Discrete Fairing of Curves and Surfaces Based on Linear Curvature Distribution

R. Schneider and L. Kobbelt

Abstract. In the planar case, one possibility to create a high quality curve that interpolates a given set of points is to use a clothoid spline, which is a curvature continuous curve with linear curvature segments. In the first part of the paper we develop an efficient fairing algorithm that calculates the discrete analogon of a closed clothoid spline. In the second part we show how this discrete linear curvature concept can be extended to create a fairing scheme for the construction of a triangle mesh that interpolates the vertices of a given closed polyhedron of arbitrary topology.

§1. Introduction

In many fields of computer-aided geometric design (CAGD) one is interested in constructing curves and surfaces that satisfy aesthetic requirements. A common method to create fair objects is to minimize fairness metrics, but since high quality fairness functionals are usually based on geometric invariants, the minimization algorithms can become computationally very expensive [10].

A popular technique to simplify this approach is to give up the parameter independence by approximating 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 equation system, but such curves and surfaces are in most cases not as fair as those depending on geometric invariants only.

An interesting approach to simplify the construction process without giving up the geometric intrinsics is to use variational calculus to derive differential equations characterizing the solution of a minimization problem. Mehlum [8] used this idea to approximate a minimal energy curve (MEC), which minimizes the functional $\int \kappa''(s)^2 ds$, with piecewise arc segments. In [1] Brunnet and Kiefer exploited the property that a segment of a MEC between two interpolation points satisfies the differential equation $\kappa'' + \frac{1}{2}\kappa^3 = 0$ to speed up the construction process using lookup tables.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 371–380.

ISBN 0-8265-1356-5.

The usage of such differential equations based on geometric invariants can be seen as a reasonable approach to the fairing problem in its own right, and it can be applied to curves as well as surfaces. For planar curves, one of the simplest differential equations is $\kappa'' = 0$. Assuming arc length parameterization, this equation is only satisfied by lines, circles and clothoids. A curvature continuous curve that consists of parts of such elements is called a clothoid spline. Most algorithms for the construction of such curves are based on techniques that construct the corresponding line, circle and clothoid segments of the spline [9]. This is possible for planar curves, but the idea does not extend to surfaces.

In this paper we will first present a fast algorithm to construct an interpolating closed discrete clothoid spline (DCS) purely based on its characteristic differential equation. Our algorithm uses discrete data, because this has been proven to be especially well suited for the efficient construction of nonlinear splines [7]. The efficiency of our construction process is largely based on an algorithm called the indirect approach. This algorithm decouples the curvature information from the actual geometry by exploiting the fact that the curvature distribution itself is a discrete linear spline. Besides the speed of the planar algorithm, it has another very important property: it directly extends to the construction of surfaces!

In Section 2 we give a short review of related work. Section 3 will first give an exact definition of a DCS and then address its efficient computation. Section 4 shows how the planar algorithm extends to surfaces. We construct fair surfaces interpolating the vertices of a given closed polyhedron of arbitrary topology by assuming a piecewise linear mean curvature distribution with respect to a natural parameterization.

§2. Related Work

A very interesting algorithm addressing the problem of constructing fair curves and surfaces with interpolated constraints was presented by Moreton and Sequin. In [10] they minimized fairing functionals, whose fairness measure punishes the variation of the curvature. The quality of their minimal variational splines is extraordinary good, but due to their extremely demanding construction process, the computation time needed is enormous. Their curves and surfaces consisted of polynomial patches.

In contrast to this approach, most fairing and nonlinear splines algorithms are based on discrete data. Malcolm [7] extended the discrete linear spline concept to efficiently calculate a discrete MEC for functional data. In [13] Welch and Witkin presented a nonlinear fairing algorithm for meshes of arbitrary connectivity, based on the strain energy of a thin elastic plate. Recently, Desbrun et al. [3] used the mean curvature flow to derive a discrete fairing algorithm for smoothing of arbitrary connected meshes.

In most fairing algorithms the original fairing functionals are approximated by simpler parameter dependent functionals. In recent years this idea was combined with the concept of subdivision surfaces to create variational

subdivision splines [12,6,4]. Especially [4] should be emphasized here, since our local parameterization needed in Section 4 is largely based on ideas presented there.

Instead of minimizing a functional, Taubin [11] proposed a signal processing approach to create a fast discrete fairing algorithm for arbitrary connectivity meshes. Closely related but based on a different approach is the idea presented by Kobbelt et al. [5], where a uniform discretization of the Laplace operator is used for interactive mesh modeling. In [3] Desbrun et al. showed that a more sophisticated discretization of the Laplace operator can lead to improved results. The direct iteration approach presented later can be seen as a nonlinear generalization of such schemes.

§3. Discrete Clothoid Splines

In this section we show how to interpolate a closed planar polygon using a discretization of a clothoid spline. Although the algorithms extends to open polygons with G^1 boundary conditions in a straightforward manner, we only consider closed curves, because that case directly extends to surfaces.

3.1 Notation and definitions

In the following we denote the vertices of the polygon that has to be interpolated with $P = \{P_1, \ldots, P_n\}$. Let $Q = \{Q_1, \ldots, Q_m\}$ be the vertices of a refined polygon with $P \subset Q$. The discrete curvature at a point Q_i is defined to be the reciprocal value of the circle radius interpolating the points Q_{i-1}, Q_i, Q_{i+1} or 0 if the points lie on a straight line, which leads to the well known formula

$$\kappa_i = 2\frac{det(Q_i - Q_{i-1}, Q_{i+1} - Q_i)}{||Q_i - Q_{i-1}||\ ||Q_{i+1} - Q_i||\ ||Q_{i+1} - Q_{i-1}||}. \quad (1)$$

离散曲率计算

Definition 1. *A polygon Q with $P \subset Q$ is called a* discrete clothoid spline *(DCS), if the following conditions are satisfied:*

1) *The interior of each segment is arc length parameterized*
   $||Q_{i-1} - Q_i|| = ||Q_i - Q_{i+1}||$, *whenever* $Q_i \notin P$,
2) *The discrete curvature is piecewise linear:* 离散曲率分段线性
   $\Delta^2\kappa_i = \kappa_{i-1} - 2\kappa_i + \kappa_{i+1} = 0$, *whenever* $Q_i \notin P$.

We will construct a DCS using an iteration procedure $Q^k \to Q^{k+1}$ until the above conditions are sufficiently satisfied. The iteration starts with an initial polygon Q^0 that interpolates the P_i (Fig. 1).

3.2 Direct approach

In this approach the location of a vertex Q_i^{k+1} only depends on the local neighborhood $Q_{i-2}^k, \ldots, Q_{i+2}^k$ of its predecessor Q_i^k. To satisfy condition 1 in Definition 1 locally, Q_i^{k+1} has to lie on the perpendicular bisector between

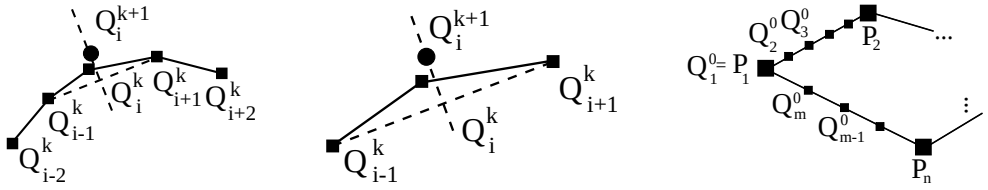

Fig. 1. The left side shows the update steps for Q_i^k in the direct and the indirect iteration. The right side shows the initial polygon Q^0. We simply subsampled the polygon P.

Q_{i-1}^k and Q_{i+1}^k (Fig. 1), reducing the 2 variate problem to a univariate one. Further we require Q_i^{k+1} to satisfy $\Delta^2 \kappa_i^{k+1} = 0$. Unfortunately this equation is nonlinear in the coordinates of the vertices Q^{k+1}, but since the update $Q_i^k \rightarrow Q_i^{k+1}$ in the k-th iteration step will be small, we can linearize the equation by using the coordinates of Q^k instead of Q^{k+1} in the denominator of equation (1). This allows us to update every Q_i^k solving a 2×2 linear system for Q_i^{k+1} during one iteration step.

3.3 Indirect approach

If we decouple the curvature values κ_i^{k+1} from the actual polygonal geometry and perform the direct iteration scheme only on the curvature values by solving $\Delta^2 \kappa_i^{k+1} = 0$, the curvature plot would converge to a piecewise linear function. The idea of the indirect approach is to use this property to create a new iteration scheme. This is done by dividing each iteration step into two sub-steps, in the first sub-step we estimate a continuous piecewise linear curvature distribution, and in the second sub-step we use those as boundary conditions to update the points Q^k. We get the curvature distribution by estimating the curvature of the polygon Q^k at every point P_j, and interpolate this curvature values linearly across the interior of the segments, assigning a curvature estimation $\tilde{\kappa}_i^{k+1}$ to every vertex Q_i^k. In the second step this curvature information is used to update the points Q_i^k, where the position of the new point Q_i^{k+1} is determined by $\tilde{\kappa}_i^{k+1}$ and the neighborhood $Q_{i-1}^k, Q_i^k, Q_{i+1}^k$ of its predecessor Q_i^k (Fig. 1). Again one degree of freedom is reduced by restricting Q_i^{k+1} to lie on the perpendicular bisector between the points Q_{i-1}^k and Q_{i+1}^k, and further we require Q_i^{k+1} to satisfy $\kappa_i^{k+1} = \tilde{\kappa}_i^{k+1}$. Using an analogous linearization technique as in the previous case, we can again update every Q_i^k solving a 2×2 linear system. It is very important to alternate both sub-steps during each iteration. Since the $\tilde{\kappa}_i^{k+1}$ are only estimates of the final DCS, an iteration process that merely iterates the second sub-step might not converge.

Compared to the direct approach, the indirect iteration scheme has some interesting advantages. The interpolation points imply forces everywhere during one iteration step; there is no slow propagation as in the direct approach. The direct approach acts due to its local formulation as a lowpass filter, here because of its global strategy the indirect iteration is much better suited to reduce the low frequency error. One iteration step is cheaper than one iteration step of the direct approach. This fact will become much more important

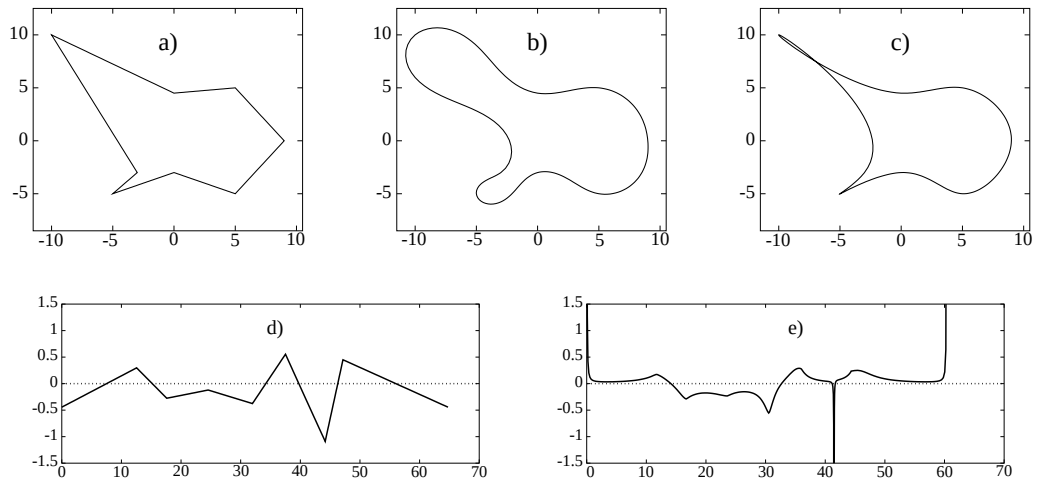

Fig. 2. a) A polygon P with 8 vertices. b) The DCS interpolating the vertices of P. The structure of the polygon is equivalent to a 5 times subdivided P. The calculation time was < 0.1 seconds. c) Periodic C^2 cubic spline interpolation of P. d) Curvature plot of the DCS. e) Curvature plot of the periodic cubic spline.

if we extend the algorithms to the surface case. Finally, since the estimated curvatures are constructed by linear interpolation they are well bounded, a fact that stabilizes the iteration procedure.

3.4 Details and remarks

Both schemes were implemented using a generic multigrid scheme, a technique that has proven to be valuable when hierarchical structures are available [4]. We first constructed a solution on a coarse level, and used a prolongation of this coarse solution as starting point for an iteration on a finer level. Applying this strategy across several levels of the hierarchy, the convergence of both approaches are increased dramatically. On the coarsest hierarchy level the initial polygon was constructed by linearly sampling the polygon P (Fig. 1). The curvature values at the interpolation points $Q_i \in P$ needed in the estimation step were calculated by simply applying formula (1) on Q_i. Without the multigrid approach, such a simple strategy would not be reasonable, because for fine polygons the occurring curvature values could become too large. Our final iteration scheme for the DCS is a combination of the two primary multigrid iterations. Such an approach has the advantage that the estimated curvatures are usually better, since the high frequency error near the interpolation points are smoothed out by the direct step. This multigrid hybrid approach turned out to be a very reliable solver.

All examples throughout the paper were implemented in Java 1.2 for Windows running on a PII with 400MHz. In Figure 2 we see that our algorithm still produces a fair solution in cases where classical periodic cubic spline interpolation using chord length parameterization fails completely. In this example we can also see why it is not only comfortable to be able to start with such a simple initial polygon Q^0, but also can be very important in some

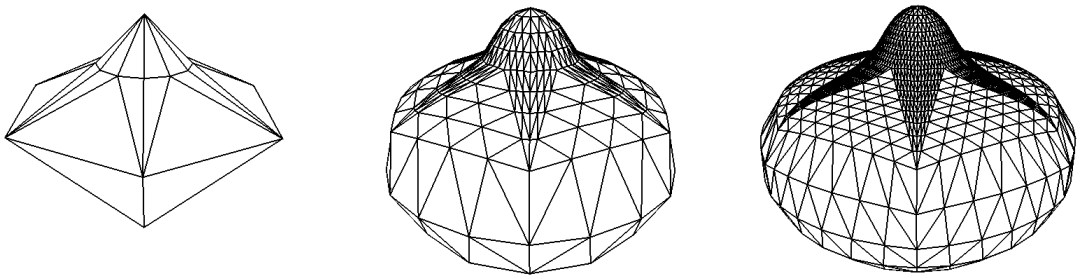

Fig. 3. Wireframe of a mesh P and the solutions of a 2 times resp. 3 times subdivided mesh. Iteration times: middle $\approx 0.2s$, right $\approx 0.4s$.

cases. Our algorithm searches the solution next to the starting polygon, so taking Q^0 to be a a linearly sampled P will prefer solutions without loops. A technique that would use the periodic cubic spline to initialize Q^0 in this example would either fail because of the enourmous curvatures that turn up, or produce a DCS with loops.

It is well known that clothoids can be parameterized using Fresnel Integrals [9]. If such a continuous solution is preferred, the discrete solution could be used to derive a closed form representation.

§4. Closed Surfaces of Arbitrary Topology

In this section we want to show how to extend the concept of planar clothoid splines to closed surfaces of arbitrary topology. Given a closed polyhedron P of arbitrary topology, we construct a discrete fair surface that interpolates the vertices of P. To be able to extend the planar algorithms, we first have to determine what curvature measure for surfaces has to be used. According to Bonnet's uniqueness problem, the mean curvature seems to be most appropriate.

4.1 Notation and definitions

In the following, we denote the vertices of the polyhedron that has to be interpolated by $P = \{P_1, \ldots, P_n\}$. Let $Q = \{Q_1, \ldots, Q_m\}$ be the vertices of a refined polyhedron with $P \subset Q$, where we require the topological mesh structure of Q to be equivalent to a uniformly subdivided P (Fig. 3). Because of this structure, we can partition the vertices of Q into three classes. Vertices $Q_i \in P$ are called interpolation vertices, Q_i that can be assigned to an edge of P are called edge vertices, and the remaining points are called inner vertices. Edge and inner vertices Q_i always have valence 6; for those points let $Q_{i,l}, l = 1..6$ be their adjacent vertices. For edge vertices we make the convention that the adjacent vertices are arranged such that $Q_{i,1}$ and $Q_{i,4}$ are edge or interpolation vertices. Let H_i resp. $H_{i,l}$ be the discrete mean curvature at Q_i resp. $Q_{i,l}$ and let n_i be the discrete unit normal vector at Q_i. Finally, let us define the following operators:

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

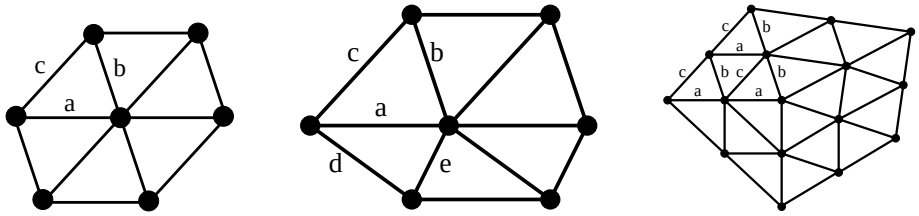

Fig. 4. Left) Parameter domain for an inner vertex. Middle) Parameter domain for an edge vertex. Right) The parameter domain for an interpolation vertex of valence 5 is a subset of this domain.

$$g_h(H_i) = \begin{cases} 1/6 \sum_{l=1}^{6} H_{i,l}, & \text{if } Q_i \text{ is an inner vertex,} \\ 1/2\,(H_{i,1} + H_{i,4}), & \text{if } Q_i \text{ is an edge vertex.} \end{cases}$$

We are now in the position to extend Definition 1 to the surface case.

Definition 2. *The polyhedron Q will be called a solution to our discrete fairing problem, if the following conditions are satisfied:*

1) *The vertices Q_i are regularly distributed. For all edge and inner vertices, there should be a $t_i \in \mathbb{R}$ such that $Q_i = g_p(Q_i) + t_i \vec{n}_i$,*
2) *The curvature is linearly distributed over each face: $g_h(H_i) = H_i$.*

Condition 2 is straightforward, it states that the mean curvature is changing linearly with respect to the barycentric parameterization of the inner and edge vertices. Condition 1 is more difficult to understand. It generalizes the DCS property that vertices in the interior of a segment were on the perpendicular bisector between its neighbors.

As in the planar case, we construct the surfaces using an iteration $Q^k \rightarrow Q^{k+1}$, starting with an initial polyhedron Q^0.

4.2 Discretization of the geometric invariants

Our discretization technique is based on the well-known idea of constructing a local quadratic least square approximation of the discrete data and estimating the needed geometric invariants from the first and second fundamental forms. For the mean curvature H, this means [2]

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

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

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 raising the speed of 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. These constraints lead us to the following local parameterization classes (Fig. 4). For inner vertices, the

parameter domain is a regular hexagon, 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. Using parts of the 2-neighborhood consisting of regular sectors solves that problem. For example, the least square approximation is already unique by using the 1-neighborhood and one complete sector of the 2-disc (marked with a,b,c in Fig. 4).

Since we are only interested in intrinsic values, we can use the fact that an affine mapping of the parameter domain only changes the parameterization of our quadratic approximation, but not geometric invariants. This fact allowed us to chose one fixed equilateral hexagon to serve as parameter domain for all inner vertices, and guaranteed a simple update step for such points.

At edge and interpolation vertices, we calculated the parameter domains by applying the blending technique proposed in [4]. This algorithm constructs a local parameterization that adapts to the underlying geometry defined by P, thus approximating a local isometric parameterization. For a detailed description of that algorithm we have to refer to that work.

4.3 Direct approach

When updating the vertex Q_i^k in an iteration step, the new position is determined by the two equations $Q_i^{k+1} = g_p(Q_i^k) + t_i \vec{n}_i^k$ and $H_i^{k+1} = g_h(H_i^k)$. With analogous arguments as in the curve setting, we linearized the mean curvature expression (2) for H_i^{k+1} by using the vertices of Q^k to calculate the coefficients of the first fundamental form E, F, G and by replacing the normal n_i^{k+1} by n_i^k when calculating the coefficients of the second fundamental form. Finally, the vertex Q_i^{k+1} is determined by solving a linear equation for t_i.

4.4 Indirect approach

The planar indirect iteration scheme directly extends to the surface setting. In the first sub-step we estimate the mean curvature of the polyhedron Q^k at the interpolation vertices and use simple linear interpolation to assign a mean curvature value $\tilde{H}_i^{k+1}$ to every edge and inner vertex Q_i^k. This means the estimated values satisfy $\tilde{H}_i^{k+1} = g_h(\tilde{H}_i^{k+1})$. In the second sub-step we then use these estimated values to determine Q_i^{k+1} using the formula $H_i^{k+1} = \tilde{H}_i^{k+1}$ with the same linearization method as in the direct approach.

4.5 Details and remarks

Since the vertices are updated along the surface normals, the direct iteration 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. [3], flows depending only on local surface properties do not have to be numerically stable for large steps during the iteration process. To allow larger

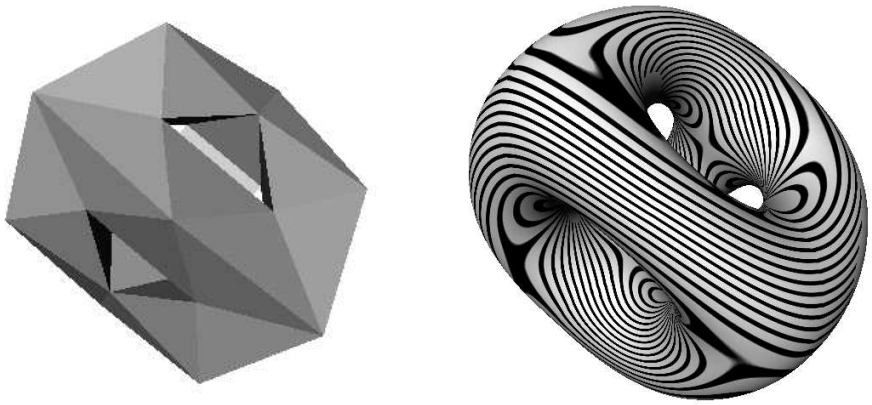

Fig. 5. Left) Tetra Thing mesh. Right) Solution of the multigrid indirect iteration after the Tetra Thing has been subdivided 5 times. The reflection lines indicate that the surface is quasi G^2 continuous. Iteration time: ≈ 4 seconds.

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. The indirect approach follows that idea and exploits global surface properties efficiently.

In the surface case the indirect approach shows its true power. The estimation sub-step is mostly simple linear interpolation and using the estimated curvature, we only need the 1-neighborhood of a vertex during its update process. The update process for interior vertices is especially simple because of the equilateral hexagon as parameter domain. For edge and interpolation vertices, we calculated the least square matrices before the first iteration step and cached the result. Our surface examples were created using a mutigrid iteration scheme based on the indirect approach.

As mentioned earlier, the construction of our local parameterization is based on the technique presented in [4], where divided difference operators to create discrete thin plate splines had to be derived. A comparison of our results showed that the ideas presented in this paper allow the construction of considerably improved meshes without increasing the computation time. The improved quality is due to the fact that our discrete surfaces are based on geometric invariants instead of second order partial derivatives, and thus the error made by guessing an isometric local parameterization in advance has less influence. Due to the fact that we estimated the local parameterization once, we do not get an exact G^2 continuity at the edges and interpolating points, but as can be seen in Figure 5, our surfaces are good approximations to G^2 surfaces. To achieve true G^2 continuity, one would have to increase the effort for the handling of edge and interpolating vertices. Future work will investigate if such efforts are worthwhile.

§5. Conclusion

The quality of the resulting curves and surfaces is superior to approaches based on quadratic functionals as the thin plate energy due to the more sophisticated approximation of geometric invariants. Although our objects are nonlinear splines, they can be calculated fast enough to be applicable in interactive design. The presented ideas are optimally suited for the construction of curves and surfaces with piecewise linear curvature distribution, but could also be extended to higher order discrete fairing approaches.

References

1. Brunnett G., and J. Kiefer, Interpolation with minimal-energy splines, Computer-Aided Design **26**(2) (1994), 137–144.
2. do Carmo, M. P., *Differential Geometry of Curves and Surfaces* Prentice-Hall, Inc Englewood Cliffs, New Jersey, 1993.
3. 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.
4. Kobbelt, L., Discrete fairing, Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces '97, 101–131.
5. Kobbelt, L., S. Campagna, J. Vorsatz, and H. P. Seidel, Interactive multi-resolution modeling on arbitrary meshes, *SIGGRAPH 97 Conference Proceedings*, 105–114.
6. Kobbelt, L., A variational approach to subdivision, Comput. Aided Geom. Design **13** (1996), 743–761.
7. Malcolm M. A., On the computation of nonlinear spline functions, SIAM J. Numer. Anal. **15** (1977), 254–282.
8. Mehlum, E., Nonlinear splines, *Computer Aided Geometric Design*, Academic Press, London, 173–207, 1974.
9. Meek, D. S., and D. J. Walton, A guided clothoid spline, Comput. Aided Geom. Design **8** (1991), 163–174.
10. Moreton H. P., and C. H. Séquin, Functional optimization for fair surface design, *SIGGRAPH 92 Conference Proceedings*, 167–176.
11. Taubin G., A signal processing approach to fair surface design, *SIGGRAPH 95 Conference Proceedings*, 351–358.
12. Weimer, H., and J. Warren, Subdivision Schemes for Thin Plate Splines, Computer Graphics Forum **17** (1998), 303–314.
13. Welch, W., and A. Witkin, Free-Form shape design using triangulated surfaces, *SIGGRAPH 94 Conference Proceedings*, 247–256.

Leif Kobbelt and Robert Schneider
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
{kobbelt, schneider}@mpi-sb.mpg.de