Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

Shape Optimization Using Reflection Lines

E. Tosun, Y. I. Gingold, J. Reisman, D. Zorin

New York University, USA

Abstract

Many common objects have highly reflective metallic or painted finishes. Their appearance is primarily defined by the distortion the curved shape of the surface introduces in the reflections of surrounding objects.
Reflection lines are commonly used for surface interrogation, as they capture many essential aspects of reflection distortion directly, and clearly show surface imperfections that may be hard to see with conventional lighting. In this paper, we propose the use of functionals based on reflection lines for mesh optimization and editing. We describe a simple and efficient discretization of such functionals based on screen-space surface parameterization, and we demonstrate how such discrete functionals can be used for several types of surface editing operations.

1. Introduction

Many human-made surfaces have highly reflective finishes: cars, kitchen appliances, lamps and jewelry are common examples. The appearance of such objects is primarily defined by the reflections of other objects. Reflections are quite sensitive to surface shape and depend on local surface quantities (normals and curvatures) as well as viewer location.

Reflection *lines* are a widely used interrogation tool for surfaces. Conceptually, these are obtained by computing reflections of a set of long linear parallel light sources, aligned with a fixed direction. Visualizing such reflections make many types of surface irregularities apparent. Reflection lines can be thought of as a special type of reflected environment, capturing the distortion introduced by the curved shape of the surface for a particular direction of features in the environment. Reflection lines are widely used in the automotive industry, and were also found to be useful in biomedical engineering as a tool for cornea shape reconstruction [HBKM96]. Advances in graphics hardware made interactive reflection line rendering widely accessible and easy to implement.

Figure 1: *An example of reflection line optimization.*

The process of evaluating surface quality, for which reflection lines are most commonly used, is complimentary to shape design. In most cases, the designer defines the surface by manipulating spline or subdivision control points, or other types of shape controls, then evaluates the quality using interrogation tools and repeats the process until the desired quality is achieved. The controls of the shape have an indirect effect on the quality measure and in the case of reflective surfaces, it may be hard to guess how the shape should be modified to achieve a desired effect. A common alternative is to formulate the surface editing problem as an optimization problem minimizing a quality functional measuring deviation from desired behavior.

Reflection lines provide a convenient framework for building such functionals for reflective surfaces. For reasons we discuss in Section 2, arbitrary manipulation of reflections is not possible. Furthermore, recovering the surface from an arbitrarily chosen distortion of a reflected image is not always possible either. However, in most cases, it is possible to find a surface producing a given pattern of reflection lines. By choosing a reflection line direction, the user chooses what feature direction in the environment can be considered most important. For example, horizontal and vertical lines are most common in urban and indoor environments, and it makes sense to use these directions for surface optimization.

In this paper we present a system for interactive surface modeling based on reflection line manipulation. We show how to discretize reflection lines on arbitrary meshes, and demonstrate that a relatively simple discretization is sufficient for shape modeling purposes, provided that a suitable normal estimation algorithm is used. We propose a numerical technique for solving the problem at rates adequate for

interactive surface manipulation. Our approach is based on two insights. First, an arbitrary mesh can be locally parameterized over the image plane away from silhouette edges; this makes it possible to reduce the number of degrees of freedom used in optimization, and greatly simplifies expressions. Second, we observe that a simple triangle-based discretization of second-order quantities using only vertex degrees of freedom can be used to compute second-order derivatives of the surface parameterization, which leads to a fast and efficient matrix assembly. We demonstrate how reflection line manipulation can be used to smooth and warp reflection lines, change reflection line density, and create surfaces with a desired reflection line pattern.

Related work. Different types of reflection lines were extensively studied in the geometric modeling literature. Some of the earliest work is described in [Kla80], where a differential-geometric description of reflection lines and an analytic expression for the variation of these lines is derived; [KK88] describes a technique for adjusting families of spline curves defining a surface based on reflection line changes. Reflection lines were also used as a surface interrogation tool in [Poe84, HHS*92, GLW96, GOZ95, TF97]. Theisel has shown that isophotes and reflection lines can be viewed as subclasses of a more general class [The01]. Functionals similar to reflection-line functionals are common in shape-from-shading vision literature (e.g. [Hor86]); however, the goal there is to reconstruct an unknown surface entirely from possibly noisy image data, rather than modify an existing mesh.

Surface fairing functionals based on characteristic line patterns are explored in [HBKM96] and [LGS99] (reflection lines) and [YBP97, CF98] (highlight lines). A technique for manipulating reflection line shape directly is described in [DB04]. In all cases, these techniques were applied to NURBS surfaces. The formulation we use is closest to [LGS99], which applies reflection line optimization to B-spline height fields; we apply a similar formulation to general meshes and present algorithms that allow us to achieve interactive performance.

Various types of *fairing* functionals for surfaces and their discretizations for spline surfaces, subdivision surfaces, and meshes are considered in [CG91, WW94, Kob00, SK01] and many other papers. Fairing functionals are used increasingly in interactive surface deformation settings, in particular for general meshes (e.g. [BK04]; see [BS07] for an excellent survey). An efficient, robust and accurate discretization of the Hessian of the function defined on an arbitrary mesh is central to our work. This problem is closely related to the problem of defining shape operators on meshes; *discrete geometry* approaches (e.g. [PP93, MDSB03, CSM03]) play an important role in making variational techniques for meshes sufficiently fast for interactive applications. [GGRZ06] contains a detailed survey of different types of approaches; our approach builds on [GSH*04].

Using user-defined reflection fields for surface optimization is similar to gradient and Laplacian deformation techniques ([SLCO*04], [YZX*04]) in that the optimization functional depends on the initial mesh geometry (in our case, through the reflection function).

2. Reflection functionals

In this section we present the relevant basic mathematics of reflections and functionals based on reflection lines. The formulations we use are similar to the ones that were used in [LGS99] for optimization of reflection lines of tensor-product B-spline height fields.

2.1. Reflection line function

We consider a somewhat simplified formulation of reflection lines, with both the viewer and the light sources located at infinity. The reflection line pattern in our model is created by long line-shaped light sources aligned with a unit length vector **a** (Figure 2). Each light source can be identified by a direction in the plane perpendicular to **a**. If we fix a zero direction, each direction corresponds to an angle θ in the range $-\pi \dots \pi$. For a point **p** of a surface, let **n** be the normal, and let **v** be the view direction, which we assume to be nonparallel to **a**. In this notation, the reflection direction at **p** is given by $\mathbf{r} = (2/|\mathbf{n}|^2)\,((\mathbf{n}\cdot\mathbf{v})\mathbf{n} - \mathbf{v})$. (We do not assume the normal to be unit length).

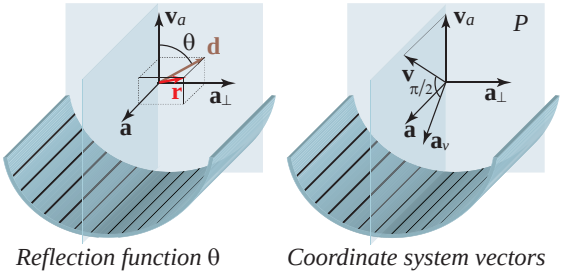

Figure 2: *Vectors used in the definition of the reflection line function θ.*

We define the *reflection line function* to be a scalar function on the surface which assigns to each point the angle θ between a zero direction and the direction **d** to the linear light source corresponding to reflected direction **r**. This direction is obtained by projecting **r** to the plane P perpendicular to **a**: $\mathbf{d} = \mathbf{r} - (\mathbf{r}\cdot\mathbf{a})\mathbf{a}$. Let $\mathbf{v}_a$ be the projection of the viewing direction **v** to the plane P, and let $\mathbf{a}_\perp$ be perpendicular to $\mathbf{v}_a$ in the plane P,

$$\mathbf{v}_a = [\mathbf{v} - (\mathbf{v}\cdot\mathbf{a})\mathbf{a}]_{norm}, \quad \mathbf{a}_\perp = \mathbf{a}\times\mathbf{v}_a$$

where $[\cdot]_{norm}$ denotes normalization. We use $\mathbf{v}_a$ as the zero direction; in this case the reflection line function is given by

$$\theta = \arctan((\mathbf{r}\cdot\mathbf{a}_\perp),(\mathbf{r}\cdot\mathbf{v}_a)) \qquad (1)$$

where $\arctan(y,x)$ produces values in the range $-\pi \dots \pi$. A reflection line is defined by a constant θ value. As reflection lines are view-dependent, it makes more sense to consider them as functions on the image plane, rather than the surface itself. The function is defined everywhere except at the

points where $\mathbf{r}$ is parallel to the light direction $\mathbf{a}$. The gradient of θ is of primary importance: the direction of the reflection lines is perpendicular to $\nabla\theta$. (Note that $|\nabla\theta|$ measures the local density of reflection lines.)

Coordinate formulation. One of the properties distinguishing the reflection line optimization from most fairing problems is that there are fixed spatial directions $\mathbf{a}$ and $\mathbf{v}$ which are a part of the problem formulation. Projections to these directions are natural choices of variables. We observe that for *silhouette* points, for which the normal is perpendicular to the view direction, perturbing the surface does not affect the reflection line function corresponding to these points; i.e. one cannot optimize the lines near the silhouettes without moving the silhouettes which is best done by techniques of the type described in [NSACO05]. This suggests that projection to the image plane leads to the natural parameterization for the problem, as in this case silhouette points will form boundaries for optimization regions.

We choose the coordinate system aligned with the image plane ($\mathbf{a}_\perp$, $\mathbf{a}_v$, $\mathbf{v}$), where $\mathbf{a}_v$ is the normalized projection of $\mathbf{a}$ to the plane perpendicular to $\mathbf{v}$ (Figure 2). The coordinates along the three axes are x,y,z: we use the standard convention for y to be perpendicular to the image, x to be horizontal, and z to be the view direction. For a vector $\mathbf{t}$, we denote $(\mathbf{t}\cdot\mathbf{a}_\perp) = t^x$, $(\mathbf{t}\cdot\mathbf{a}_v) = t^y$, and $(\mathbf{t}\cdot\mathbf{v}) = t^z$.

Using this notation we reduce the components of our expression to $r_1 = (\mathbf{r}\cdot\mathbf{a}_\perp) = 2n^z n^x$ and $r_2 = (\mathbf{r}\cdot\mathbf{v}_a) = -2n^z n^y \sin\alpha + (n^{z2} - n^{x2} - n^{y2})\cos\alpha$, where α is the angle between $\mathbf{v}$ and $\mathbf{a}$.

If we regard the surface as locally parametrized over the image plane, i.e. given by $z = f(x,y)$, we take the non-unit length normal to be $\mathbf{n} = (f_x, f_y, 1)$, where f_x and f_y are derivatives of f in two directions, and the expression for $\theta = \arctan(r_1, r_2)$ further simplifies to

$$\theta(x,y) = \arctan\left(2f_x, -2f_y\sin\alpha + (1 - f_x^2 - f_y^2)\cos\alpha\right). \tag{2}$$

2.2. Optimization problems

Given a user-defined reflection function our goal is to determine a surface which approximates this field as closely as possible. Mathematically, we can formulate the problem in several ways: exact match of the reflection function, minimization of the difference between the desired and actual reflection function, and minimization of the difference in line directions and density captured by the gradient of the reflection function (cf. [LGS99]). We briefly review these options here, with emphasis on the allowable boundary conditions.

One can observe that if $\theta(x,y)$ is given, Equation 2 is a first-order PDE which can be solved using the characteristic ODE system $\dot{x} = F_{p^x}$, $\dot{y} = F_{p^y}$, $\dot{p}^x = -F_x$, $\dot{p}^y = -F_y$, where $F(x,y,p^x,p^y) = (1 - f_x^2 - f_y^2)\tan\theta^*(x,y) - 2f_x$. With suitable assumptions on the left-hand side and boundary conditions, the solution exists. However, as the system is first-order, only initial value problems generally have solutions. In particular one cannot expect the problem to have solutions if the values are prescribed on the boundary of a patch. We also note that if instead of specifying θ we had specified the reflection vector $\mathbf{r}$, the resulting system of two PDEs would not necessarily have a common solution even with no boundary conditions.

If the boundary of a region is fixed, the best we can do is to minimize the difference in reflection functions. Instead of fitting the angle values $\theta(x,y)$, we avoid the problems with the discontinuity of θ values by using the functional

$$\int_S (\cos\theta - \cos\theta^*)^2 + (\sin\theta - \sin\theta^*)^2 dxdy, \tag{3}$$

where the integral is over the image plane projection of the region of interest, with projection assumed to be one-to-one.

The Euler-Lagrange equation for this problem is second-order; therefore one hopes to be able to solve the problem with Dirichlet data on the boundary, but not with Neumann data. This implies that one cannot expect the solution to blend smoothly with the rest of the surface if the optimization is performed only on a small area.

Finally, instead of fitting the function values one can fit the gradient of the reflection function to the gradient of the desired function:

$$\text{Minimize} \int_S (\nabla\theta - \nabla\theta^*)^2 dxdy,\ \theta|_S = \theta_0,\ \frac{\partial}{\partial n}\theta|_S = 0, \tag{4}$$

where $\partial/\partial n$ is the derivative along the boundary normal. In this case, the corresponding Euler-Lagrange equation is fourth-order, similar to the PDE for the thin-plate energy, and one can prescribe both Dirichlet and Neumann boundary conditions ensuring smooth transition between the optimized patch and the surface. In this energy, we need an expression for $\nabla\theta$. As $r_1 = 2f_x$ and $r_2 = -2f_y\sin\alpha + (1 - f_x^2 - f_y^2)\cos\alpha$,

$$\nabla\theta = \frac{r_2\nabla r_1 - r_1\nabla r_2}{r_1^2 + r_2^2}$$

All problems that we have considered assume that a reflection line function $\theta(x,y)$ is prescribed.

3. Discretization and numerical methods

We aim to design a discretization of problem 4 which balances accuracy, robustness and efficiency required by interactive applications.

A C^1 finite-element or arbitrary mesh C^1 spline discretization would be most straightforward but is relatively expensive. We use a more efficient and easier to implement alternative which combines a discrete geometric approach with finite differences. We use triangle-centered discretization stencils for both first and second-order derivatives which leads to simple discretization of Equation 4. While there is

no rigorous convergence guarantee by construction, we show excellent behavior for most mesh types.

Reduction to parametric case. As it was discussed in the previous section, our optimization problems can be solved in a functional setting by using surface parameterization over the image plane. However, while obtaining such a parameterization is computationally expensive for high-order surfaces (e.g. subdivision surfaces and splines); for meshes, the parametrization is easily obtained by a simple linear transformation: we rotate the coordinate system so that the image plane coincides with the (x,y) plane and the projection of the light direction $\mathbf{a}$ to the image plane is aligned with the y axis, i.e. use the coordinate system $(\mathbf{v},\mathbf{a}_v,\mathbf{a}_\perp)$ (Figure 2). As one cannot reliably control reflections near silhouette points, we fix all vertices close to silhouettes, i.e. fix vertices of all triangles with normals $\mathbf{n}$, for which $|\mathbf{n}\cdot\mathbf{v}| <$ (we use $= 0.02$ in all cases).

For surfaces with no silhouettes (e.g. nearly flat patches) additional boundary conditions are necessary: typically, we want the modified surface patch to join smoothly with the rest of the surface. After preprocessing, the mesh is decomposed into disjoint pieces, each of which is a piecewise linear height field over the image plane.

Our functionals depend on the components of the gradient and Hessian of f, which we discretize next. We use *triangle-centered* discretization, i.e. a single value of the gradient or Hessian is assigned to each face rather than vertex. This leads to simple formulas for the gradients and Hessians, and makes it possible to consider a minimal number of special cases. Each discretization associates a 2×6 and 3×6 matrix of coefficients G and H with each triangle. if $f_T = (f(\mathbf{p}_1), f(\mathbf{p}_2), f(\mathbf{p}_3), f(\mathbf{q}_1), f(\mathbf{q}_2), f(\mathbf{q}_3))$ (Figure 3) then Gf_T and Hf_T yield the gradient and Hessian respectively.

Discretizing gradients. To discretize the gradient $\nabla f = (f_x, f_y)$ over the image plane, we use standard piecewise-linear continuous finite elements.

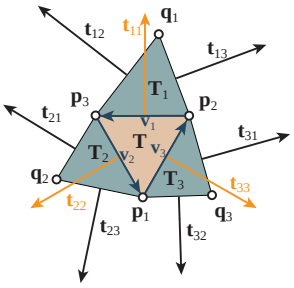

Figure 3: *Vectors used in the gradient and Hessian definitions; all points are in the image plane. The vectors $\mathbf{t}_{ij}$ are perpendicular to corresponding triangle sides and have the same length as these sides.*

We observe that the gradient can be found in the form $\sum_i c_i \mathbf{t}_{ii}$ where c_i are determined by $\sum_i c_i(\mathbf{t}_{ii}\cdot\mathbf{v}_j) = f_j - f_k$, and (i,j,k) is a cyclic permutation of $(1,2,3)$. This yields

$$\nabla_{discr} f_T = \frac{1}{2A}\sum_{i=1,2,3} f(\mathbf{p}_i)\mathbf{t}_{ii} \tag{5}$$

where $f(\mathbf{p}_i)$ denotes the value of f at vertex $\mathbf{p}_i$. The coefficients $\mathbf{t}_{ii}/2A$ do not change and need to be computed only when the surface is rotated.

Discretizing Hessians. Discretizing Hessians is considerably more difficult: while for gradients a piecewise linear approximation depending only on function values at triangle vertices is adequate, for second derivatives one needs to use more vertices, or introduce additional degrees of freedom. As the total number of derivatives of order ≤ 2 is six, one needs at least six degrees of freedom per stencil to capture local behavior correctly.

Most discretizations of second-order quantities (typically, curvature) used in geometric modeling are *vertex-centered*, which is inconvenient for our purposes. To be compatible with the gradient discretization, we use a triangle-centered stencil shown in Figure 3. †

For triangles without vertices of valence three, it has six degrees of freedom, exactly the number needed for discretizing the Hessian.

On this stencil, an approximation to the Hessian can be constructed in a number of different ways. We use a combination of two approaches.

Triangle-averaged discretization. Cohen-Steiner and Morvan [CSM03] describes a general approach to computing shape operators for meshes by averaging elementary shape operators corresponding to edges. While convergence of this technique was only established in the integral sense, and for a restricted class of meshes, simplified versions of this approach were shown to work well in practice. The most common example is the well-known cotangent formula [PP93], which (for small deformations) is equivalent to expressions of [CSM03] summed over a single ring of edges around a vertex [HPW05]. Similarly, the triangle-averaged discretization on the stencil of Figure 3 introduced in [GSH*04] uses averaging over three edges of a triangle. By linearizing this formula, we obtain the following expression for the Hessian:

$$\frac{1}{A}\left(\sum_{i,j,j=i}\frac{1}{A_j} f(\mathbf{q}_j)\mathbf{t}_{ii}\otimes\mathbf{t}_{ij} + \sum_i \frac{1}{A_i} f(\mathbf{p}_i)\mathbf{t}_{ii}\otimes\mathbf{t}_{ii}\right) \tag{6}$$

† Another possible option is to use a single triangle and add edge-based degrees of freedom as it was done in [GGRZ06]. We have experimented with a linearized version of this discretization. In contrast to the general curvature discretization (3 coordinates per vertex), addition of edge degrees of freedom in our setting (one degree of freedom per vertex) adds a significant computational cost. Furthermore, stability of the nonlinear solve is decreased which further decreases performance.

where $\mathbf{t}_{ij}$ are side perpendiculars to the triangles of the mesh projected to the image plane, shown in Figure 3. A distinctive feature of this discretization is its robustness and simplicity: only for triangles with very small area may the coefficients in the formula be large.

This is a consistent (converging to the correct values) discretization of the Hessian for special types of meshes. Specifically, the Hessian is consistent for meshes in which vertices $\mathbf{q}_i$ are reflections of $\mathbf{p}_i$ with respect to the centers of opposite edges $i = 1,2,3$. This includes regular meshes and any affine transformations of regular meshes.

For general meshes, the discretization introduces mesh-dependent error in Hessian approximation, as shown in Figure 5. For different types of meshes, the results are mesh-dependent, no matter how fine the mesh is. Importantly for our application, the errors are low-frequency while high-frequency errors have the most effect on the visual quality of results.

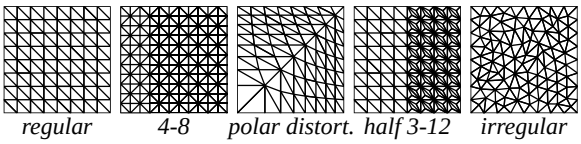

Figure 4: *Mesh types used in convergence experiments.*

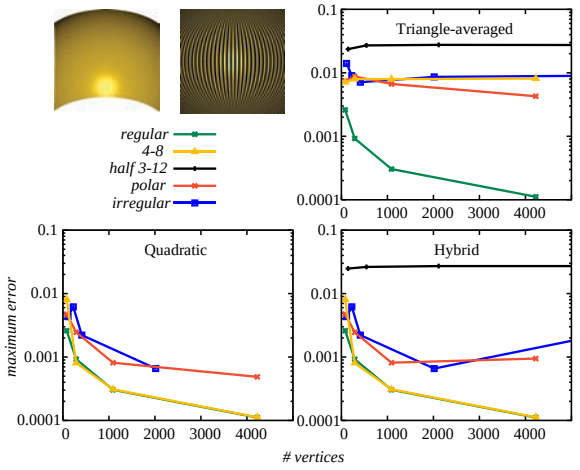

Figure 5: *Convergence experiments: A spherical surface patch was recovered from an analytically computed reflection function gradient for different mesh connectivities and resolutions. Three discretization types are shown: triangle-averaged, quadratic fit and hybrid. Quadratic interpolation cannot be applied to meshes with vertices of valence 3; optimization also fails on higher resolution irrregular meshes because it contains stencils with all vertices close to a conic. The error is measured relative to the size of the object along the view direction.*

Quadratic interpolation discretization. An alternative is to use a finite-difference approach to discretization: we compute a quadratic function Q satisfying $Q(\mathbf{p}_i) = f(\mathbf{p}_i)$, $Q(\mathbf{q}_i) = f(\mathbf{q}_i)$, $i = 1\ldots3$, and use its quadratic term coefficients to estimate the Hessian. The advantage of this approach is that by construction it is consistent whenever the quadratic function is defined. This is not sufficient for convergence of the discrete problem solutions to the continuous solution, (see [GGRZ06]) but improves independence of the result from mesh connectivity. Unfortunately, the approach is significantly less robust and the following proposition holds:

For six points $\mathbf{w}_i \in \mathbf{R}^2$, $i = 1\ldots6$, there is a unique quadratic function satisfying $Q(\mathbf{w}_i) = z_i$, for arbitrary choice of z_i, if and only if these six points are not on the same conic (see Appendix.)

Whenever six points of the stencil are *close* to a common conic, the coefficients of the quadratic interpolant become large and Hessian estimation becomes highly unreliable.

Hybrid discretization. As it can be seen from the plots above, quadratic interpolation yields good estimates in most cases, but it is not robust. In practice, we observe that we have several triangles per mesh for which the six-point stencil is close to a conic, and quadratic interpolation produces low-quality results. To solve this problem, we combine two techniques: the triangle-averaged scheme is used when the quadratic interpolation is unstable, i.e. if the stencil contains five points or less, or vertices are close to a conic. We evaluate stability for a specific six-point stencil by comparing the magnitude of the discrete Hessian coefficients to $1/l_{max}^2$, where l_{max} is the maximal edge length in the stencil. If any coefficient exceeds C/l_{max}^2 (we use $C = 5$), we use triangle-averaged discretization instead of the quadratic interpolation. As it can be seen from the convergence plots, the resulting scheme retains the accuracy of the quadratic fit and yet does not suffer from its robustness problems, although it produces large errors for meshes with many degenerate cases. Such meshes appear to be unusual. Our stability criterion is motivated by the observation that if the function value is 0 at all but one vertex of a stencil, and is of the order l_{max}^2 (i.e. squared distance to other vertices) at that one vertex, one can expect to get second derivative magnitudes on the order of 1 (for a mesh close to regular). Coefficients much larger than $1/l_{max}^2$ lead to instability.

While numerically the discretization is more accurate, we note that we have observed few differences in visual quality when using the triangle-averaged discretization alone. Finally, while we found this discretization adequate for the functionals considered in this paper, its performance for thin-plate or Willmore energy is not as good ([JR07]).

Discretizing normals. An essential part of an interactive system supporting reflection manipulation is rendering of reflection lines and environment maps. Hardware environment maps use vertex normals to compute reflected directions and look up values in the environment texture. As Figure 6 shows, standard vertex normal computation techniques

produce low-quality results for complex meshes. Instead, we use a local fit to obtain better normals. For every vertex $\mathbf{p}$, we collect a ring N_1 of triangles around it, and all triangles edge-adjacent to N_1. The minimal number of vertices in such a configuration is six, unless the whole mesh has a smaller number of vertices. We compute an initial normal $\mathbf{n}_{init}$, and project vertices to the plane perpendicular to $\mathbf{n}_{init}$.

Let w be the vector of projected point positions of length K, and let $f(w)$ be the vector of function values at these points. For each vertex, we precompute a $2 \times K$ matrix of coefficients C_{norm} mapping the vector $f(w)$ to the *linear* coefficients of a best fit quadratic function in the coordinate system with origin at vertex $\mathbf{p}$, with z axis aligned with the initial normal (when it is not defined uniquely, we choose the quadratic function with minimal norm of coefficients; the coefficients are computed using LAPACK function GESDD). These coefficients define a plane passing through the vertex, and we use the normal to this plane as our final normal. As long as we do not change the plane we have used for local normal estimation, the normal approximation can be recomputed rapidly as the surface is modified. If the motion is large, the quality deteriorates, and $\mathbf{n}_{init}$ needs to be recomputed.

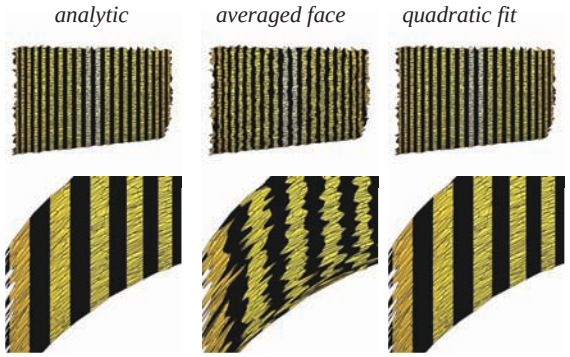

Figure 6: *Comparison of the vertex normal quality; the surface is obtained by sampling points from a cylinder. While face averages do not perform well for this mesh, our quadratic fit procedure yields results visually indistinguishable from using analytic normals.*

3.1. Numerical implementation

The reflection-based functionals are more complex than most expressions commonly used for surface optimization, therefore computing Hessians and gradients are also more expensive. The energy cannot be replaced by a linearized functional, because then it would not capture the shape of reflection lines in most cases. Either a full non-linear Newton solve for the energy minimum or a gradient-only method would be prohibitively expensive, the former due to Hessian computation, and the latter because of the large number of iterations required.

To improve performance we use an inexact Newton method with line search. Instead of a full Hessian computation, we compute the Hessian once for the *linearized problem*, and use it instead of the full Hessian optimization at all iterations.

If we assume small values of f_x and f_y a simple calculation shows that the equation for the reflection function reduces to

$$\theta_{lin}(x,y) = 2 f_x \cos\alpha.$$

The gradient of θ_{lin} remains simple, and is, up to a constant, $[f_{xx}, f_{xy}]$. For the quadratic energy based on θ_{lin}, the Hessian is easy to compute and does not depend on function values.

As the gradient problem is fourth-order and the condition number of the Hessian matrices grows as N^4, iterative solvers are not efficient; instead, we use a direct solver (PARDISO). The direct solver performs a sparse LU factorization of the matrix, and solves the system using backsubstitution. As the Hessian matrix does not change, the matrix factorization needs to be performed only once. For each nonlinear iteration, only gradients need to be recomputed. As shown in Figure 7, while using an approximate Hessian increases the number of nonlinear iterations required, each iteration becomes much faster, and there is a considerable net win in performance. This performance improvement depends on the complexity of the target reflection function gradient $\nabla\theta^*$ and varies in the range $2\times$ to $10\times$. When the target function is smooth, which is the most common case (forward optimization in Figure 7), the speedups are higher, while for less smooth targets speedups are lower.

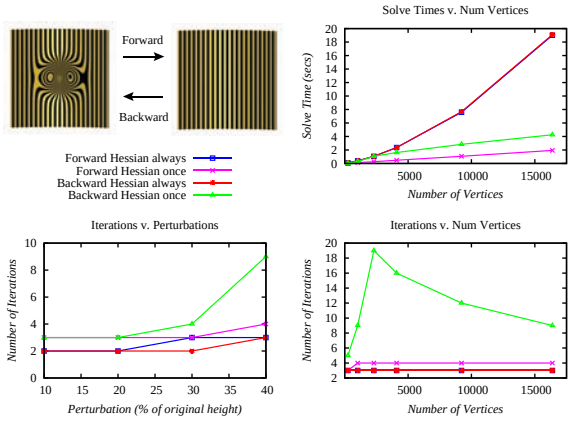

Figure 7: *Speedups from approximate Hessian computation, dependence on the mesh size. Two model problems: creation and elimination of a bump on a cylinder. Times are given for a Pentium D 3GHz processor.*

4. Reflection line manipulation experiments

Different types of reflection manipulation using discrete reflection line functionals follow the same general pattern. First, the user selects an area to modify and specifies the boundary conditions. Any boundary segment may be free, fixed, or clamped, the latter means that two layers of vertices are fixed at the boundary to ensure a smooth match of

the surface with the rest of the mesh. Then the target reflection function gradient is defined using user input and the surface minimizing the reflection functional is computed. The last two steps may be repeated in the interaction loop.

Density and direction change. The simplest type of reflection line manipulation is requesting a fixed line direction and density in an area, i.e. specifying a fixed target $\nabla_{discr}\theta^*$ everywhere. The energy minimization in this case attempts to modify reflection lines as requested, while maintaining a smooth or continuous join of the selected patch with the rest of the surface. The user can also adjust a fall-off curve $c(t) \geq 1$ which determines how gradual the transition between the modified area and the rest of the mesh should be. The energy of each triangle is scaled by $c(d)$ where d is the distance to the center of influence.

Examples of this type of manipulation are shown in Figure 9, and Figure 11. In Figure 9, one can see how adjusting reflection line density makes it possible to control the appearance of reflections, in particular the sizes of reflected objects. Rotating the desired reflection line direction at a point is shown in Figure 9 on the right. In this example, a surface imperfection creates a "twist" in the reflection line field which can be removed by local rotation. Figure 11, demonstrates the difference between conventional smoothing and reflection line optimization: typically, smoothing algorithms flatten the surface which does not necessarily improve the reflection line shape.

Smoothing. Isotropic or directional smoothing can be used to transform initial $\nabla\theta$ values to the target values. This operation is similar to reflection line smoothing described in [LGS99]. Directional smoothing is particularly useful, as it straightens reflection lines without changing the behavior in the orthogonal direction. An example is shown in Figure 12, left. In this example, the initial reflection function was smoothed using Laplacian smoothing in the image domain, and then used as the target reflection function.

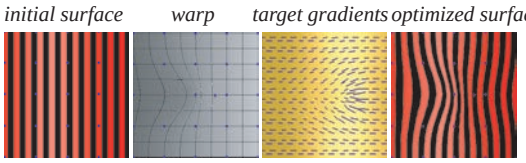

Figure 8: *Warping stages: initial reflection lines, warped reflection function θ, target $\nabla\theta^*$ computed per triangle, reflection lines on the optimized surface.*

Warping. In the case of warping, the goal is to apply an arbitrary user-specified transformation to the reflection lines. In our current implementation, the transformation of reflection lines is specified by a two-dimensional spline, however any other image warping technique can be used. The general formula for the transformed reflection function is

$$\theta_{warp}(s,t) = \theta_{init}(w^{-1}(s,t))$$

where $w(s,t) : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ is the warping function.

Observe that the inverse of the warping function needs to be computed, which can be relatively expensive for smooth deformations. To avoid explicit inversion of $w(s,t)$, we implement the warp using texture mapping and image-domain operations. Rather than attempting to transform $\nabla_{discr}\theta$ values needed by the functional directly, we transform reflection function values, and then compute the gradient. First, the reflection function θ values are computed, interpolated to vertices, and used as color values to render the mesh to a texture θ_{init}. A two-dimensional spline $w(s,t) : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ is created. Initially, the control points are equispaced so w is an identity. As the user moves control points, the spline is rendered to a new texture θ_{warp}, using θ_{init} as a texture map. For image-space mesh vertex positions we sample $\Delta\theta = \theta_{warp} - \theta_{init}$, and compute its gradient for each triangle using the gradient formula 5, with appropriate corrections applied to values to eliminate the jump between $-\pi$ and π. We add resulting $\nabla_{discr}\Delta\theta$ values to the initial $\nabla_{discr}\theta$ values. Note that by construction $\Delta\theta = 0$ if w is the identity function. An example of warping is shown in Figure 10; where the goal is to improve the shape of reflections at a part of a car hood.

Finally, one can use our technique to create surfaces approximating an arbitrary reflection line pattern, as shown in Figure 12, right. Any grayscale image can be used to specify $\nabla\theta^*$, as long as it is sufficiently smooth (otherwise, the approximation is likely to be poor).

5. Conclusions

We have described a simple and efficient approach to discretizing reflection line based functionals on meshes and demonstrated how these functionals can be used in an interactive system to optimize the shape of reflective surfaces.

One limitation of the proposed approach (which is also responsible for its comparatively high efficiency) is that the vertices of the mesh move only in the direction perpendicular to the image plane. This means that small scale surface details which make the projection to the image plane not one-to-one cannot be removed, and creates a disturbance in the surface during optimization. Although it can be applied to large perturbations, the technique is best suited for smaller adjustments of surfaces that are already relatively smooth. While we tried to eliminate obvious inefficiencies in our implementation, our code is far from optimal.

Our discretization can be easily combined with other surface optimization techniques, to be applied simultaneously or as a postprocess.

Acknowledgements. The authors would like to thank Robb Bifano, Jeff Han, and Harper Langston for their help with the paper and video production. This work was partially supported by the award NSF CCR-0093390, IBM Faculty Parternship award and a Rudin Foundation fellowship.

Appendix: Quadratic interpolation on a six-point stencil

The quadratic interpolation problem for a six-point stencil requires solving a linear system, which may be singu-

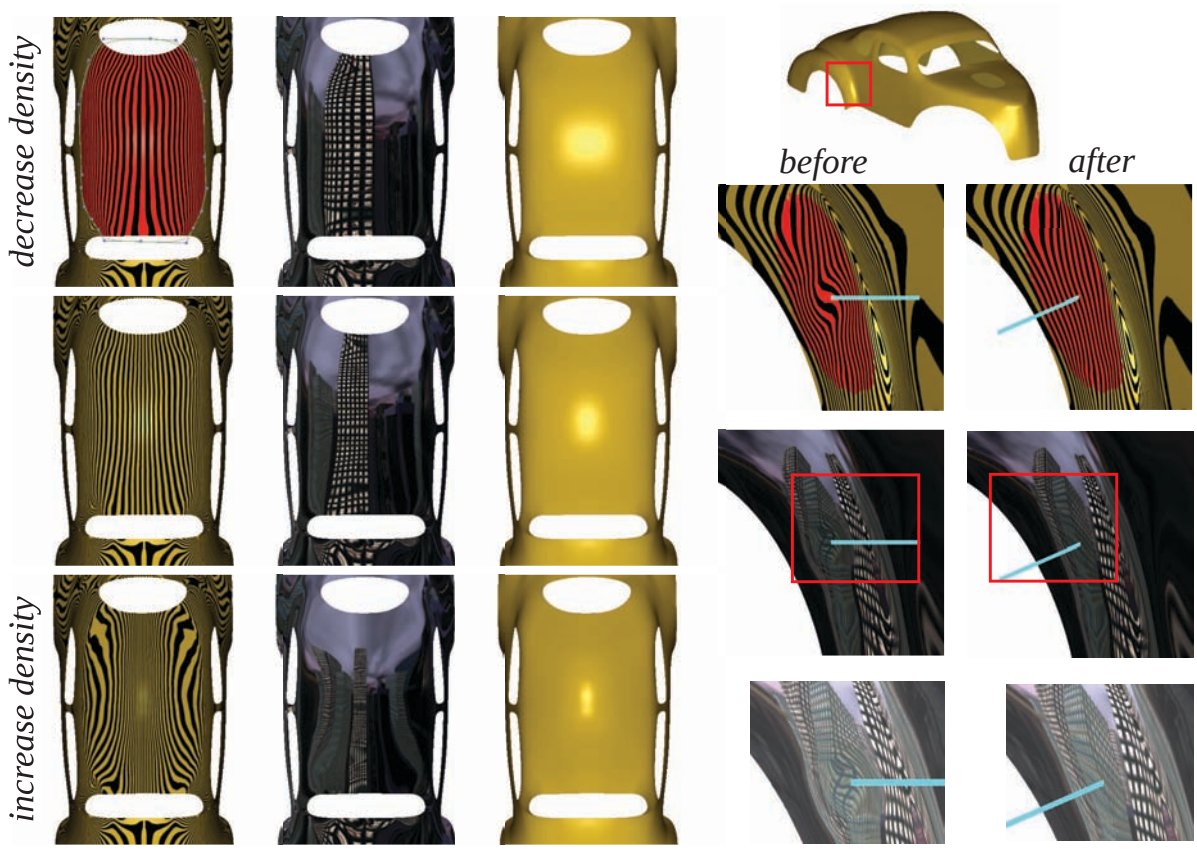

Figure 9: *Left: Changing reflection line density. A fixed reflection line direction is specified with density decreased at the top and increased at the bottom. Right: reflection line untwisting. The reflection function gradient is rotated to get desired appearance.*

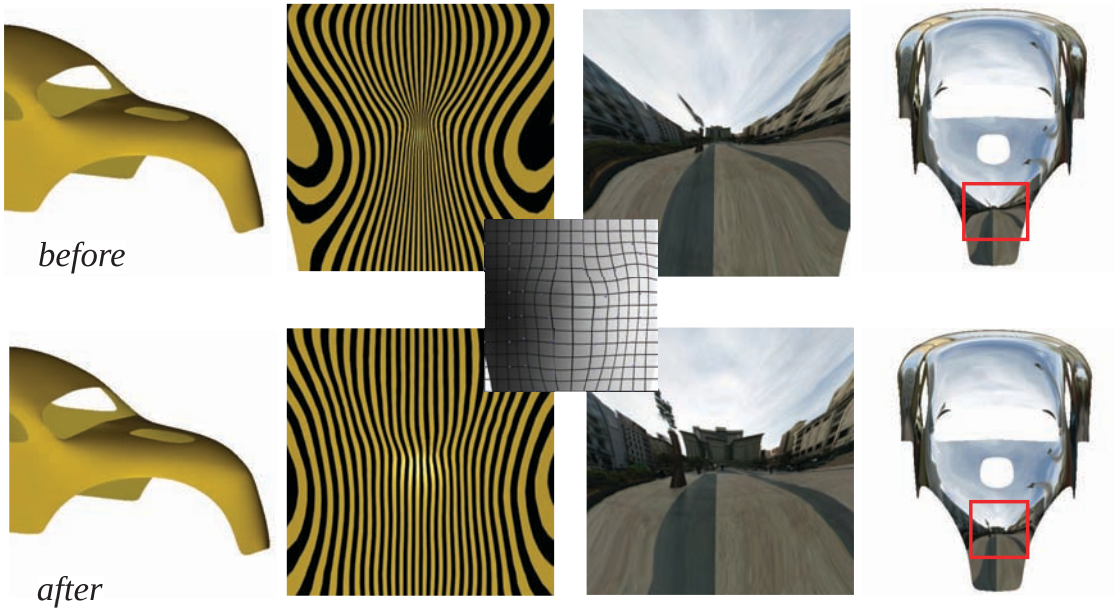

Figure 10: *Reflection line warping on a car hood. An intermediate warp is shown in the middle. Note that the change in the shape is barely perceptible but the change in the reflection is substantial.*

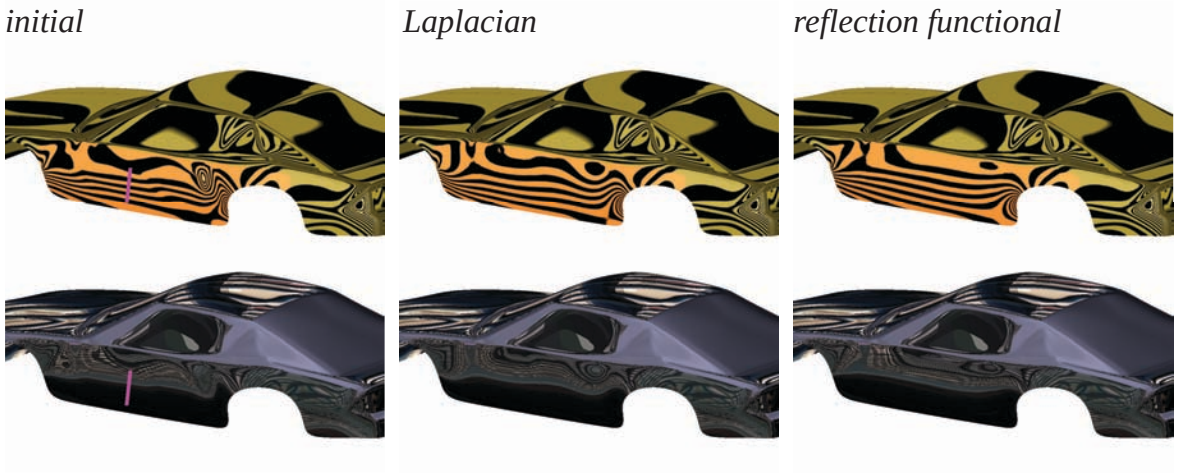

Figure 11: *Prescribing fixed reflection line direction on a car, compared to Laplacian smoothing. Note that Laplacian smoothing retains reflection line wiggles.*

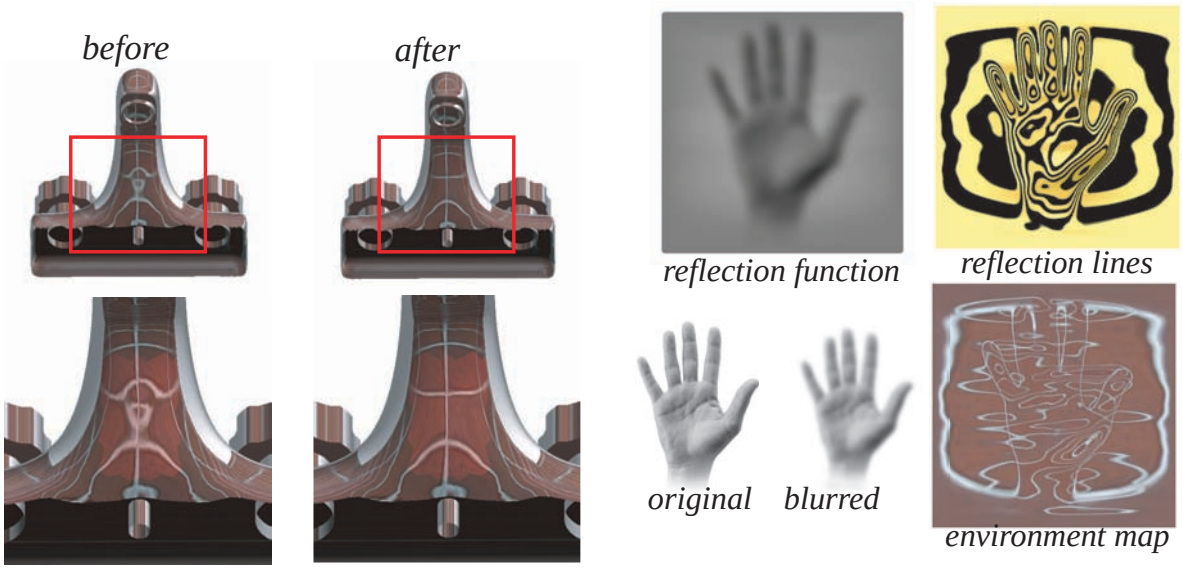

Figure 12: *Left: Reflection line smoothing on a faucet; the initial reflection line function is smoothed and used as the target reflection line function. Right: Reconstructing a surface with predefined reflection line pattern based on an blurred image.*

lar. A simple geometric condition for testing singularities of point configurations can be derived from geometric considerations, along with an explicit expression for the quadratic interpolant, which simplifies implementation.

We denote the six points of the stencil in the image plane by $\mathbf{w}_i$, $i = 1 \dots 6$. We need to find a quadratic function Q, such that $Q(\mathbf{w}_i) = z_i$, $i = 1 \dots 6$. Assume that all points are given in homogeneous coordinates. Then we can define Q as a 3×3 matrix: $Q(\mathbf{w}_i) = \mathbf{w}_i^T Q \mathbf{w}_i$. We observe that by linearity of the problem it is sufficient to solve it for $z_i = \delta_{ij}$, for $j = 1 \dots 6$, to obtain six basis matrices Q_j. Assume $j = 1$, i.e. $z_i = 0$ for $i > 1$. It well known that any five points in the plane are on a conic. Note that the problem of finding Q_1 is equivalent to the problem of finding such conic: Q_1 vanishes at all points $\mathbf{w}_i$, $i > 1$, i.e. defines a conic passing through these points, and conversely given a nontrivial conic with matrix M passing through $\mathbf{w}_i$, $i > 1$, we obtain Q_1 as $M/(\mathbf{w}_1^T M \mathbf{w}_1)$. If $\mathbf{w}_1^T M \mathbf{w}_1 = 0$, either there are multiple conics passing through the points, or the system has no solution; in either case, the system for Q_i is singular.

The conic matrix M can be computed using a matrix form of the Braikenridge-Maclaurin construction, [CG67]. Specifically, define $\mathbf{l}_{i,i+1} = \mathbf{w}_i \times \mathbf{w}_{i+1}$ (3d cross product applied to the homogeneous point representation). Let $R(\mathbf{a})$ be the skew symmetric matrix satisfying $R\mathbf{x} = \mathbf{a} \times \mathbf{x}$. Then M_1 is given by $R(\mathbf{w}_1)R(\mathbf{l}_{34})R(\mathbf{r})R(\mathbf{l}_{23})R(\mathbf{w}_5)$, where $\mathbf{r} = \mathbf{l}_{12} \times \mathbf{l}_{45}$ and Q_i are obtained by cyclically permuting $\mathbf{w}_i$ and applying the same formula.

References

[BK04] BOTSCH M., KOBBELT L.: An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics 23*, 3 (2004), 630–634.

[BS07] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* (2007). To appear.

[CF98] CAIMING Z., FUHUA C.: Removing local irregularities of nurbs surfaces by modifying highlight lines. *Computer Aided Design 30*, 12 (1998), 923–930.

[CG67] COXETER H. S. M., GREITZER S. L.: *Geometry Revisited.* Math. Assoc. Amer., 1967.

[CG91] CELNIKER G., GOSSARD D.: Deformable curve and surface finite-elements for free-form shape design. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1991), ACM Press, pp. 257–266.

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycles. *Proc. 19th Annu. ACM Sympos. Comput. Geom.* (2003), 237–246.

[DB04] DESLANDES A., BONNER D. L.: Reflection line control. U.S. Patent 6717579, 2004.

[GGRZ06] GRINSPUN E., GINGOLD Y., REISMAN J., ZORIN D.: Computing discrete shape operators on general meshes. *Eurographics (Computer Graphics Forum) 25*, 3 (2006).

[GLW96] GREINER G., LOOS J., WESSELINK W.: Surface modeling with data dependent energy functionals. *Comput. Graph. Forum 15* (1996), 175–186.

[GOZ95] GUID N., OBLONSEK C., ZALIK B.: Surface interrogation methods. *Computers & Graphics 19*, 4 (1995), 557–574.

[GSH*04] GINGOLD Y., SECORD A., HAN J. Y., GRINSPUN E., ZORIN D.: *A Discrete Model for Inelastic Deformation of Thin Shells*. Tech. rep., Aug 2004.

[HBKM96] HALSTEAD M. A., BARSKY B. A., KLEIN S. A., MANDELL R. B.: Reconstructing curved surfaces from specular reflection patterns using spline surface fitting of normals. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 335–342.

[HHS*92] HAGEN H., HAHMANN S., SCHREIBER T., NAKAJIMA Y., WORDENWEBER B., HOLLEMANN-GRUNDSTEDT P.: Surface interrogation algorithms. *IEEE Computer Graphics and Applications 12*, 5 (Sept. 1992), 53–60.

[Hor86] HORN B. K. P.: *Robot Vision*. The MIT Press, 1986.

[HPW05] HILDEBANDT K., POLTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Eurographics Symposium on Geometry Processing: SGP'05* (2005), pp. 85–90.

[JR07] J. REISMAN E. GRINSPUN D. Z.: *A note on the triangle-centered quadratic interpolation discretization of the shape operator*. Tech. rep., New York University, 2007.

[KK88] KAUFMANN E., KLASS R.: Smoothing surfaces using reflection lines for families of splines. *Computer Aided Design 20*, 6 (1988), 312–316.

[Kla80] KLASS R.: Correction of local surface irregularities using reflection lines. *Computer-Aided Design 12*, 2 (March 1980), 73–77.

[Kob00] KOBBELT L.: Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer 16*, 3-4 (2000), 142–158.

[LGS99] LOOS J., GREINER G., SEIDEL H.-P.: Modeling of surfaces with fair reflection line pattern. In *Shape Modeling International* (1999), pp. 256–.

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.). Springer-Verlag, Heidelberg, 2003, pp. 35–57.

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. *International Conference on Computer Graphics and Interactive Techniques* (2005), 1142–1147.

[Poe84] POESCHL T.: Detecting surface irregularities using isophotes. *Computer-Aided Geometric Design 1*, 2 (1984), 163–168.

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experiment. Math. 2*, 1 (1993), 15–36. 1058-6458.

[SK01] SCHNEIDER R., KOBBELT L.: Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design 18*, 4 (2001), 359–379.

[SLCO*04] SORKINE O., LIPMAN Y., COHEN-OR D., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), ACM Press, pp. 179–188.

[TF97] THEISEL H., FARIN G.: The curvature of characteristic curves on surfaces. *IEEE Comput. Graph. Appl. 17*, 6 (1997), 88–96.

[The01] THEISEL H.: Are isophotes and reflection lines the same? *Computer Aided Geometric Design 18*, 7 (2001), 711–722.

[WW94] WELCH W., WITKIN A.: Free-form shape design using triangulated surfaces. *Computer Graphics 28*, Annual Conference Series (1994), 247–256.

[YBP97] YIFAN C., BEIER K. P., PAPAGEORGIOU D.: Direct highlight line modification on nurbs surfaces. *Computer-Aided Geometric Design 14*, 6 (1997), 583–601.

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 644–651.