Generating Fair Meshes with G^1 Boundary Conditions

Robert Schneider
Max-Planck Institute for Computer Sciences
Computer Graphics Group
Im Stadtwald, 66123 Saarbrücken, Germany
schneider@mpi-sb.mpg.de

Leif Kobbelt
Max-Planck Institute for Computer Sciences
Computer Graphics Group
Im Stadtwald, 66123 Saarbrücken, Germany
kobbelt@mpi-sb.mpg.de

Abstract

In this paper we present a new algorithm to create fair discrete surfaces satisfying prescribed G^1 boundary constraints. All surfaces are built by discretizing a partial differential equation based on pure geometric intrinsics. The construction scheme is designed to produce meshes that are partitioned into regular domains. Using this knowledge in advance we can develop a fast iterative algorithm resulting in surfaces of high aesthetic quality that have no local mean curvature extrema in the interior.

1. Introduction

A common method to guarantee excellent surface fairness is to minimize fairness functionals based on geometric invariants. One of the best known functionals in that category are

$$\int_A \kappa_1^2 + \kappa_2^2 \, dA,$$

punishing large curvature values and

$$\int_A \left(\frac{d\kappa_1}{d\hat{e}_1}\right)^2 + \left(\frac{d\kappa_2}{d\hat{e}_2}\right)^2 dA,$$

punishing the variation of the curvature. Here κ_1 and κ_2 are the principal curvatures and $\hat{e}_1$ and $\hat{e}_2$ are the corresponding principal curvature directions. A minimization process based on such fairing functionals - this is especially true for the latter - leads to surfaces of extraordinary quality, but due to the demanding construction process, the required computation time can be enormous [17].

A popular technique to simplify this approach is to give up the parameter independence and approximate the geometric invariants with higher order derivatives. For some important fairness functionals this results in algorithms that enable the construction of a solution by solving a linear system [9].

A representant of this category is the thin plate energy

$$\int\int f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2 \, dxdy, \tag{1}$$

which can be used to create surfaces satisfying C^1 boundary conditions.

Instead of minimizing a functional, another approach first applies variational calculus to transform the minimization problem into the problem of solving a differential equation with constraints. For the functional (1) the optimal surfaces can be characterized by the partial differential equation (PDE)

$$\Delta^2 f = 0,$$

where Δ is the Laplacian operator. A solution can then be constructed with standard methods such as finite differences [2], finite elements [21] or by exploiting the knowledge that the solution can be expressed in closed form using a Green's function [1]. The PDE approach was also the starting point for operators which enable the construction of fair meshes fast enough to be used in interactive mesh modeling [10].

The parameter dependent schemes enable fast construction algorithms, but the resulting surfaces usually do not satisfy the same high aesthetic requirements as those created with geometric invariants since their quality highly depends on the underlying parameterization.

Instead of using variational calculus, the PDE approach can also be seen as a reasonable approach to the fairing problem in its own right, which is especially important for fairing based on geometric invariants. This means instead of searching for intrinsic energy functionals that lead to handy PDE's, it seems promising to search directly for simple intrinsic PDE's producing fair solutions. In our case this idea leads to the question which PDE based on geometric invariants seems suited for the creation of fair surfaces satisfying G^1 boundary constraints. We will discuss this question in section 2. Our algorithm to construct the surfaces is based on discrete data. This has been proven to be especially well suited for the construction of nonlinear splines

[13]. The idea behind the construction process is to design an algorithm that creates meshes with subdivision connectivity composed of regular patches. All patch edges besides those that lie on the boundary are geodesics and all vertices in the interior of each patch have a local 1-neighborhood with hexagonal shape. Exploiting this knowledge allows us to speed up the construction algorithm considerably. We will explain this technique in the sections 3 and 4. In section 5 we show how to calculate the regular discrete solution using an iterative scheme. Section 6 will give some implementation details and finally in section 7 we present some examples demonstrating the capability of our algorithms.

2. Problem definition

We first have to determine which PDE's seem suited to solve our fairing problem. Let us summarize what properties have to be satisfied by the PDE:

- it should be based on geometric intrinsics
- it has to be as simple as possible
- it has to allow us G^1 boundary constraints
- the resulting surface has to be fair

In many cases it is worthwhile to analyze the univariate case first, when searching for a solution of a multivariate problem. The analogous formulation of the problem in the plane is simple, because the G^1 boundary condition are especially easy to formulate. Given 2 points and 2 unit vectors, find a curve segment that interpolates the G^1-Hermite data. A differential equation that satisfies all properties in the planar case is not hard to find

$$\frac{\partial^2 \kappa}{\partial s^2} = 0, \tag{2}$$

where κ constitutes the signed planar curvature and s the arc length. The solutions are clothoids, circles and lines. Because of the nice property that the extremal curvature values occur at the border and no local curvature extrema can appear in the interior, the curve is guaranteed to be fair [16].

At this point the question arises if this formulation can be extended to describe a solution to our original bivariate problem. An analogon to the second derivative operator is the Laplace-Beltrami operator Δ_B, which extends the planar Laplacian to a smooth surface [19]. In the planar case we know that the curvature distribution defines a curve uniquely up to a rigid motion. For surfaces Bonnet's uniqueness problem [3] suggests that the mean curvature H is a curvature measure that comes next to this property. This in mind, a candidate for such a PDE seems

$$\Delta_B H = 0. \tag{3}$$

This PDE is simple and it is based on geometric intrinsics, but does it satisfy the two remaining properties?

For a regular surface one can always find a local conformal parameterization $\phi(u,v) \rightarrow \mathbb{R}^3$ [5], where conformal means local isometry up to a scalar factor, i. e. there is a $\lambda(u,v)$ such that $\lambda(u,v) =< \phi_u, \phi_u >=< \phi_v, \phi_v >$ and $< \phi_u, \phi_v >= 0$. With respect to such coordinates the Laplace-Beltrami operator can be expressed as

$$\Delta_B = \frac{1}{\lambda(u,v)^2}\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2}\right) = \frac{1}{\lambda(u,v)^2}\Delta, \tag{4}$$

so beside of a nonzero factor it simplifies to the planar Laplace Operator Δ. As is shown in [5] in conformal coordinates we further get $\Delta\phi = 2\lambda(u,v)^2 H\vec{n}$, where $\vec{n}$ is the unit normal vector of the surface. This means we can interpret (3) as a nonlinear extension of

$$\Delta^2 \phi = 0.$$

As mentioned before, this equation can be used to construct surfaces with C^1 boundary constraints, so it seems promising that (3) is a candidate that allows us G^1 boundary constraints.

From the mean value property of the Laplacian follows that the extremal values of $\Delta\phi = 0$ occur on the boundary, so because of (4) this means that the extremal mean curvature values will be reached at the border and that there are no local extrema in the interior. This extends the univariate property of (2) and hence the resulting surfaces satisfy an important fairing concept [4].
This leads us to the following problem formulation:

Problem 1 *Find a surface that satisfies the G^1 boundary constraints and whose interior solves the equation*

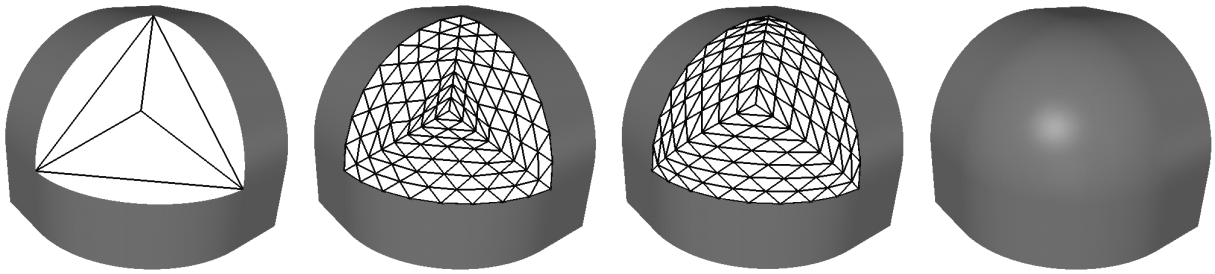
$$\Delta_B H = 0.$$

3. Discretization background

In this section we define what is meant by a discrete solution of Problem 1. The definition leads to meshes consisting of regular patches. Following an idea presented in [20] we will exploit this structure by assigning a local parameterization to each mesh vertex which allows us to create accurate approximations of the discrete solution that can be calculated very efficiently. To simplify the notations, we will assume that the border consists of one single closed curve, but the algorithm extends to a border consisting of a number of curves (Fig. 8).

3.1. Notation and definitions

Let $p(t)$ be a parameterization of our border p, let $P = \{P_1, \ldots, P_N\}$ be the set of the N vertices characterizing the



(a) (b) (c) (d)

Figure 1. Suitcase corner with 3 sides. a) A mesh $\bar{\mathrm{P}}$ with one extraordinary vertex. b) The initial mesh M^0 is constructed by using a linear subdivision scheme on $\bar{\mathrm{P}}$ taking into account that the border vertices Q have to be interpolated. c) Wireframe of the discrete solution after subdividing 3 times. d) Shaded solution after subdividing 5 times.

border of an N-sided hole and let $\bar{P}$ be a simple mesh that fills the hole defined by P. This $\bar{P}$ determines the mesh topology of the discrete solution (Fig. 1).

The set P divides the border curve p into N segments. Let us approximate each of the N segments by a polygon with r edges, producing a polygon Q with vertices $Q = \{Q_1, \ldots, Q_{rN}\}$ and the property $P \subset Q \subset p$. Let M be a mesh with vertices M_i whose topology is equivalent to a r-fold uniformly subdivided $\bar{P}$ interpolating the border points Q (Fig. 1 b) c)). Because of the subdivision connectivity structure we can partition the points M_i into 4 classes:

- The border vertices Q are also called interpolation vertices. Their position is fixed.
- The vertices of $\bar{P}$ that are not in P are called extraordinary vertices.
- The vertices that can be assigned to an interior edge of $\bar{P}$ are called edge vertices.
- The remaining vertices that can be assigned to the interior of a triangle face of $\bar{P}$ are called inner vertices.

For simplicity, we will denote all M_i that are no border vertices as free vertices. For vertices of valence v let $M_{i,l}, l = 1..v$ denote their adjacent neighbors. Because of the subdivision connectivity structure, edge and inner vertices M_i always have valence 6. For edge vertices we define the convention that the adjacent vertices are arranged such that $M_{i,1}$ and $M_{i,4}$ are aligned to the same edge. Let H_i resp. $H_{i,l}$ be the discrete mean curvature at M_i resp. $M_{i,l}$ and let $\vec{n}_i$ be a discrete unit normal vector at M_i. In section 4.4 and 4.8 we explain how we construct this intrinsic values.

Let us further define a mean value operator g_p for all free vertices. For edge and inner vertices this operator is

$$g_p(M_i) = \begin{cases} 1/6 \sum_{l=1}^{6} M_{i,l}, & \text{if } M_i \text{ is an inner vertex,} \\ 1/2\ (M_{i,1} + M_{i,4}), & \text{if } M_i \text{ is an edge vertex,} \end{cases}$$

for extraordinary vertices we will give its definition in section 4.7.

Finally we can define what is meant by a discrete solution of our problem

Definition 1 *A mesh M is called a regular discrete solution of problem 1, if the following conditions are satisfied:*

1. *At all interpolation points of M the G^1 boundary constraints are satisfied.*
2. *The vertices M_i of M should be regularly distributed. For all free vertices, there should be a $t_i \in \mathbb{R}$ such that $M_i = g_p(M_i) + t_i \vec{n}_i$.*
3. *At every free vertex M_i we have $\Delta_B(H_i) = 0$.*

A geodesic line has the property that its main normal vector is parallel to the surface normal vector at every point of the line [5], so we note that condition 2 implies that the edge vertices will form a geodesic net in the interior of the mesh M. The mean value operator g_p at the extraordinary vertices strongly influences the geodesic net structure. Since the inner vertices are distributed as regularly as possible, this means condition 2 guarantees that the solution is partitioned into regular patches (Fig. 1 c)), which is a property that gives us much information about the mesh M in advance. In [20] meshes consisting of regular patches were used to efficiently construct surfaces with piecewise linear mean curvature distribution, we will modify this idea to create a very efficient construction algorithm for a regular discrete solution.

3.2. Choosing the interpolation vertices

Because of condition 2 in definition 1, the discrete solution will be a collection of regular mesh patches. The interpolation vertices Q_i have to reflect that, otherwise the mesh structure will be distorted near the border and this would decrease the quality of our local parameterizations. Hence, a reasonable distribution of the interpolation vertices should satisfy

$$||Q_i - Q_{i-1}|| \approx ||Q_{i+1} - Q_i||$$

for all $Q_i \notin P$. For piecewise smooth boundary curves, uniform sampling of each of the N smooth border sides using an approximated arc length parameterization will satisfy that condition.

4. Discretization of the geometric invariants and the Laplace-Beltrami operator

For inner, edge and extraordinary vertices, our discretization technique is based on the well known idea to construct a local quadratic least square approximation of the discrete data and estimate all needed values from this approximation.

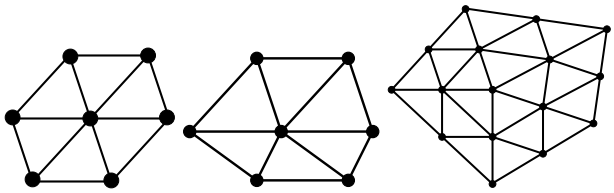


Figure 2. The parameter domains for inner, edge and extraordinary vertices.

4.1. General parameterization strategy

Instead of recalculating a local parameterization during each step in the iteration, we exploit the fact that our meshes are well structured to determine a local parameterization in advance, thus accelerating our algorithms considerably. The decision, which local parameterization should be assigned to a vertex Q_i was influenced by three mayor constraints: simplicity, regularity and uniqueness of the quadratic approximation. This constraints lead us to the following local parameterization classes. For inner vertices, the parameter domain is a regular hexagon and for edge vertices it is composed of two regular hexagon halves. Calculating the matrices needed for a least square approximation in both cases, it is easy to see, that this parameterizations always lead to a unique least square approximation. Only at the interpolation vertices it is not always sufficient to use the 1-neighborhood. This is obvious, if the valence v of the vertex is 3 or 4, but even if the valence is higher, the least square approximation can fail in the 1-neighborhood. This problem is similar to the problem that occurs when local quadratic least square approximation is used for arbitrary mesh fairing. In [25] it was proposed to determine the condition number of the matrix needed for the least square approximation and to reduce the number of the basis functions if the problem is ill conditioned. Instead of that approach, we can exploit the fact that the extraordinary vertices are located in a piecewise regular neighborhood. Using the complete 2-neighborhood consisting of regular sectors for vertices with small valence or parts of this neighborhood for vertices of high valence we can make the least square approximation unique.

4.2. Construction of a local parameterization

We now have to define how to actually calculate the local parameterizations for free vertices. In [9] Kobbelt presented an approach to approximate a local isometric parameterization by exploiting the subdivision connectivity structure of a subdivided mesh that minimizes the thin plate energy. In our case this algorithm is even better suited to construct good approximations for a local parameterization. The iteration process leads to meshes that are partitioned into regular regions making it much simpler to get good local approximations by parameter domain blending. The idea of the algorithm is to start with parameter domain seeds at those vertices, where the regular mesh structure is interrupted and to complete the local parameterizations at the remaining vertices using blending operators. In our case the parameter domain seeds have to be calculated at the extraordinary vertices and the vertices P at the border. At extraordinary vertices we can determine a local parameter domain seed by applying an exponential map on its 1-neighborhood. This operator maps a 1-neighborhood into the plane, such that the distances of the vertex to its neighbors and the ratio of the adjacent angles are preserved [25]. The 2-disc domain for extraordinary vertices as presented in Figure 2 then results by applying one linear subdivision step. At the vertices P we do not use the exponential map because the local neighborhood is open, but here we can exploit the fact that we know the tangent plane. We project the neighborhood into the tangent plane and scale the resulting vectors such that the vertex-neighbor distances are still preserved.

The remaining local parameter domains for the edge and inner vertices (as we will see in section 4.8 there is no need for a local parameterization at the remaining interpolation vertices) are then derived by linear blending the seed triangles, for a detailed description of this blending process see [9]. Instead of blending the triangle parameters de-

scribed there, we got improved results when blending the edge lengths of the triangles.

4.3. Determining a least square approximation

Let us first consider the functional case. Given a local triangular parameterization domain and function values f_i at the vertices of the domain, we search for a quadratic function

$$f(x) = a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6$$

which minimizes the least square error at the domain vertices. This problem is well known and leads to a linear system $A\vec{a} = \vec{f}$, where $\vec{a}$ is the vector of the coefficients a_j and $\vec{f}$ is the vector of the function values f_i. If the domain has more than 6 vertices and the matrix A has full rank, this linear system is overdetermined and the best least square solution can be expressed as

$$\vec{a} = (A^tA)^{-1}A^tf. \tag{5}$$

In our case - we have 3-D point information instead of function values f_i - the least square approximation is constructed by applying the functional scheme to the x, y and z values of the vertices simultaneously.

4.4. Discretization of the geometric invariants

Using the coefficients of the least square approximation, we can determine the normal vector and the first and second fundamental form at the center vertex of the parameter domain and can express the mean curvature H as [5]

$$H = \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2}, \tag{6}$$

where E, F, G are the coefficients of the first fundamental form and e, f, g are those of the second fundamental form.

When we are only interested in intrinsic values, we can use the fact that an affine mapping of the parameter domain does only change the parameterization of our quadratic approximation, but no geometric invariants. This means to merely determine the normal and the mean curvature for inner vertices, it is sufficient to use an equilateral hexagon as parameter domain, thus allowing us an especially simple calculation of those values for such vertices. This is another nice advantage of the local parameterization construction.

4.5. Discretization of the Laplace-Beltrami operator

The parameterization technique presented in [9] approximates a local isometric parameterization. In our case, where we have patches with especially nice regular structure, the approximation quality of such an approach improves considerably. Moreover we do not even need a local isometric parameterization. In our case it is already sufficient to have a local conformal parameterization, a wrong scaling of the local parameter domains will not influence the solution. This means we can interpret our local parameterizations as good approximations of local conformal parameterizations.

For an conformal parameterization the equation $\Delta_B H = 0$ is equivalent to $\Delta H = 0$, so in our case we can approximate Δ_B with the planar Laplacian operator Δ. Following the discretization technique of Δ presented in [9], we can approximate Δ_B using the local quadratic approximation resulting in

$$\Delta = 2(a_1 + a_2),$$

where the constant factor is of no interest in our case. Using expression (5) this means we can find coefficients λ_i and $\lambda_{i,l}, l = 1, \ldots, v$ such that

$$\Delta_B H_i = \lambda_i H_i + \sum_{l=1}^{v} \lambda_{i,l} H_{i,l}. \tag{7}$$

For the special case when the domain is an equilateral triangle and using the fact that multiplying with a nonzero factors does not influence the results we arrive at the simple umbrella operator

$$\Delta H_i = -H_i + \sum_{l=1}^{6} \frac{1}{6} H_{i,l}. \tag{8}$$

which was used in [10] for interactive mesh modeling.

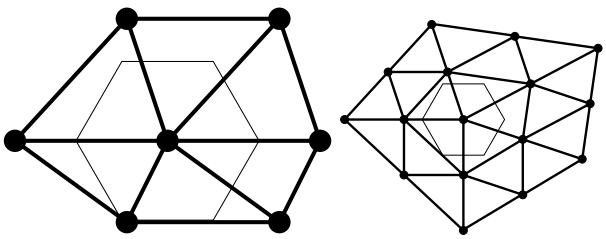


Figure 3. At edge and extraordinary vertices we can restrict the update step to the simple case where the parameter domain is an equilateral hexagon. The new parameter domain is chosen to be the maximum equilateral hexagon in the interior of the local 1-neighborhood of the center vertex.

4.6. Handling of edge and extraordinary vertices

As mentioned above, the discretization of the geometric invariants and the Laplace-Beltrami operator is especially

comfortable when the parameter domain is an equilateral hexagon, but this special case has more nice properties. The equation $\Delta_B H_i = 0$ using (7) defines a sparse linear system in the unknowns H_i and such systems are usually optimally solved using iterative multigrid algorithms. Assuming the parameter domain at every vertex to be an equilateral hexagon would produce a matrix that is symmetric and positive definite and hence we would be able to use very simple iterative multigrid solver as for example based on Gauss-Seidel iteration. Gauss-Seidel is known to be a slow iterative solver when only applied at the finest level, but a fast and comfortable solver in combination with multigrid methods [10]. Unfortunately the matrix is no longer symmetric and positive definite when working with arbitrary parameter domains, hence it is no longer guaranteed to be always non singular and if the linear system is solvable it would require more involved iterative schemes like e. g. biconjugate gradient iteration.

We discovered that only the discretization at edge and extraordinary vertices affected the convergence of our multigrid solver. For such vertices we found a very nice and simple technique to avoid such problems. We first calculate the least square approximation using the mean curvature values H_i and $H_{i,l}, l = 1, \ldots, v$ of a vertex M_i and then sample this least square approximation at a regular hexagon that lies in the interior of the parameter domain (Fig. 3) resulting in the new values $\tilde{H}_{i,l}, l = 1, \ldots, 6$. An iteration step at edge and extraordinary vertices is then led back to the comfortable symmetric case (8) by using the new sampled $\tilde{H}_{i,l}$ instead of the original $H_{i,l}$. Since the $\tilde{H}_{i,l}$ lie on the quadratic least square approximation for the original parameter domain, a recalculation of the least square approximation using the new domain and the new neighbors will reproduce the same function, hence this sampling technique does not lead to false local informations. But an update step (8) with this new sampled points $\tilde{H}_{i,l}$ has again the simplicity and stability advantages of the equilateral case.
Note: It is important that the equilateral hexagon lies in the interior of the original domain, otherwise the sampling step is no longer an interpolation but an extrapolation and hence loses its stability.

4.7. Mean value operator for extraordinary vertices

The geodesic net defined by the edge vertices is determined by the position of the extraordinary vertices, so the mean value operator g_p at such vertices defines the partitioning of the discrete solution. The resampling technique presented in section 4.6 can also be applied to the local quadratic least square approximation of the 1-neighborhood of M_i resulting in new vertices $\tilde{M}_{i,l}$. We found it a reasonable strategy that the geodesic net of the discrete solution should have a shape that is similar to those of the simple mesh $\bar{P}$, so we defined the mean value operator here to be the center of the subsampled vertices $\tilde{M}_{i,l}$. Hence, for the final definition of the operator g_p we get

$$g_p(M_i) = \begin{cases} \frac{1}{6} \sum_{l=1}^{6} M_{i,l}, & M_i \text{ inner vertex,} \\ \frac{1}{2} (M_{i,1} + M_{i,4}), & M_i \text{ edge vertex,} \\ \frac{1}{6} \sum_{l=1}^{6} \tilde{M}_{i,l}, & M_i \text{ extraord. vertex.} \end{cases}$$

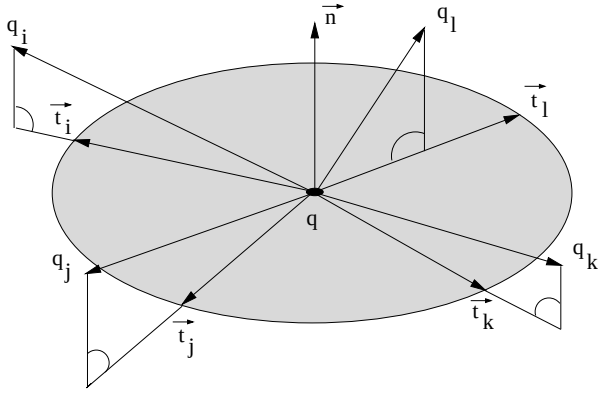


Figure 4. Projecting the neighborhood of $\mathbf{q}$ onto the plane defined by $\vec{\mathbf{n}}$ and normalizing the results we get the normal curvature directions $\vec{\mathbf{t}}_\mathbf{i}$.

4.8. Discretization of the mean curvature at interpolation vertices

At the interpolation vertices we have a completely different situation. Here we only have one sided information of the mesh neighborhood, but therefore we know the tangent plane of the final surface. Since these vertices do not have to be updated during the iteration process, we only need the value of the discrete curvature.

Even if we would not only know the G^1 boundary conditions but the surrounding surface, we must not use outer points for the curvature estimation since the resulting surface does not have to be curvature continuous at the border. In [10] a rough approximation of the solution of $\Delta^2 f = 0$ with C^1 boundary constraints was solved by applying the umbrella operator for interior and border vertices, but this simple strategy is not suitable for high quality surfaces.

A curvature estimation algorithm that seems ideal for our needs was presented by Moreton and Sequin [17]. The idea behind this approach is to use the fact that normal curvature distribution can not be arbitrary, but is determined by the Euler-formula [5]. We describe how to discretize the mean curvature at a mesh vertex q, when the normal vector $\vec{n}$ at q is known.

Let $q_j, j = 1 \dots m$ be the vertices adjacent to q and let $\vec{b}_x$ and $\vec{b}_y$ be an arbitrary orthonormal basis of the plane defined by the normal $\vec{n}$. To each vertex q_j we can assign a unit direction vector $\vec{t}_j$ by projecting q_j into the plane and scaling this projection to unit length (Fig. 4). For each vertex q_j we can now estimate a normal curvature $\tilde{\kappa}_j$ as the inverse of the circle radius defined by q, q_j and t_j.

Using the Euler formula, we can express the normal curvature κ_n for a direction $\vec{t}$ by the principal curvatures κ_1 and κ_2 and the principal curvature directions $\vec{e}_1$ and $\vec{e}_2$. Let t_x and t_y be the coordinates of t in the basis $\vec{b}_x, \vec{b}_y$ and let e_x and e_y be the coordinates of $\vec{e}_1$, then the normal curvature can be expressed as

$$\kappa_n = \begin{pmatrix} t_x \\ t_y \end{pmatrix}^t \cdot K \cdot \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

with

$$K = \begin{bmatrix} e_x & e_y \\ -e_y & e_x \end{bmatrix} \cdot \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \cdot \begin{bmatrix} e_x & e_y \\ -e_y & e_x \end{bmatrix}^{-1}.$$

The idea of Moreton and Sequin is to use the normal curvatures $\tilde{\kappa}_j$ to create a linear system and find estimates for the unknown curvature values by determining the least square solution. Let $t_{j,x}$ and $t_{j,y}$ denote the coordinates of $\vec{t}_j$, then we get

$$Ax = b$$

where

$$A = \begin{bmatrix} t_{0,x}^2 & t_{0,x}t_{0,y} & t_{0,y}^2 \\ t_{1,x}^2 & t_{1,x}t_{1,y} & t_{1,y}^2 \\ \cdot & \cdot & \cdot \\ t_{m,x}^2 & t_{m,x}t_{m,y} & t_{m,y}^2 \end{bmatrix}, \; b = \begin{bmatrix} \tilde{\kappa}_0 \\ \tilde{\kappa}_1 \\ \cdot \\ \tilde{\kappa}_m \end{bmatrix}$$

and

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} e_x^2\kappa_1 + e_y^2\kappa_2 \\ 2e_xe_y(\kappa_1 - \kappa_2) \\ e_x^2\kappa_2 + e_y^2\kappa_1 \end{pmatrix}$$

Using the normal equations, the least square solution $\tilde{x}$ can be expressed as $\tilde{x} = (A^tA)^{-1}A^tb$. Since $H = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{1}{2}(x_0 + x_2)$ this means we can approximate the mean curvature as $\tilde{H} = \frac{1}{2}(\tilde{x}_0 + \tilde{x}_2)$.

5. Surface construction

We construct the surface using an iterative solver $M^k \to M^{k+1}$ starting with an initial mesh M^0. In each iteration step every free vertex M_i^k is moved to a new position M_i^{k+1}, while the interpolation vertices Q do not change their position. The equations determining the new position M_i^{k+1} are nonlinear unlike the equations usually occurring in iterative algorithms for parameter dependent functionals. The idea behind the construction of high quality surfaces based on nonlinear equations is to work with the original equations and linearize the construction process instead of simplifying the functionals itself [18, 6, 20]. This is often possible by keeping those values constant that will only have small changes during one single iteration step.

In our case we used the mean curvature linearization technique presented in [20] which is well suited when local parameterizations are available. In [20] this technique was used to enable a fast construction algorithm for meshes with subdivision connectivity having piecewise linear mean curvature distribution. As was mentioned there, this technique can be applied for two principle iteration approaches. The first approach linearizes the nonlinear equations locally, the second is based on the idea to use the available global information about the curvature distribution. We will now show how these ideas can be used to derive an iteration scheme for our fairing problem.

5.1. Direct approach

When updating the vertex M_i^k in an iteration step, the new position is determined by the two equations

$$M_i^{k+1} = g_p(M_i^k) + t_i\vec{n}_i^k \;\text{ and }\; \Delta_B(H_i^{k+1}) = 0.$$

The first equation reduces the 3-variate to a univariate problem and the second equation is then used to determine the vertex position. Because of our discretization of the Laplace-Beltrami operator, the latter condition can be expressed as

$$\lambda_i H_i^{k+1} + \sum_{l=1}^{6} \lambda_{i,l} H_{i,l}^{k+1} = 0. \tag{9}$$

To linearize this nonlinear equation, we exploit the fact that the updated mesh vertices M_i^{k+1} will be near M_i^k in the $k+1$ iteration and hence the vertices and the mean curvature values will not change much. This in mind, we use $H_{i,l}^k$ instead of $H_{i,l}^{k+1}$ in equation (9) and simplify H_i^{k+1} by using the vertices M_i^k instead of M_i^{k+1} to calculate the coefficients E, F and G of the first fundamental form. So using (6) equation (9) leads to

$$\frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2} = -\frac{1}{\lambda_i}\sum_{l=1}^{6} \lambda_{i,l} H_{i,l}^k$$

and we have to find a t_i such that this equation is satisfied. Applying (5) and replacing the normal vector $\vec{n}_i^{k+1}$ by $\vec{n}_i^k$ when calculating the coefficients of the second fundamental form e, f and g, this equation becomes linear in t_i and we can determine the new vertex M_i^{k+1} by solving this linear equation.

Since the vertices are updated along the surface normals, this approach can be interpreted as some kind of curvature flow, where the speed is determined by a local equation system. As was pointed out by Desbrun et al. [6], flows depending only on local surface properties do not have to be numerically stable for large steps during the iteration process. To allow larger update steps, Desbrun et al. used the backward Euler method to develop an algorithm called implicit integration for fairing of arbitrary meshes based on Laplacian and on mean curvature flow. The idea behind this approach is to use global surface information instead of considering the local neighborhood only. In [20] Schneider and Kobbelt presented an algorithm called indirect iteration, that efficiently exploits global surface properties for subdivision connectivity meshes, when fairing is based on linear curvature distribution. We show in the following that this idea can be extended to our fairing problem as well.

5.2. Indirect iteration

The central idea of the indirect iteration is to decouple the curvature information from the actual mesh geometry. In our case this means if we could iterate $\Delta_B(H_i^{k+1}) = 0$ without affecting the curvature information H_j^{k+1} along the interpolation points $Q_j^{k+1} = Q_j$, we would arrive at a solution of the following Dirichlet problem:
Find scalar values H_i^{k+1} such that $\Delta_B(H_i^{k+1}) = 0$ with given boundary values H_j^{k+1} at the interpolation vertices. So the analogon of the linear indirect approach presented in [20] can be formulated as follows:

- Determine the actual curvature values H_j^k at the interpolation vertices Q_j using the technique described in section 4.8.
- Determine mean curvature values $\tilde{H}_i^{k+1}$ for the free vertices by solving the Dirichlet problem $\Delta_B(\tilde{H}_i^{k+1}) = 0$ using the determined mean curvature values H_j^k at the border.
- Use the $\tilde{H}_i^{k+1}$ to update the free vertices M_i^k. In the update step the new position is then determined by the two equations $M_i^{k+1} = g_p(M_i^k) + t_i \vec{n}_i^k$ and $H_i^{k+1} = \tilde{H}_i^{k+1}$.

The nonlinear equation is linearized by analogous techniques as described in the direct approach, thus the new position can again be determined by solving a linear equation for t_i. As was mentioned in [20] it is important to restart at step 1 after step 3. An iteration scheme that only iterates step 3 would not converge in general, since $\tilde{H}_i^{k+1}$ are only estimates of the according values of the final solution.

In this formulation the indirect iteration is very expensive, because in every iteration step we have to solve a linear system for the next mean curvature estimates, but this can be avoided. We know that the actual discrete mean curvature values in the interior are approximations for the solution of the linear system occurring in step 2 and the quality of the approximation improves rapidly during the iteration process. Further it is not necessary to solve the linear system exactly, it is sufficient to approximate the solution. This means we can change the first and second step:

- Determine the actual curvature values H_i^k at the interpolation vertices *and* at all free vertices.
- Use the H_i^k at the free vertices as initial values and determine new mean curvature values $\tilde{H}_i^{k+1}$ by applying s iteration steps of an iterative solver for the Dirichlet problem.

If the actual mesh M^k is already a good approximation of the discrete solution, it is not necessary to choose a large s.

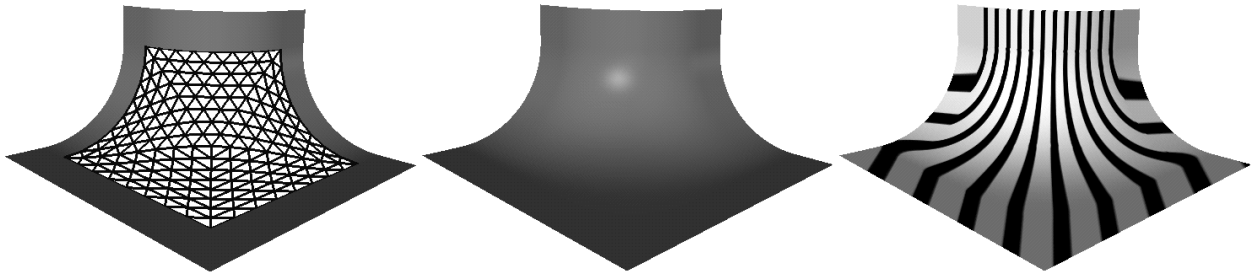
6. Details

Because of the subdivision connectivity of the Mesh M, all examples were created using a multigrid iteration, a technique that has proven to be excellent when hierarchic structures are available [8, 9]. Here a solution is first created on a coarse level and then serves as starting point for the iteration scheme in the next hierarchy level. Applying this strategy across several levels of the hierarchy, the convergence speed of the construction process can increase dramatically. Since the high frequency error is smoothed because of the hierachical structure, we can use the simple Gauss-Seidel algorithm when iterating step 2 of the indirect iteration scheme.

The mesh topology of the constructed discrete solution is determined by $\bar{P}$. For the N-sided hole filling problem such a triangular mesh can often be generated automatically, for holes with a more complex boundary or for problems with more than one boundary curve, such a triangular mesh has to be provided by the user.

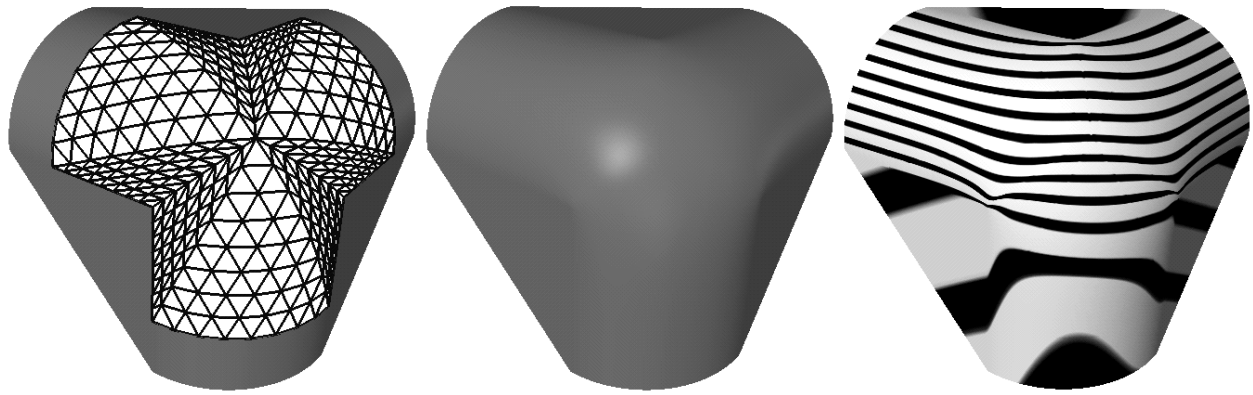
The initial mesh M^0 is constructed using a linear subdivision scheme, this means we apply linear subdivision on the mesh $\bar{P}$ but take into account that M^0 has to interpolate the vertices Q (Fig. 1 b)). This is not only fast and convenient, but it guarantees that the vertices M_i^0 are regularly distributed. Since this approach produces peaks at the extraordinary vertices, it would not be a reasonable idea to start iterating on the finest level, but it works fine in the multigrid scheme since we only have to subdivide to the coarsest level.

As mentioned earlier the direct approach is not suited if large update steps can occur, therefore our implementation is based on the indirect iteration scheme. Since large update steps are only necessary in coarse hierarchy levels, our iteration number s is chosen small for fine hierarchy levels and large for the coarsest level. Instead of prescribing s,



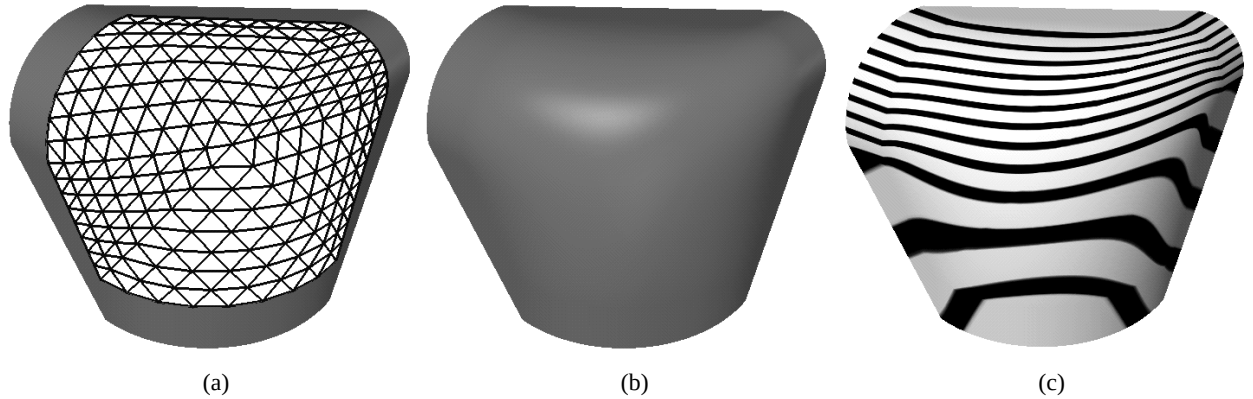
(a) (b) (c)

Figure 5. The classical house corner problem. a) A regular discrete solution of a 3 times subdivided mesh $\bar{\mathrm{P}}$ with 1 extraordinary vertex and 5 sides. b) Shaded discrete solution of a 5 times subdivided mesh. c) Lines of reflection of a 5 times subdivided mesh indicating that the solution satisfies the G^1 boundary conditions and is at least G^2 in the interior.



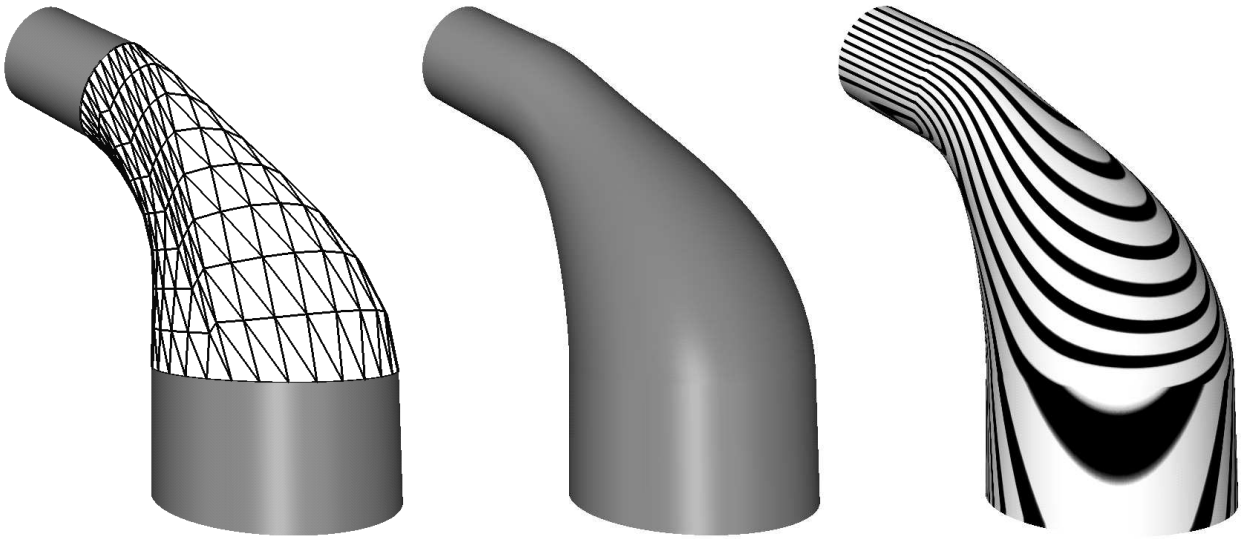
(a) (b) (c)

Figure 6. Suitcase corner with 9 sides and 3 sharp edges at the border. a) Discrete solution of a 3 times subdivided mesh. b) and c) Because of the sharp edges we subdivided 6 times. Again the lines of reflection indicate the G^1 continuity at the border and the higher continuity in the interior.



(a) (b) (c)

Figure 7. Suitcase corner with 6 sides where 3 sides are defined by cylinders with different sizes. a) Discrete solution of a 3 times subdivided mesh. b) and c) Discrete solution of a 5 times subdivided mesh.



(a) (b) (c)

Figure 8. Blending two cylinders of different size. a) Discrete solution of a 3 times subdivided mesh. b) and c) Discrete solution of a 5 times subdivided mesh.

one can iterate on each level until all free vertices satisfy $\Delta_B(H_i^k) < \epsilon$ for a given tolerance ϵ.

When solving $H_i^{k+1} = \tilde{H}_i^{k+1}$ in the update step of the indirect iteration, we apply the resampling technique presented in section 4.6 also to the neighbor vertices of $M_{i,l}^k$ resulting in new vertices $\tilde{M}_{i,l}^k$ and use these vertices to update M_i^k. Thus we again profit from the simplicity and stability advantages of the equilateral case.

At edge and extraordinary vertices we cached the matrices $(A^tA)^{-1}A^t$ in (5) thus allowing us a fast least square approximation. For inner vertices this is not necessary, since the resulting formulas here are simple enough to be calculated on the fly.

When starting on the coarsest level, the initial local parameterization is constructed using the vertices of $\bar{P}$ as seeds. Since the resulting parameter domains are only rough approximations, we recalculate the parameter domains once when the mesh approximates the final solution at a coarse level. In the examples this was done when the solution of a 3 times subdivided mesh had been constructed.

7. Examples

In CAGD there are many fields of application where G^1 boundary constraints naturally appear. One of the most important is the N-sided hole filling problem, which is without doubt one of the most studied problems in geometric modeling [7, 14, 23, 12]. Solutions for some classical N-sided hole problems obtained with the proposed techniques can be seen in the figures. The lines of reflection indicate the G^2 continuity in the interior (smooth lines) and G^1 continuity at the border (continuous lines) in all our examples. As can be seen in Figure 8 our algorithm is not restricted to a border that consists of one single closed curve only. In this example we calculated the solution for a blending problem.

The construction algorithm is implemented in Java 1.2 for Windows running on a PII with 400MHz. For our presented examples the average iteration time for a 3 times subdivided mesh was ≈ 1.0 seconds and for 5 times subdivided meshes ≈ 4.0 seconds. An optimized C implementation would improve these timings without doubt.

8. Conclusion

Despite the fact that the construction idea is based on pure geometric intrinsics and the resulting high quality fairness, we presented a discrete fairing algorithm that is fast enough to become interesting in interactive design. The resulting meshes have an especially interesting structure, they are composed of regular patches where the interior patch edges form geodesics on the discrete surface.

This property allowed us to construct an efficient iteration algorithm, which should also be very useful when one does not need a discrete solution, but a surface composed of spline patches. In such a case the piecewise regular structure may be exploited when approximating or interpolating the mesh using polynomial splines. The presented algorithm allowed us G^1 boundary constraints, by increasing

the degree of the intrinsic PDE it should be possible to allow higher geometric continuity constraints at the border using analogous techniques.

References

[1] Bloor, M. I. G., and M. J. Wilson, Interactive Design Using Partial Differential Equations, in *Designing Fair Curves and Surfaces*, ed. N. S. Sapidis, SIAM, Philadelphia, 1994.

[2] Bloor, M. I. G., and M. J. Wilson, Using partial differential equations to generate free-form surfaces, Computer Aided Design, 22 (1990), 202–212.

[3] Bonnet, O., *Mémoire sur la théorie des surfaces applicables*, J. Éc. Polyt 42, 1867.

[4] Burchard, H. G., J. A. Ayers, W. H. Frey, and N. S. Sapidis, Approximation with Aesthetic Constraints, in *Designing Fair Curves and Surfaces*, ed. N. S. Sapidis, SIAM, Philadelphia, 1994.

[5] do Carmo, M. P., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Inc Englewood Cliffs, New Jersey, 1993.

[6] Desbrun, M., M. Meyer, P. Schröder, and A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, SIGGRAPH 99 Conference Proceedings, 317-324.

[7] Gregory, J. A., V. K. H. Lau, and J. Zhou, Smooth Parametric Surfaces and n-Sided Patches, in *Computation of Curves and Surfaces*, ed. W. Dahmen, M. Gasca and C. A. Micchelli, Kluwer Academic, 1990, 457–498.

[8] Hackbusch, W., *Iterative Lösung grosser schwachbesetzter Gleichungssysteme*, Teubner Verlag, Stuttgart, 1993.

[9] Kobbelt, L., Discrete fairing, Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces, 101–131, 1996.

[10] Kobbelt, L., S. Campagna, J. Vorsatz, and H-P. Seidel, Interactive Multi-Resolution Modeling on Arbitrary Meshes, Proceedings of SIGGRAPH '98, 105–114.

[11] Kobbelt, L., A variational approach to subdivision, *Computer Aided Geometric Design* 13 (1996), 743–761.

[12] Levin, A., Filling N-sided holes using combined subdivision schemes, to appear in Proceedings of the Saint-Malo Conference '99.

[13] Malcolm, M. A., On the computation of nonlinear spline functions, SIAM J. Numer. Anal. 15, 254–282, 1977.

[14] Malraison, P., A Bibliography for N-sided surfaces, in *Mathematics of Surfaces VIII*, Information Geometers Limited, Winchester, 419–430, 1998.

[15] Mehlum, E., Nonlinear splines, *Computer Aided Geometric Design*, Academic Press, London, 173–207, 1974.

[16] Meek, D. S., and D. J. Walton, Planar G^1 Hermite interpolation with spirals, *Computer Aided Geometric Design* 15 (1998), 787–801.

[17] Moreton, H. P., C. H. Séquin, Functional optimization for fair surface design, SIGGRAPH 92 Conference Proceedings, 167–176.

[18] Pinkall, U., K. Polthier, Computing discrete minimal surfaces and their conjugates, Experimental Mathematics 2, 15–36, 1993.

[19] Rauch, J., *Partial Differential Equations*, Springer-Verlag, New York, 1991.

[20] Schneider, R., and L. Kobbelt, Discrete Fairing of Curves and Surfaces based on linear Curvature Distribution, to appear in Proceedings of the Saint-Malo Conference '99.

[21] Schwarz, H. R., *Methode der finiten Elemente*, Teubner Verlag, Stuttgart, 1991.

[22] Taubin, G., A signal processing approach to fair surface design, SIGGRAPH 95 Conference Proceedings, 351–358.

[23] Varady, T., and A. Rockwood, Geometric construction for setback vertex blending, Computer Aided Design 29 (1997), 413–425.

[24] Weimer, H., J. Warren, Subdivision Schemes for Thin Plate Splines, Computer Graphics Forum 17 (1998), 303–314.

[25] Welch, W., and A. Witkin, Free-Form shape design using triangulated surfaces, SIGGRAPH 94 Conference Proceedings, 247–256.