \mathbf{p}_1' - \mathbf{p}_0$ and $\mathbf{v}_0 = \mathbf{u}_0 \times \mathbf{n}_0$.

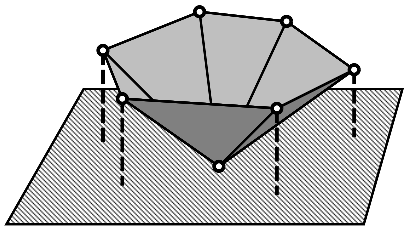

Figure 4: Projecting the adjacent neighbors $\mathbf{p}_i$ of a vertex $\mathbf{p}_0$ into an estimated tangent plane yields parameter values (u_i, v_i) for a local parameterization which is isometric at $(u, v) = (0, 0)$, i.e., the (discrete) first fundamental form is the identity.

Another heuristic which does not even require to estimate a normal vector is the *discrete exponential map* which emerges from a local parameterization in polar coordinates [40]. The parameter values are defined as

$$(u_i, v_i) = h_i\Big(\cos(\sum_{j=1}^{i-1}\alpha_j),\ \sin(\sum_{j=1}^{i-1}\alpha_j)\Big)$$

where h_i is proportional to the edge length $\|\mathbf{p}_i - \mathbf{p}_0\|$ (geodesic distance) and the angle α_j is proportional to the angle $\angle(\mathbf{p}_j, \mathbf{p}_0, \mathbf{p}_{j+1})$ such that

$$\sum_{j=1}^{n}\alpha_j = 2\pi.$$

All these heuristics uniquely define the parameter values (u_i, v_i) up to rotations around the origin in the parameter plane. However it turns out that although the second order partial derivatives (and hence the second fundamental form) depends on the orientation in the parameter plane, the curvature values H, K, and T do not (rotation invariance) [1]. Consequently, we can arbitrarily choose the orientation, e.g., by forcing (u_1, v_1) to lie on the u-axis.

One immediate application of the discrete curvature estimation on meshes is the detection and visualization of surface artifacts. When real objects are scanned or mechanical deformation processes are simulated numerically, the resulting surface geometry is often given as a mere polygonal mesh with no additional shape information. *Reverse engineering* is necessary to obtain a continuous representation on which curvature analysis can be done.

With the divided difference operators available, we can plot the color coded discrete curvature directly on the given mesh and hence the *surface interrogation* can be performed without reconstructing a full-size CAD model.

Fig. 5 shows a typical mesh for finite element analysis. The color coded mean curvature H is plotted directly on the mesh such that cylindrical and flat regions are clearly visible due to constant color.

3 Discrete Fairing

Once we know how to compute curvatures on triangle meshes, the next step is to use this information for optimizing the fairness of a

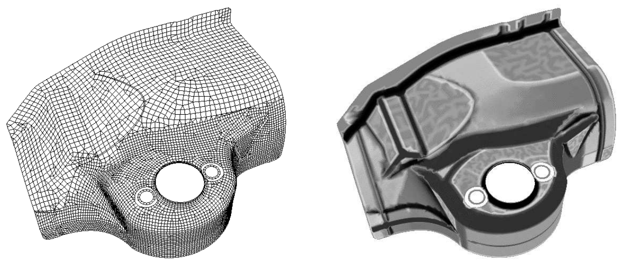

Figure 5: Precise curvature analysis on polygonal mesh models can be performed by using locally adapted divided difference operators. This enables direct surface interrogation on discrete data without reverse engineering of CAD models.

given mesh by reducing its curvature or the variation of its curvature.

A standard measure for the global surface quality in geometric modeling is the thin plate energy [30, 34]

$$E(\mathcal{S}) \;=\; \int_{\mathcal{S}} \kappa_1^2 + \kappa_2^2. \tag{2}$$

This geometric functional is rather difficult to handle and hence it is usually approximated by its ”linearized” version

$$E(\mathcal{S}) \;\approx\; \int_{\Omega} F_{uu}^2 + 2F_{uv}^2 + F_{vv}^2 \tag{3}$$

where the quality of the total curvature's approximation by the squared second partial derivatives depends on how strongly the actual parameterization $F(u,v)$ deviates from an isometric parameterization. The effect of this approximation error on the resulting minimum energy surface is very difficult to estimate. If we are using polynomial splines, then the functional (3) can obviously be evaluated exactly. Even for subdivision surfaces which emerge from generalized B-spline subdivision, the value of (3) can be computed from the given control mesh [16, 29].

In the discrete setting, the integral can be approximated by a weighted sum over all vertices (*quadrature formula*) and the partial derivatives are approximated by divided differences:

$$E(\mathcal{S}) \;=\; \sum_i \omega_i \left(\Big(\sum_j \alpha_{i,j}\,\mathbf{p}_{i,j}\Big)^2 + 2\Big(\sum_j \beta_{i,j}\,\mathbf{p}_{i,j}\Big)^2 + \Big(\sum_j \gamma_{i,j}\,\mathbf{p}_{i,j}\Big)^2 \right). \tag{4}$$

Here $\mathbf{p}_{i,j}$ are the neighbors of the vertex $\mathbf{p}_i$ and $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ are the coefficients of the second order divided differences at $\mathbf{p}_i$ which are obtained by solving the local interpolation problem. The weight coefficients ω_i represent the discrete surface element, i.e. their sizes are proportional to the surface area that is associated with the corresponding vertex $\mathbf{p}_i$. Hence, a good choice for ω_i is the sum of the adjacent triangles' areas (constant factors do not matter).

The important advantage of the discrete functional (4) is that the local divided difference operators can be constructed with respect to locally isometric parameterizations while the continuous functional (3) requires a global (or at least patch-wide) parameterization. Hence in the discrete setting, the original functional (2) is properly sampled and the approximation (of the functional) can be improved by reducing the sample step width, i.e. by refining the mesh. In the continuous setting the approximation error between (2) and (3) cannot be reduced by refining the spline space since this does not change the planarity of the parameter domain Ω which is the source for the distorsion of the parameterization (= non-isometry) [12]. Fig. 6 demonstrates the effects resulting from the difference between both types of approximation.

The functional $E(\mathcal{S})$ is quadratic in every vertex and hence its partial derivatives are linear expressions. The unique minimum of $E(\mathcal{S})$ is found if all partial derivatives with respect to the vertices vanish. We find

$$\frac{\partial E(\mathcal{S})}{\partial \mathbf{p}_i} \;=\; \sum_k w_{i,k}\,\mathbf{p}_k \;\overset{!}{=}\; 0 \tag{5}$$

where the sum in fact ranges over all direct neighbors of $\mathbf{p}_i$ and all *their* neighbors (2-ring neighborhood) because all other $w_{i,k}$ vanish.

Equation (5) is one row of a large linear system whose solution is the minimum energy mesh. The number of rows in this linear system equals the number of vertices which can be moved freely in order to reduce the global energy. The system is sparse since in every row only the entries corresponding to vertices in the 2-ring neighborhood are non-vanishing.

The easiest way to solve such a system is by an iterative alogrithm like the Gauß-Seidel scheme. We solve each row for the diagonal element

$$\mathbf{p}_i \;=\; -\frac{1}{w_{i,i}} \sum_{k \neq i} w_{i,k}\,\mathbf{p}_k \tag{6}$$

which yields a local update rule. By repeatedly cycling through the vertices and applying the update rule, the discrete thin plate energy (4) is iteratively reduced.

As we will discuss in Section 6 we additionally have to define proper boundary conditions for the optimization problem since otherwise the mesh will collapse eventually.

Ignoring this fact, we can still apply the Gauß-Seidel update rule like a linear filter operation in order to remove high frequency noise from the mesh data. The reason why this works is because the special spectral properties of the iteration matrix imply that convergence is fast in high frequency subspaces and rather slow in low frequency subspaces. Hence the noise which affects every other vertex in the mesh is removed after a few iterations while larger features are more or less preserved. Intuitively, this behaviour can be explained by the observation that the local update rule effectively reduces the curvature by considering only a small portion of the mesh while such changings propagate very slowly to more distant mesh regions.

Notice that although the global convergence of the iterative scheme is slow, we still observe slight changings in the global shape: the application of the update rule as a mere filter operation causes a global shrinking effect [36]. This cannot happen if proper boundary conditions are imposed.

In the above heuristics, we used the geometric constellation of a vertex $\mathbf{p}_i$ and its neighbors $\mathbf{p}_{i,j}$ to derive a locally isometric parameterization. Since this constellation changes during the iterative updating it seems reasonable to also update the local parameterization. Assuming that the current mesh is always the best known approximation to the optimal mesh, it might make sense to estimate the local surface metric based on this information.

However, besides the high computational costs for repeatedly recomputing the coefficients of the divided difference operator, the major argument against this strategy is the instability of the resulting scheme. If we change the parameterization after every iteration we in fact also change the underlying energy functional that is to be minimized. Hence there is no reason to assume that the iteration will ever converge.

Since each asymmetry in the neighborhood of a vertex $\mathbf{p}_i$ is balanced by adapting the local parameterization, the energy func-

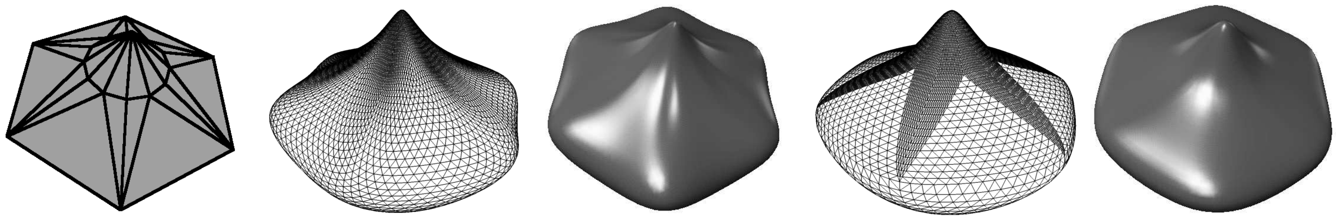
Figure 6: The triangle mesh on the left is to be interpolated by a surface which minimizes the thin plate energy. In the center, the geometric functional is approximated by the continuous quadratic functional with respect to some uniform parameterization which does ignore the unequal distribution of interpolation points. Consequently the resulting surface has a rather uniform distribution of mesh vertices (center left). This surface (center) is optimal but with respect to the *wrong* functional! For the surface on the right a parameterization has been used which adapts to the given surface's metric. Hence divided differences are computed with respect to locally isometric parameterizations and the true functional is approximated much better. The vertex distribution on the mesh is similar to the distribution in the original mesh (center right).

tional's gradient implies no force component acting *within* the tangent plane. Consequently, the resulting non-linear system becomes unstable because vertices can move freely on the surface. For example a sphere is a surface with minimum curvature variation. If we have a mesh with vertices lying on a sphere then we obtain the same curvature estimate for each vertex (under the assumption that the local divided difference operators are properly adjusted). Allowing the parameterization to be updated during the iteration leads to a situation where the vertices can flow freely on the sphere. Some authors tried to cope with this problem by restructuring the mesh, i.e. edges are flipped when the local triangle distorsion violates some upper bound [40], but this is not appropriate in many applications.

Hence, in order to stabilize the optimization problem, we have to apply the iterative mesh filtering with divided difference operators being constructed once for a fixed parameterization. The difficulties with this fixed parameterization emerge from the fact that we have to estimate the surface metric of the result a priori. In practice, it turns out that the heuristics of Sect. 2 lead to satisfactory results. In Sect. 5 we will come back to this issue in a slightly different context.

4 Multi-level smoothing

Although the update rule (6) works like an effective filter operation to remove high frequency noise from a given mesh, its global convergence is rather slow. A standard technique in numerical analysis to deal with such situations are multi-level approaches where the same optimization problem is solved on different levels of detail in order to accelerate the global convergence [14]. The idea behind this approach is that iterations on coarse levels are cheap and hence a feasible solution can be obtained. This coarse-scale solution is *prolongated* in order to find a good starting configuration for the iterative solver on the next finer level. The prolongation operator is designed such that the error on the finer level is mostly high frequency and hence can be reduced effectively by the update rule. We can mimic this behaviour in our mesh filtering setting by applying the Gauß-Seidel update rule (6) to a hierarchical representation of the mesh.

For an arbitrary mesh, a hierarchical structure emerges from the application of a mesh decimation algorithm. In [24] a sequence of nested unstructured meshes is generated by successively removing vertices from a given mesh by edge collapses (*fine-to-coarse hierarchy*). Starting on a coarse scale, we can reconstruct finer resolutions (and eventually the original mesh) by applying the inverse operations (vertex splits) in reverse order (*progressive meshes*) [17].

There are different ways to define distinct levels of resolution within the continuous spectrum of resolutions provided by a progressive mesh representation. We combine sub-sequences of vertex split operations or edge collapses into (macro-) upsample and downsample operations. Criteria for the distinct levels can be the exponential growth in complexity by a certain factor, the removal or insertion of an independent set of vertices, or a specific average edge length on each level of detail. Here, a set of vertices is called independent if the neighborhoods which are affected by the corresponding collapse operations are disjoint.

A complete V-cycle multi-level algorithm consists of the following operations: the *restriction operator* maps the original data to a coarser level, the *iterative solver* approximates the solution by running several Gauß-Seidel update cycles on the coarse mesh, the *prolongation operator* reconstructs the original fine mesh topology, and some more update iterations on the fine mesh yield the final result. Notice that the restriction and prolongation can run over several refinement levels.

When using this technique in the context of mesh filtering, we can exploit the typical convergence behaviour of the Gauß-Seidel update for the design of sophisticated geometric low-pass filters. The schedule of these filters is similar to the V-cycle algorithm. We first apply mesh decimation to the given mesh until all the detail that is to be filtered is removed. On this level of detail we start alternating the re-insertion of vertices by vertex splits and the iterative mesh smoothing by Gauß-Seidel updates. Since on each level, the highest frequency noise is removed, we eventually end up with a mesh that has the same connectivity as the original one but with the geometric detail removed (cf. Fig. 7). In [13] this generalized notion of signal processing for triangle meshes is investigated in more detail.

5 Variational subdivision

In the last section, we discussed the situation when a fine triangle mesh is given and discrete fairing techniques are applied to improve its quality. A rather different setup in freeform modeling is the scattered data interpolation problem where only few points in space are given and a smooth interpolating surface is sought. The topology of the interpolating surface is usually part of the input data. Hence the given data points usually come as the vertices of a coarse triangle mesh and we want to generate a refined mesh with the same topology which interpolates or approximates the original points.

5.1 Subdivision operators

The most effective technique in the context of geometric modeling with polygonal meshes are *subdivision schemes*. A subdivi-

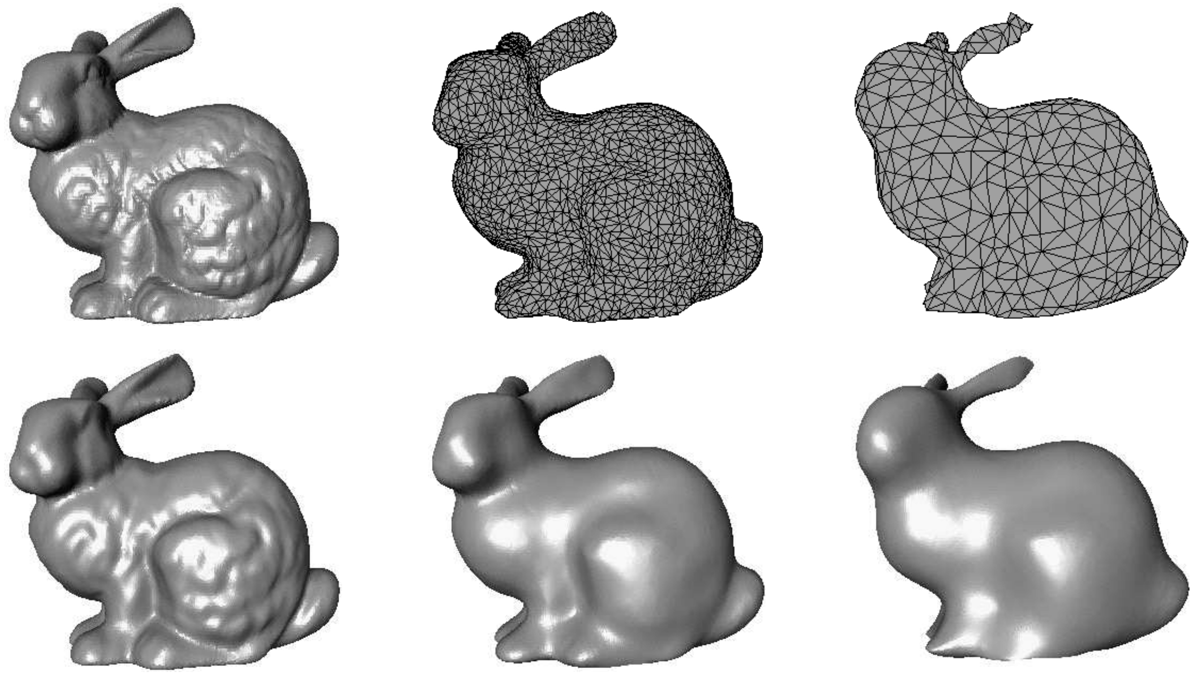

Figure 7: The effect of the V-cycle multi-level smoothing. On the left top the original bunny data set is shown. If we apply the smoothing update operator, we obtain the result on the left bottom. Only the highest frequency noise is removed since the convergence on lower frequency bands is too slow. We can apply mesh decimation (restriction) to switch to coarser levels of detail (top row, left-to-right). Alternating mesh smoothing and re-insertion of the vertices yields the surfaces in the bottom row. All meshes in the bottom row have the same connectivity. Notice how the degree by which detail information is removed, corresponds to the coarseness of the base mesh in the V-cycle scheme (top row).

sion scheme is given by a set of rules to refine a polygonal control mesh. By iteratively applying these rules, we generate a sequence of meshes which eventually converge to a smooth limit surface (cf. Fig. 8). This technique is very similar to the knot-insertion operation for spline surfaces [25]. Inserting a knot-line into the spline representation of a surface requires the computation of new control vertices according to simple rules (linear combinations of old control vertices). It is well-known that the control meshes converge to the spline surface if the knot-lines eventually become dense [31].

Subdivision schemes are a generalization of knot-insertion in the sense that the refinement rules of a subdivision scheme can be applied to arbitrary meshes, i.e., the tensor-product regularity of spline control meshes is not required. This enables the generation of surfaces with arbitrary topology and not only triangular or quadrilateral patches.

The subdivision operator always consists of two parts. The first is a topological split operation by which new control vertices are inserted into the mesh. The split operation is chosen *uniform* such that all new vertices are regular and the number of extraordinary vertices in the refined meshes is constant (*subdivision connectivity*). The second part is a smoothing operator which moves the control vertices according to weighted averages of neighboring vertices. The weights of the smoothing rule usually depend on the valence of the vertices only. In Fig. 9 a typical sequence of meshes generated by this process is shown. This type of subdivision operator is called *stationary* since the same rule is applied in each step of the iterative refinement.

There are two major classes of subdivision schemes. One class are the *interpolatory* schemes which do not change the position of the old vertices (and hence the refined mesh always interpolates the coarser one). For interpolatory schemes, the limit surface interpolates all intermediate control vertices. The other class are the *non-interpolatory* schemes where all vertices are shifted by the smoothing operator. In this case the limit surface only approximates the control vertices (cf. cubic spline curves and their control polygon).

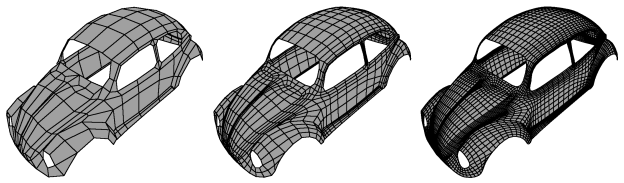

Figure 8: Subdivision schemes can be applied to arbitrary meshes. The advantages of spline surfaces (built-in smoothness and intuitive local shape control by control vertices) are preserved with the additional flexibility that smooth surfaces with arbitrary topology can be generated.

For all important settings, i.e., interpolatory / non-interpolatory and triangle meshes / quadrilateral meshes, there are stationary subdivision schemes which generate C^1 continuous limit surfaces [2, 5, 26, 6, 21, 42]. However in terms of *surface fairness*, the global C^1 continuity is sometimes not sufficient. This is why in sophisticated geometric modeling applications the quality of surfaces is usually measured by some physically based global energy functional [3, 15, 30, 34, 39].

We can combine the two techniques, i.e., the high quality surface generation by energy minimization (variational modeling, fairing) and the generation of surfaces with arbitrary topology by subdi-

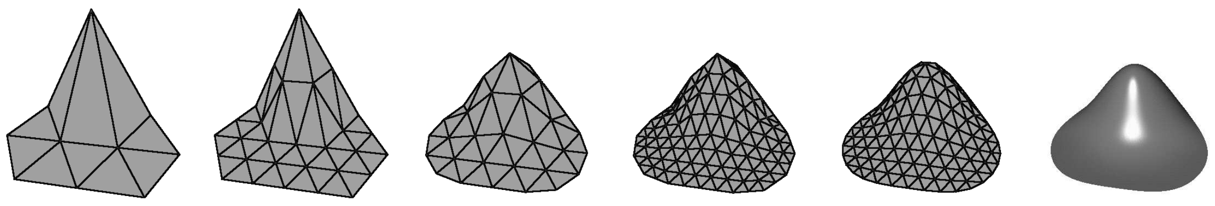

Figure 9: A sequence of meshes generated by a stationary subdivision scheme. From left to right we alternatingly apply the split operator and the smoothing operator.

vision schemes, in the following way: Instead of using the fixed smoothing rules of stationary subdivision schemes, we place the vertices in the refined mesh such that a global energy functional is minimized [20, 22, 37, 38]. Since otherwise the algorithmic structure of subdivision schemes is preserved, we call such algorithms: *variational subdivision schemes*.

As we discussed in the last section, global energy minimization requires the solution of a sparse linear system. Hence the refinement operator of a variational subdivision scheme consists of a splitting step which introduces new degrees of freedom for the optimization and the stationary smoothing operator is replaced by the Gauß-Seidel update rules of an iterative solver. As each Gauß-Seidel step merely computes a linear combination of nearby vertices, the computational complexity of variational subdivision schemes is not significantly higher than for stationary schemes if the number of Gauß-Seidel iterations is bounded (in fact, a constant factor).

On the other hand, the alternating splitting step and Gauß-Seidel iteration can be considered as a multi-level algorithm for solving the optimization problem on the finest resolution level. Hence, variational subdivision is an efficient method for computing meshes with minimum discrete curvature energy.

In order to guarantee that the resulting fine mesh interpolates the originally given vertices, the Gauß-Seidel updates must not be applied to those vertices which already belong to the initial mesh. All other vertices are allowed to move in every step of the variational refinement scheme. A variant of the variational refinement schedule is to apply the update rule only to those vertices which are inserted in the current step of the iterative refinement. If all others are kept fixed then this variant acts like an interpolatory subdivision scheme. Obviously, the variant that moves *all* vertices which are not part of the initial mesh, leads to a superior mesh quality. Nevertheless, the convergence can be accelerated if only the new vertices are moved on every refinement level.

Keeping several vertices fixed during the iterative energy minimization prevents the mesh from shrinking and collapsing. This is important because the iterative solver does not converge to a non-degenerate solution if no proper boundary conditions are set. A more detailed discussion on boundary conditions can be found in Sect. 6.

5.2 Local parameterization

The remaining open problem in the definition of a variational refinement scheme is to find a local parameterization for each vertex in order to compute the coefficients of the local divided difference operators. In the simplest case, we choose the uniform parameterization and precompute the weight coefficients for all possible valences (*umbrella algorithm* [24]). This is possible since the use of a uniform split operator for the mesh refinement leads to meshes with subdivision connectivity where all but the original vertices are regular (i.e. have valence 6).

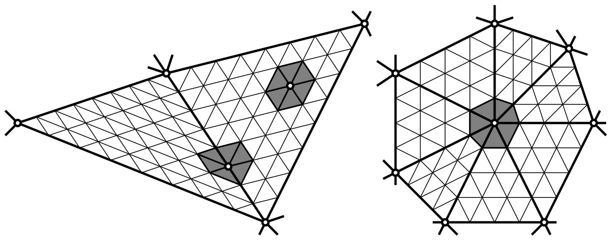

Figure 10: In a subdivision connectivity mesh, the neighborhood of a face-vertex always has the structure of an affine grid while the neighborhood of an edge-vertex looks like two affine grids meeting along a straight line. Only the neighborhood of the extraordinary vertices (right) has all degrees of freedom for the local parameterization.

The situation is more difficult if we want to apply the more sophisticated heuristics that lead to approximately isometric local parameterizations. The difficulties arise from the fact that we cannot take the spatial configuration of the neighboring vertices into account because there is no canonical starting position for the newly inserted vertices. Hence we cannot project the neighborhood into an estimated tangent plane or compute length and angles in the parameter domain. One way to get around this problem would be to compute a tentative solution, e.g., by using the uniform parameterization. Based on this solution, a better parameterization could be estimated. However, adapting the parameterization to the last iteration's solution leads to instability of the iterative algorithm as pointed out at the end of section 3.

A feasible solution to the parameterization problem has to estimate the local parameterization without knowing the actual positions of the vertices. In [22] a solution to this problem is proposed which is based on the following considerations. First, in a subdivision connectivity mesh there are three different classes of vertices. *Extraordinary vertices* correspond to the vertices of the original mesh, *edge-vertices* topologically lie on an edge of the original mesh and *face-vertices* lie somewhere in the interior of an original triangle. Due to the local (semi-) regularity of subdivision connectivity meshes we can find associated *templates* for local parameterizations (cf. Fig 10).

In the neighborhood of a face-vertex the parameter values can be sampled from an affine grid. This is natural since subdivision meshes are regular within the original base triangles. This reduces the degrees of freedom to three (two angles and one edge length). For edge-vertices a natural parameterization is piecewise affine, i.e., affine over the three triangles belonging to the one or the other adjacent base triangle respectively. Hence, there are only five degrees of freedom (four angles and one edge length). Finally for the extraordinary vertices, arbitrary parameterizations are allowed since

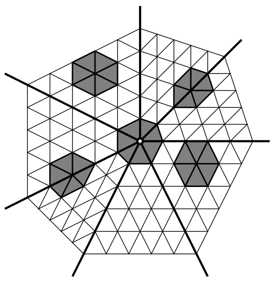

Figure 11: A local parameterization for an extraordinary vertex can be extrapolated to obtain a (semi-regular) parameterization for the whole parameter plane (*crystal seed*). This parameterization is piecewise affine in each sector. Hence the induced parameterizations for the neighboring vertices match the corresponding template definitions and can be used for the blending operation.

we cannot make any assumptions on their valence or the spatial configuration of their neighbors.

When setting up a variational subdivision scheme we start with the initial (coarse) mesh where all vertex positions are given since they provide the interpolation constraints for the optimization problem. We can estimate a local parameterization for each of the original vertices by using one of the heuristics leading to local isometry (e.g. projection into the tangent plane or exponential map parameterization). The crucial idea is then to define the local parameterizations for the edge- and face-vertices (which are inserted during the split operation) by *blending* the parameterizations of the nearest extraordinary vertices. The motivation for this approach is the notion that for smooth meshes, the distribution of the uniform samples within one base triangle should be as equal as possible. Hence the difference between neighboring parameterizations should become smaller on higher refinement levels.

A priori, the blending operation for local parameterizations is not well-defined but since we use the above templates for the edge- and face-vertices, we can simply blend the free parameters (angles and edge lengths) to obtain parameterizations for the intermediate vertices.

The local parameterization for the extraordinary vertices acts like a crystal seed which uniquely defines affine and bi-affine parameterizations for each sector in its vicinity (cf. Fig. 11). We can use the template parameters of these prototypical parameterizations and do barycentric blending along base mesh edges and across base mesh triangles (cf. Fig. 12).

Variational subdivision schemes can be implemented very efficiently. Besides the iterative application of the Gauß-Seidel update rules, there is a small overhead for the computation of the divided difference coefficients. This has to be done only once for each vertex since the local parameterization of a coarse-scale vertex does not change on higher refinement levels.

As the alternating refinement and smoothing corresponds to a multi-level algorithm for solving the optimization problem on the finest level, only a constant number of Gauß-Seidel iterations is necessary on each intermediate level. This leads to computational cost which are linear in the number of triangles of the resulting mesh. In our current implementation of variational subdivision, we generate about $10K\triangle$ per second on a 250 Mhz / R10K SGI workstation.

6 Boundary conditions

Variational subdivision schemes generate smooth meshes that interpolate the originally given vertices of the initial base mesh. The underlying discrete optimization problem becomes well-posed because the interpolation conditions make its solution unique. In the more general setting of discrete fairing in Sect. 3, we did not explicitly solve optimization problems but we merely applied the Gauß-Seidel update rules as low-pass filters to the mesh. By exploiting the special convergence behaviour of these operators, we can derive effective low-pass filters with adjustable pass-band frequency. However, in terms of mesh optimization, the proposed algorithms do not make any sense if the underlying optimization problem is ill-posed.

In order to turn the heuristic mesh filtering into an iterative algorithm for solving a well-posed optimization problem, we need to impose non-homogeneous boundary conditions. In the context of geometric modeling the typical boundary conditions are interpolation constraints for surface points and/or normal vectors.

In the global linear system characterizing the optimal mesh, each constraint provides another row. The trivial case is the interpolation of a point P in space by some mesh vertex $\mathbf{p}_i$. Here, an additional row

$$\Big(\ldots,0,1,0,\ldots\Big)\begin{pmatrix}\vdots\\ \mathbf{p}_i\\ \vdots\end{pmatrix} = P \qquad (7)$$

is added as the ith row into the global system. In the Gauß-Seidel algorithm this constraint is satisfied by simply skipping the update of $\mathbf{p}_i$ when cycling through the mesh vertices. If the point P is to be interpolated in the *interior* of a triangle $T = \triangle(\mathbf{p}_i,\mathbf{p}_j,\mathbf{p}_k)$ we have to specify the interpolation condition by barycentric coordinates α, β, and γ within the triangle T. The resulting row for the global system is

$$\Big(\ldots,\alpha,\ldots,\beta,\ldots,\gamma,\ldots\Big)\begin{pmatrix}\vdots\\ \mathbf{p}_i\\ \mathbf{p}_j\\ \mathbf{p}_k\\ \vdots\end{pmatrix} = P.$$

In practice it can be observed that the iterative solvers converge faster, if the interpolation constraints are imposed within the interior of triangles instead of at the vertices. For symmetry reasons the update steps for the three vertices $\mathbf{p}_i$, $\mathbf{p}_j$, and $\mathbf{p}_k$ are combined. The three vertices are shifted in order to re-establish the interpolation condition while altering the geometry of the mesh as little as possible. Let $\mathbf{c}$ be the correction vector

$$\mathbf{c} = P - \alpha\mathbf{p}_i - \beta\mathbf{p}_j - \gamma\mathbf{p}_k$$

then we distribute the correction among the three vertices by

$$\begin{aligned}\mathbf{p}_i &\mathrel{+}= \tfrac{\alpha}{\alpha^2+\beta^2+\gamma^2}\,\mathbf{c}\\ \mathbf{p}_j &\mathrel{+}= \tfrac{\beta}{\alpha^2+\beta^2+\gamma^2}\,\mathbf{c}\\ \mathbf{p}_k &\mathrel{+}= \tfrac{\gamma}{\alpha^2+\beta^2+\gamma^2}\,\mathbf{c}\end{aligned}$$

such that the mesh modification is minimal in the least squares sense (least norm solution for the update vectors). Notice that the three vertices are subject to an energy reducing update step induced by the global energy functional. Hence the interpolation condition is violated during the energy minimization and then re-enforced when processing the constraint. The symmetric treatment of the

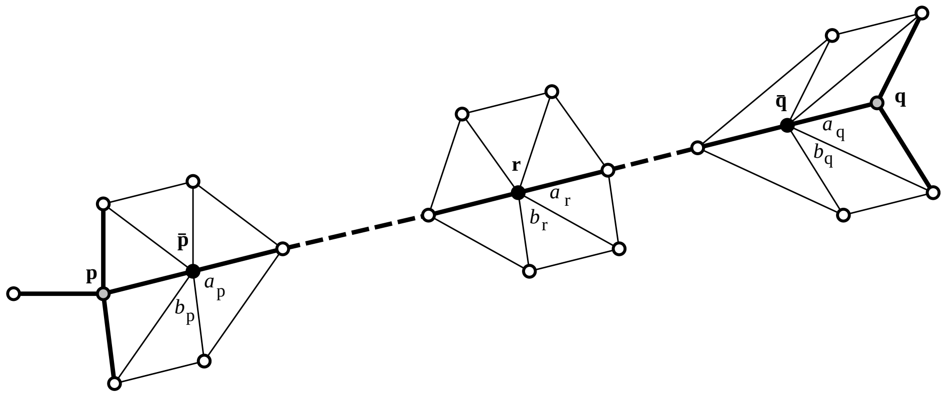

Figure 12: Defining local parameterizations by blending along a base edge. The local parameterizations of the two extraordinary vertices **p** and **q** induce bi-affine parameterizations for their direct neighbors $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$. The template parameters of these parameterizations (a_p, b_p) and (a_q, b_q) can be used for weighted blending along the base edge which connects **p** and **q**. The coefficients (a_r, b_r) then define the local parameterization for the vertex $\mathbf{r} = (1-u)\,\mathbf{p} + u\,\mathbf{q}$. The same technique can be used for the face-vertices in the interior of a base triangle where we use barycentric weights for the blending.

interpolation constraint is necessary since the corresponding row in the linear system might not be diagonal dominant and hence the iterative solving algorithm might turn instable otherwise.

The interpolation of *normal vectors* is most easily achieved by freezing all three vertices of a triangular face. This imposes a positional and normal constraint on the mesh. Other formulations for normal constraints based on the vanishing inner product of mesh edges with the normal vector, tend to make the iterative solving algorithm unstable due to lacking diagonal dominance.

In many applications, strict interpolation constraints are not natural and approximation constraints within a prescribed tolerance ε make more sense, e.g., when the input data is disturbed by noise. Instead of freezing the position of some mesh vertices, we can implement *approximation constraints* by projecting the constrained vertices back to an ε-sphere around the approximation points after every Gauß-Seidel iteration. This simple approximation scheme however uses an *isotropic* error metric which is too strict in many cases.

If we assume the interpolating mesh to be locally flat then the approximation error has to be measured in normal direction only (*anisotropic error metric*. This provides more flexibility such that the mesh can relax within the tangent plane. It mimics the effect of the parameter correction in classical scattered data interpolation [18].

For a given interpolation point P with (estimated) normal vector N, the approximation error is measured by $E = |(\mathbf{p}_i - P)^T N|$. In case this error is larger than the prescribed ε, we project $\mathbf{p}_i$ back to a plane which is parallel to the tangent plane:

$$\mathbf{p}_i \quad -= \quad \Big((\mathbf{p}_i - P)^T N - \varepsilon\Big)\, N$$

Notice that at the mesh boundaries, a different distance criterion has to be used since projecting the approximation error onto the normal vector cannot prevent shrinking effects for open surfaces.

7 Applications

With the effective and flexible techniques of *discrete fairing* and *variational subdivision schemes* available, we can use polygonal mesh representations for various standard problems within the area of sophisticated free form surface modeling. The overall objective behind all the applications we are presenting is the attempt to avoid, bypass, or at least delay the mathematically involved generation of spline CAD-models whenever it is appropriate.

Especially in the early design stages (*conceptual design*) it is usually not necessary to have a spline parameterization of a surface. The focus on polygonal mesh representations might help to free the creative designer from being confined by mathematical restrictions. In later stages the conversion into a spline model can be based on more reliable information about the intended shape. Moreover, since technical engineers are used to perform numerical simulations on polygonal approximations of the true model anyway, we also might find short-cuts that allow to speed up the turn-around cycles in the design process, e.g., we could alter the shape of a mechanical part by modifying the FE-mesh directly without converting back and forth between different CAD-models.

7.1 Scattered data interpolation

One area where the discrete fairing approach can be applied is the filling of undefined regions in a CAD model or in a measured data set. Of course, such problems can be solved by fairing schemes based on spline surfaces as well. However, the discrete fairing approach allows one to split the overall (quite involved) task into simple steps: we always start by constructing a triangle mesh defining the global topology. This is easy because no G^1 or higher boundary conditions have to be satisfied. Then we can apply the variational subdivision technique to generate a sufficiently dense piecewise regular point set on the objective surface. This part includes the refinement and energy minimization but it is almost completely automatic and does not have to be adapted to the particular application. In a last step we can fit polynomial patches to the refined data. Here we can restrict ourselves to pure fitting since the fairing part has already been taken care of during the generation of the dense data. In other words, the discrete fairing has recovered enough information about an optimal surface such that staying as close as possible to the generated points (in a least squares sense) is expected to lead to high quality surfaces.

Consider the point data in Figure 13. The very sparsely scattered points in the middle region make the task of interpolation rather difficult since the least squares matrix for a locally supported B-spline basis might become singular. To avoid this, fairing terms have to be included into the objective functional. This however brings back all the problems mentioned earlier concerning the possibly poor quality of parameter dependent energy functionals and the prohibitive complexity of non-linear optimization.

Alternatively, we can connect the points to build a spatial triangulation. Variational subdivision then recovers the missing information under the assumption that the original surface was sufficiently fair. The uneven distribution of the measured data points and the strong distortion in the initial triangulation do not cause severe instabilities since we define individual parameterizations for every vertex when computing the divided difference coefficients. These enable to take the local geometry into account.

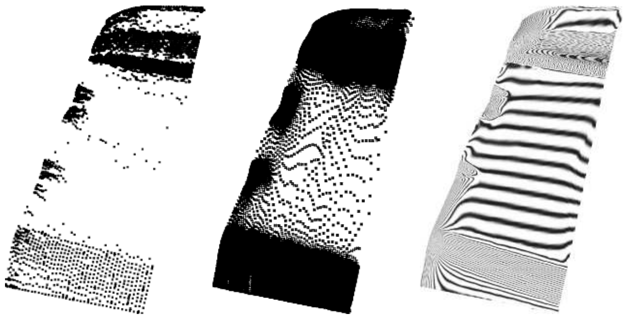
Figure 13: The original data on the left is very sparse in the middle region of the object. Triangulating the points in space and discretely fairing the iteratively refined mesh recovers more information which makes least squares approximation much easier. On the right, reflection lines on the resulting spline surface are shown.

7.2 Filleting and blending

Another standard problem in CAD is the *blending* or *filleting* between surfaces. Consider the simple configuration in Figure 14 where several plane faces (dark grey) are to be connected smoothly. We first close the gap with a coarse triangle mesh. Such a mesh can easily be constructed for any reasonable configuration with much less effort than constructing a piecewise polynomial representation. The boundary of this initial mesh is obtained by sampling the surfaces to be joined.

We then refine the mesh and, again, apply the discrete fairing machinery. The smoothness of the connection to the predefined parts of the geometry is guaranteed by letting the blend surface mesh overlap with the given faces by one row of triangles (all necessary point information is obtained by sampling the given surfaces). The vertices of the triangles belonging to the original geometry are not allowed to move but since they participate in the global fairness functional they enforce a smooth connection. As this corresponds to the discrete C^1 boundary constraints of Sect. 6 where points and normals are fixed, the discrete blending technique is able to solve the Hermite-type interpolation problem.

7.3 Applications to multiresolution modeling

In [24] we describe how to use the discrete fairing technique in the context of a multiresolution modeling tool which is able to process triangle meshes with arbitrary connectivity. The central idea is to build up multiresolution decompositions based on the hierarchical representation of meshes. Since we do not assume any special connectivity, the hierarchy has to be build from fine to coarse by using a mesh decimation algorithm.

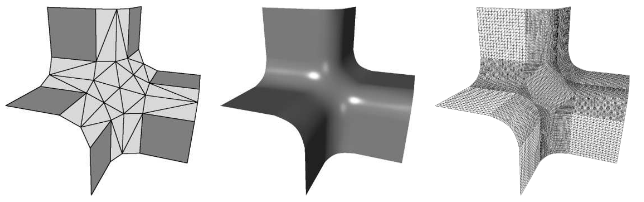
Figure 14: Creating a “monkey saddle“ blend surface joining six prescribed planes. Any blend surface can be generated by closing the gap with a triangle mesh first and then applying variational subdivision.

Standard multiresolution decompositions for meshes which are based on stationary subdivision schemes require the fine mesh to have subdivision connectivity. Starting on a coarse level of detail in such a decomposition we can either reconstruct the original data by re-inserting the detail information that has been removed during decomposition or we can apply the plain stationary subdivision operator to obtain a smoothed version of the original mesh. We call the resulting representation *coarse-to-fine hierarchy* since the coarsest base mesh completely determines the structure of all refinement levels.

For mesh hierarchies emerging from the application of a mesh decimation algorithm, the nested levels of detail are built from fine-to-coarse. The finest refinement level is the original mesh which can have arbitrary connectivity. The next coarser level is generated by performing the decimation until a certain criterion is met. The decimation is usually implemented by *edge collapses* such that we can reconstruct the original mesh by applying the corresponding *vertex split* operations in reverse order.

However, this technique only yields a *topological hierarchy* with differently *coarse* levels of detail. In the standard coarse-to-fine setting, the analogous *geometric hierarchy* with differently *smooth* levels of detail is obtained by applying the underlying stationary subdivision scheme without detail reconstruction. In the plain progressive mesh setting we can go coarse to fine only by re-inserting the previously removed vertices. This, however, necessarily reconstructs the detail and does not provide geometric levels of detail.

In order to obtain true multiresolution semantics on arbitrary meshes, we have to compute a geometric hierarchy. For this we first re-insert the vertices by vertex splits. Once the refined level of resolution is reconstructed, we apply discrete fairing with all vertices from the coarser level fixed and the new ones being allowed to move. Eventually, this yields a mesh with the same connectivity as the original one but with the geometric detail removed. This technique works since discrete fairing can be applied to meshes with arbitrary connectivity.

The geometric difference between the original mesh and its smoothed version provides a multiresolution decomposition for arbitrary meshes. For reasonable reconstruction of the detail after the global shape has been modified (*multiresolution modeling*) we have to encode the position of the original mesh's vertices relative to the local geometry of the smoother mesh (*local frames*). Notice that simply storing the difference between the original vertex position and its position after the discrete fairing is not appropriate since then the detail reconstruction can produce shape artifacts if the tangent plane of the smooth geometry changes [9, 10].

Based on this multiresolution representation for arbitrary meshes, we implemented the flexible mesh modeling tool, `HUMID`. Fig. 15 demonstrates the interaction metaphor that enables sophisticated modeling operations in realtime.

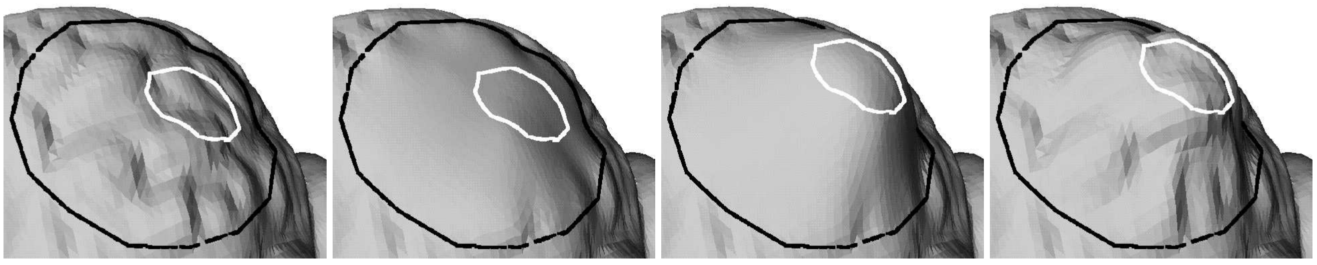

Figure 15: A flexible metaphor for multiresolution edits. On the left, the original mesh is shown. The black line defines the region of the mesh which is subject to the modification. The white line defines the handle geometry which can be moved by the designer. Both boundaries can have an arbitrary shape and hence they can, e.g., be aligned to geometric features in the mesh. The boundary and the handle impose C^1 and C^0 boundary conditions to the mesh and the smooth version of the original mesh is found by applying discrete fairing while observing these boundary constraints. The center left shows the result of the curvature minimization (the boundary and the handle are interpolated). The geometric difference between the two left meshes is stored as detail information with respect to loacal frames. Now the designer can move the handle polygon and this changes the boundary constraints for the curvature minimization. Hence the discrete fairing generates a modified smooth mesh (center right). Adding the previously stored detail information yields the final result on the right. Since we can apply fast multi-level smoothing when solving the optimization problem, the modified mesh can be updated with several frames per second during the modeling operation. Notice that all four meshes have the same connectivity.

8 Conclusion

In this paper we gave a survey of discrete fairing and variational subdivision methods which combine several powerful techniques for freeform surface design. The major features are that good approximations for geometric curvatures can be computed by using divided difference operators with respect to local parameterizations, arbitrary topology surfaces can be handled by adapting effective subdivision techniques, and highly efficient iterative algorithms for computing optimal meshes result from generalizing multi-level methods to nested grids with arbitrary connectivity.

This general technique can be used in all areas of geometric modeling and CAD/CAM where an approximation of the actual surface by a reasonably fine triangle mesh is a sufficient representation. If compatibility to standard CAD formats matters, a spline fitting post-process has to follow the discrete surface generation or modification. This fitting step can rely on densely sampled information about the intended shape if variational subdivision is used. Moreover, the discrete fairing technique enables the generalization of multiresolution modeling semantics to arbitrary connectivity meshes.

The problem to handle complex polygonal meshes has not been addressed in this paper. In priciple, we can either use mesh decimation techniques to remove vertices and triangles in geometrically flat regions of a surface or (in the context of subdivision techniques) we can avoid the exponentially increasing complexity by applying adaptive refinement techniques.

References

[1] M. do Carmo, *Differential geometry of curves and surfaces*, Prentice Hall, 1976

[2] E. Catmull, J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, CAD 10 (1978), pp. 350 – 355

[3] Celniker G. and D. Gossard, *Deformable curve and surface finite elements for free-form shape design*, SIGGRAPH 91 proceedings, 257 – 265.

[4] A. Ciampalini and P. Cignoni and C. Montani and R. Scopigno, *Multiresolution Decimation Based on Global Error*, The Visual Computer, 1997, 228 – 246

[5] D. Doo / M. Sabin, *Behaviour of Recursive Division Surfaces Near Extraordinary Points*, CAD 10 (1978), pp. 356 – 360

[6] N. Dyn, J. Gregory, D. Levin, *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Controll*, ACM Trans. Graph. 9 (1990), pp. 160–169

[7] N. Dyn, *Subdivision Schemes in Computer Aided Geometric Design*, Adv. Num. Anal. II, Wavelets, Subdivisions and Radial Functions, W.A. Light ed., Oxford Univ. Press, 1991, pp. 36–104.

[8] Farin G., *Curves and Surfaces for CAGD*, Academic Press, 1996, 4th edition

[9] D. Forsey and R. Bartels, *Hierarchical B-Spline Refinement*, SIGGRAPH 88 proceedings, 205 – 212

[10] D. Forsey and R. Bartels, *Surface Fitting with Hierarchical Splines*, ACM Transactions on Graphics, 1995, 134 – 161

[11] M. Garland and P. Heckbert, *Surface Simplification Using Quadric Error Metrics*, SIGGRAPH 97 proceedings, 209 – 218

[12] Greiner G., *Variational design and fairing of spline surfaces*, Computer Graphics Forum **13** (1994), 143–154.

[13] Guskov I., Sweldens W., Schröder P., *Multiresolution Signal Processing for Meshes*, SIGGRAPH 99 proceedings

[14] Hackbusch W., *Multi-Grid Methods and Applications*, Springer Verlag 1985, Berlin.

[15] Hagen H. and Schulze G., *Automatic smoothing with geometric surface patches*, CAGD **4** (1987), 231–235.

[16] Halstead M., Kass M., DeRose T., *Efficient Fair Interpolation Using Catmull-Clark Surfaces*, SIGGRAPH 93 proceedings, 35 – 44

[17] Hoppe H., *Progressive Meshes*, SIGGRAPH 96 proceedings, 99 – 108

[18] Hoschek J. and Lasser D. *Fundamentals of computer-aided geometric design*, A.K. Peters, 1993

[19] Klein R., *Netzgenerierung impliziter und parametrisierter Kurven und Flächen in einem objektorientierten System*, Ph. D. Thesis, CS Dept., University of Tübingen, 1995

[20] Kobbelt L., *A variational approach to subdivision*, CAGD **13** (1996), 743–761.

[21] Kobbelt L., *Interpolatory subdivision on open quadrilateral nets with arbitrary topology*, Comp. Graph. Forum **15** (1996), 409–420.

[22] Kobbelt L., *Discrete fairing*, Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces, 1997, 101 – 131

[23] Kobbelt L., Campagna S., Seidel H-P., *A general framework for mesh decimation*, Proceedings of the Graphics Interface conference 98, 43 – 50

[24] Kobbelt L., Campagna S., Vorsatz J., Seidel H-P., *Interactive Multi-Resolution Modeling on Arbitrary Meshes*, SIGGRAPH 98 proceedings, 105 – 114

[25] Lane J., Riesenfeld R., *A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces*, IEEE Trans. on Pattern Anal. and Mach. Int., 2 (1980), pp. 35–46

[26] Loop C., *Smooth subdivision surfaces based on triangles*, Master Thesis, Utah University, USA, 1987

[27] Lorensen W.E. and Cline H.E., *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, SIGGRAPH 87 proceedings, pp. 163–169

[28] B. Levy and J. Mallet, *Non-Distorted Texture Mapping for Sheared Triangulated Meshes*, SIGGRAPH 98 Proceedings, 343 – 352

[29] C. Mandal, H. Qin, B. Vemuri, *Dynamic Smooth Subdivision Surfaces for Data Visualization*, IEEE Vis 97 proceedings, 371 – 377

[30] Moreton H. and C. Séquin, *Functional optimization for fair surface design*, SIGGRAPH 92 proceedings, 167 – 176.

[31] H. Prautzsch, L. Kobbelt, *Convergence of subdivision and degree elevation*, Adv. Comp. Math. 2 (1994), J.C. Baltzer, 143 – 154

[32] U. Reif, *A unified approach to subdivision algorithms near extraordinary vertices*, CAGD 12 (1995), pp. 153 – 174

[33] R. Ronfard and J. Rossignac, *Full-Range Approximation of Triangulated Polyhedra*, Computer Graphics Forum, Proceedings of Eurographics 96, C67 – C76

[34] N. Sapidis ed., *Designing Fair Curves and Surfaces*, SIAM, Philadelphia, 1994

[35] W. Schroeder and J. Zarge and W. Lorensen, *Decimation of Triangle Meshes*, SIGGRAPH 92 proceedings, 65 – 70

[36] Taubin G., *A signal processing approach to fair surface design*, SIGGRAPH 95 proceedings, 351 – 358

[37] J. Warren, *Subdivision schemes for variational splines*, Preprint, 1998, to appear in CAGD

[38] H. Weimer, J. Warren, *Subdivision Schemes for Thin Plate Splines*, Computer Graphics Forum, Eurographics 98 conference issue, C303 – C313

[39] Welch W. and A. Witkin, *Variational surface modeling*, SIGGRAPH 92 proceedings, 157 – 166

[40] Welch W. and A. Witkin, *Free-form shape design using triangulated surfaces*, SIGGRAPH 94 proceedings, 247 – 256

[41] D. Zorin, *C^k Continuity of Subdivision Surfaces*, Ph. D. Thesis, CS Dept., California Institute of Technology, 1996

[42] D. Zorin, P. Schröder, W. Sweldens, *Interpolating Subdivision for Meshes with Arbitrary Topology*, SIGGRAPH 96 proceedings, 189 – 192