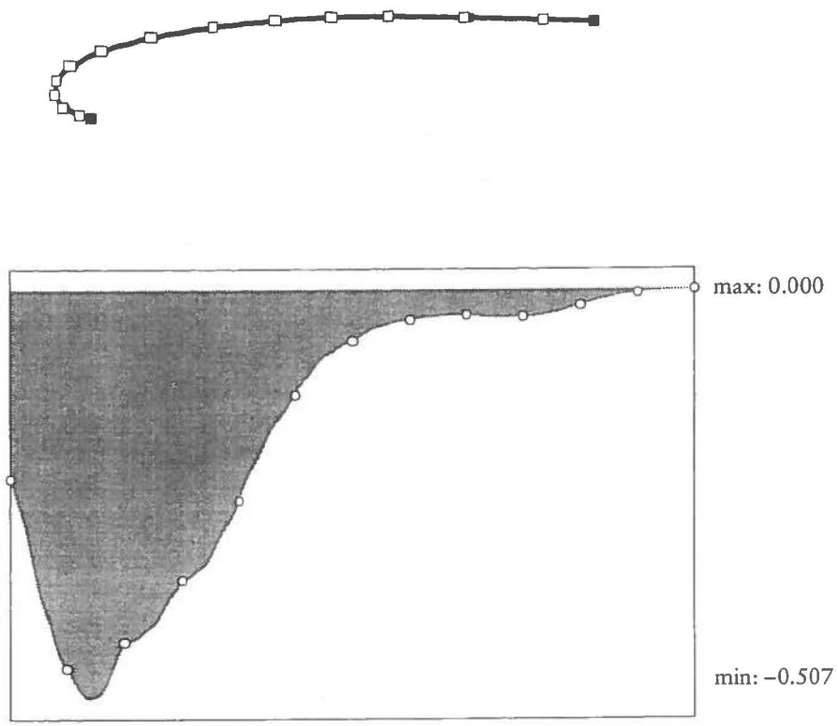

Figure 23.4 Curve smoothing: the smoothed curve and its curvature plot.

without the use of curvature plots. The faired curve does not have this shape defect any more.

In practice, the improved vertex $\hat{\mathbf{d}}_j$ may be farther away from the original vertex $\mathbf{d}_j$ than a prescribed tolerance allows. In that case, we restrict a realistic $\hat{\mathbf{d}}_j$ to be in the direction toward the optimal $\hat{\mathbf{d}}_j$, but within tolerance to the old $\mathbf{d}_j$.

Other methods for curve fairing exist. We mention Kjellander's method [358], which moves a data point to a more favorable location and then interpolates the changed data set with a C^2 cubic spline. This method is global. A method that fairs only data points, not spline curves, is presented by Renz [507]. This method computes second divided differences, smoothes them, and "integrates" back up. Methods that aim at the smoothing of single Bézier curves are discussed by Hoschek [333], [335]. Variations on the described method are given in [215].

A method that tries to reduce the degrees of each cubic segment to quadratic is given in [200].

23.3 Surface Interrogation

Curvature plots are useful for curves; it is reasonable, therefore, to investigate the analogous concepts for surfaces. Several authors have done this, including Beck et al. [49], Farouki [219], Dill [169], Munchmeyer [435], [434], and Forrest [242]. An interesting early example is on page 197 of Hilbert and Cohn-Vossen [323]. Surfaces have two major kinds of curvature: Gaussian and mean; see Section 19.6. Both kinds can be used for the detection of surface imperfections. Another type of curvature can be useful, too: *absolute curvature* κ_{abs}. It is defined by

$$\kappa_{\text{abs}} = |\kappa_1| + |\kappa_2|,$$

where κ_1 and κ_2 are the maximal and minimal normal curvatures at the point under consideration.

Gaussian curvature does not offer much information about generalized cylinders of the form

$$\mathbf{c}(u, v) = (1 - u)\mathbf{x}(v) + u[\mathbf{x}(v) + \mathbf{v}].$$

Even if the generating curve $\mathbf{x}(v)$ is highly curved, we still have $K \equiv 0$ for these surfaces. A similar statement can be made about the mean curvature H, which is always zero for minimal surfaces, no matter how complicated.

Color Plates IV and V illustrate the use of curvatures in nonengineering applications. Plate IV shows the digitized model of a bone (digitized as a mesh and locally fitted with Bézier patches) and a color coding of the absolute curvature: where the curvature is high, the surface is "painted" red, and where it is low, it is "painted" blue. This process is referred to as *texture mapping* in computer graphics.

Color Plate V shows an application in archeology: a digitized vessel (Native American) is represented as a triangle mesh (left, with original texture). A cross section is computed, fitted with a B-spline curve, and its curvature is displayed (middle). Finally, Gaussian curvature is used as a texture map (right).

Another method for surface interrogation is the use of *reflection lines*, first described in Klass [360]. Poeschl [486] introduced a simplified method, *isophotes*. Reflection lines are a standard surface interrogation tool in the styling shop of a car manufacturer. They are the pattern that is formed on the polished car surface by the mirror images of a number of parallel fluorescent strip lights. If the

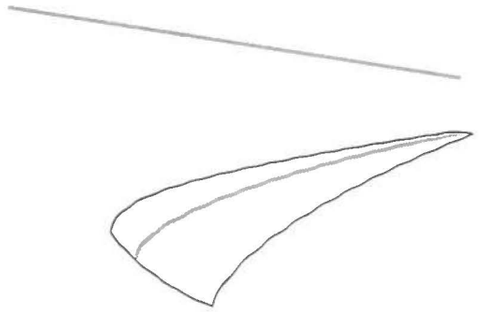

Figure 23.5 Isophotes: a line light source (top) is reflected by a surface.

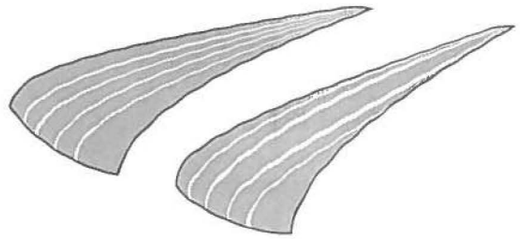

Figure 23.6 Isophotes: left, a surface with "perfect" isophotes; right, after a perturbation was applied to the surface.

mirror images are "nice," then the corresponding surface is deemed acceptable. Whereas reflection lines depend on the position of an observer, isophotes consider only the angle formed between surface normals and light source. The principle is illustrated in Figure 23.5.

Reflection lines and isophotes can easily be simulated on a raster graphics device (mark points whose normal points to one of the light sources). With some more effort, they can also be computed on a line drawing device (see Klass [360]).

Figure 23.6 shows a surface with several isophotes and the effect that a small perturbation can have on them.

Reflection lines and curvature "paintings" have different usages: reflection lines are not as fine-tuned as curvatures; they are prone to miss local shape defects of a surface.[5] On the other hand, curvatures of a surface may look perfect, yet it might not have a "pleasant" overall shape—reflection lines have a better chance of flagging global imperfections.

5 This is because reflection lines may be viewed as a first-order interrogation tool (involving only first derivatives), whereas curvature plots are second-order interrogation tools.

Once imperfections are detected in a tensor product B-spline surface, we would want methods to remove them without time-consuming interactive adjustment of control polygons. One such method is to apply the curve-smoothing method from Section 23.2 in a tensor product way: smooth all control net rows, then all control net columns. The resulting surface is usually smoother than the original surface. Figure 23.6 is an example of this method: the right figure is a B-spline surface; four iterations of the program `fair_surf` produced the figure on the left.

More involved methods for surface fairing exist; they aim for the enforcement of convexity constraints in tensor product spline surfaces. We mention Andersson et al. [8], Jones [348], and Kaufmann and Klass [354].

23.4 Implementation

The routine `curvatures` may be used to generate curvature values of a rational Bézier curve. It writes the values into a file that might be read by another program that generates a curvature plot.

To compute the curvature at the parameter value t, the curve is subdivided using the (rational) de Casteljau algorithm. Of the two subpolygons that are generated, the larger one is selected, and its beginning curvature is computed. Since the subdivision routine `rat_subdiv` orders both subpolygons beginning at the subdivision point, only one curvature routine `curvature_0` is needed.

```
void curvatures(coeffx,coeffy,weight,degree,dense)
/*     writes  signed curvatures of  a rational  Bezier curve into
       a file.
input:
     coeffx, coeffy:  2D Bezier polygon
     weight:          the weights
     degree:          the degree
     dense:           how many curvature values to compute
output:
                      written into file outfile
*/
```

The routine `curvature_0` is a simple application of (10.10):

```
float curvature_0(bez_x,bez_y,weight,degree)
/* computes curvature of rational  Bezier curve at t=0
   Input: bez_x, bez_y, weight: control polygon and weights
          degree:               degree of curve
*/
```

Plate IV.
A digitized bone (left) and absolute curvature color coding (right). Figure courtesy of PRISM at Arizona State University.

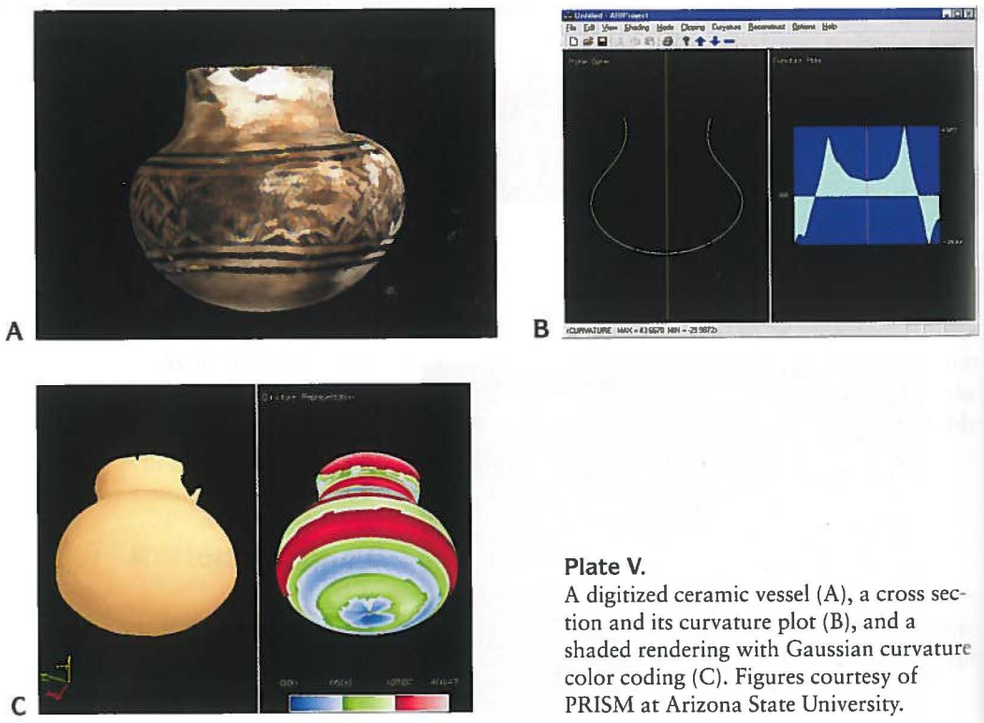

Plate V.
A digitized ceramic vessel (A), a cross section and its curvature plot (B), and a shaded rendering with Gaussian curvature color coding (C). Figures courtesy of PRISM at Arizona State University.