Suggestive Contours for Conveying Shape

Doug DeCarlo[1] Adam Finkelstein[2] Szymon Rusinkiewicz[2] Anthony Santella[1]

[1]Department of Computer Science & Center for Cognitive Science
Rutgers University

[2]Department of Computer Science
Princeton University

Abstract

In this paper, we describe a non-photorealistic rendering system that conveys shape using lines. We go beyond contours and creases by developing a new type of line to draw: the *suggestive contour*. Suggestive contours are lines drawn on clearly visible parts of the surface, where a true contour would first appear with a minimal change in viewpoint. We provide two methods for calculating suggestive contours, including an algorithm that finds the zero crossings of the radial curvature. We show that suggestive contours can be drawn consistently with true contours, because they anticipate and extend them. We present a variety of results, arguing that these images convey shape more effectively than contour alone.

Keywords: non-photorealistic rendering, contours, silhouettes

1 Introduction

Our interpretation of natural imagery draws upon a wealth of visual cues, including contours,[1] texture, shading, shadow, and many others. Since each individual cue can be noisy, ambiguous or even misleading, our visual system integrates all the information it receives to obtain a consistent explanation of the scene.

When artists design imagery to portray a scene, they do not just render visual cues veridically. Instead, they select which visual cues to portray and adapt the information each cue carries. Such imagery departs dramatically from natural scenes, but nevertheless conveys visual information effectively, because viewers' perceptual inferences still work flexibly to arrive at a consistent interpretation.

In this paper, we suggest that lines in line-drawings can convey information about shape in this indirect way, and develop tools for realizing such lines automatically in non-photorealistic rendering (NPR). We start from the observation that many lines in natural and artistic imagery come from *contours*, where a surface turns away from the viewer and becomes invisible. As in the rendering at left in Figure 1, contours can be quite limited in the information they convey about shape on their own. But the visual system is readily capable of relaxing the natural interpretation of lines as contours; instead, it can read lines merely as features where a surface bends sharply away from the viewer, yet remains visible—as features that are almost contours, that become contours in nearby views. We call these *suggestive contours*. When we draw suggestive contours alongside contours, as in the rendering at right in Figure 1, we exaggerate the shape information provided by contours to make a sparse line-drawing easier to understand. Figure 1 suggests how the two kinds of lines together convey an object's shape consistently and precisely.

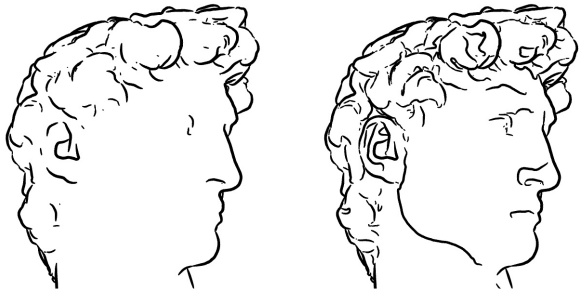


Figure 1: An example showing the expressiveness added by suggestive contours. The left image is drawn using contours alone, while the right image uses both contours and suggestive contours.

In this paper, we describe new NPR techniques based on suggestive contours. Our system produces still frames, such as those in Figure 1, which combine contours with a selection of those suggestive contours that are stable, given small perturbations of the viewpoint or surface. In introducing, characterizing and implementing suggestive contours, we make the following contributions:

- We offer several intuitive characterizations of lines that can augment true contours to help convey shape.
- We provide mathematical definitions corresponding to each of these intuitive characterizations, and show that these definitions are equivalent.
- We describe the mathematical relationship between suggestive contours and contours, showing that suggestive contours, despite being drawn on clearly visible parts of the surface, integrate with true contours in a seamless and consistent way.
- We provide two algorithms (one in object space and one in image space) for finding and rendering suggestive contours.
- We show imagery created by our implementation, demonstrating that suggestive contours complement contours in conveying shape effectively.

1.1 Related work

Lines are the scaffold of non-photorealistic rendering of 3D shape; and contours, which offer perhaps the strongest shape cue for smooth objects [Koenderink 1984], make great lines [Gooch et al. 1999; Hertzmann and Zorin 2000; Kalnins et al. 2002; Markosian et al. 1997; Raskar 2001; Winkenbach and Salesin 1994]. In many cases, however, contours alone cannot convey salient and important aspects of a rendered object, and additional lines are needed.

Before this work, all other lines drawn from general 3D shapes have been features fixed on the surface. Creases are the most frequent example [Kalnins et al. 2002; Markosian et al. 1997; Raskar 2001; Saito and Takahashi 1990; Winkenbach and Salesin 1994]; this can yield a pronounced improvement, but only when creases are conspicuous features of an object's shape (as for a cube, for example). On smooth surfaces, ridges and valleys, also called crest

[1]There is significant variability in terminology—these are often called silhouettes [Markosian et al. 1997; Hertzmann and Zorin 2000].

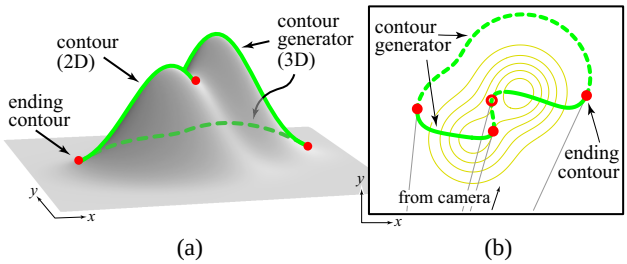


Figure 2: The contour generator is the set of points on the surface whose normal vector is perpendicular to the viewing direction. (a) When projected into the image, its visible portions are called the contour. (b) A topographic map of the surface in (a) with the contour generator shown in green. The portion that projects to the contour is drawn solid.

lines, provide features much like creases, and can help convey the structure and complexity of an object [Interrante et al. 1995]. Ridges and valleys, however, lack the view-dependent nature that hand-drawn pictures possess. Consider a drawing of a face. In profile, an artist would indeed draw a line along the ridge of the nose (as in Figure 1). However, for a frontal view the artist would instead draw lines on one or both sides of the nose (as in the top right of Figure 9).

Whelan [2001] has proposed view-dependent lines for rendering terrain, called formulated silhouettes. Formulated silhouettes are determined from those regions that become occluded in a single alternative view where the camera is lowered by a prescribed amount. This represents a related approach to ours in so far as these lines approximate some fraction of the suggestive contours of terrain.

Because we propose image-space as well as object-space algorithms for computing suggestive contours, our work compares with previous image manipulations that find lines in real images. In particular, Pearson and Robinson [1985] and Iverson and Zucker [1995] extract linear features along the darkest parts of the image (valleys in the image). Our image-space algorithm takes a broadly similar form (though of course without the robustness required for real images), and thus our theoretical arguments offer a new perspective on these techniques.

1.2 Background: Contours

It is obvious what contours are in an image, but defining suggestive contours requires an understanding of contours on objects. This section draws in part on Cipolla and Giblin [2000] to summarize the geometry of contour formation, and the important surface properties, such as curvature, that contours reflect.

Consider a view of a smooth and closed surface S from a perspective camera centered at $\mathbf{c}$. The *contour generator* is defined as the set of points that lie on this surface and satisfy:

$$\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{p}) = 0 \tag{1}$$

where $\mathbf{p} \in S$ is a point on the surface, $\mathbf{n}(\mathbf{p})$ is the unit surface normal at $\mathbf{p}$, and $\mathbf{v}$ is the view vector: $\mathbf{v}(\mathbf{p}) = \mathbf{c} - \mathbf{p}$. From the typical (generic) viewpoint, the contour generator consists of a set of disconnected loops on the surface. The *contour* consists of the visible portions of these curves, projected into the image plane. Wherever the contour generator is viewed from one of its tangent directions, the contour abruptly stops—this is an *ending contour*. Figure 2(a) illustrates contour generators, contours, and ending contours. A top view of this surface appears in (b), with the contour generators from (a) portrayed directly on the surface.

When working with polyhedra, it is easy to compute the locations of contours. The contour generator is the set of polyhedral edges that join a polygon facing the camera with one facing away from it [Appel 1967]. This strategy detects sign changes in $\mathbf{n} \cdot \mathbf{v}$. However, with smooth surfaces, we must solve (1) over the entire surface [Hertzmann and Zorin 2000].

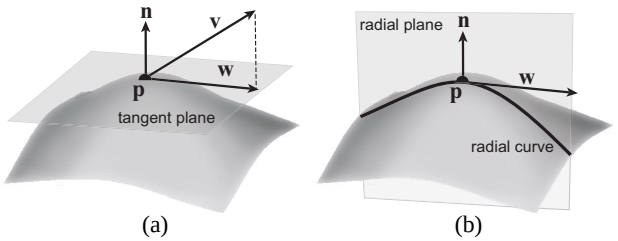


Figure 3: (a) The view vector $\mathbf{v}$ is projected onto the tangent plane to obtain $\mathbf{w}$. (b) The radial plane is formed by $\mathbf{p}$, $\mathbf{n}$ and $\mathbf{w}$ and slices the surface along the radial curve—the curvature of which is $\kappa_r(\mathbf{p})$.

In characterizing suggestive contours, we will also use the notion of the *curvature* of a curve. The curvature $\kappa(\mathbf{p})$ at a point $\mathbf{p}$ on a curve is the reciprocal of the radius of the circle that best approximates the curve at $\mathbf{p}$ [Hilbert and Cohn-Vossen 1932; do Carmo 1976]. Smaller curvature values correspond to larger circles; a line has curvature zero. The sign of the curvature requires an orientation to be specified (using a normal vector). Our convention is that when the circle is beneath the curve (i.e. the normal vector points away from the center of the circle), the curvature is positive: a convexity.[2] Concave parts have negative curvature. Zero curvature corresponds either to an inflection point or a line.

The curvature of the surface S at a point $\mathbf{p}$ is measured along a chosen curve that sits on the surface and passes through $\mathbf{p}$. Commonly, this curve is obtained by intersecting the surface with the plane that contains $\mathbf{p}$, the unit surface normal $\mathbf{n}$, and a specific direction $\mathbf{d}$ which lies in the tangent plane of S at $\mathbf{p}$. This construction yields the *normal curvature*, which varies smoothly with the direction, and ranges between the principal curvatures $\kappa_1(\mathbf{p})$ and $\kappa_2(\mathbf{p})$, which are realized in their respective principal curvature directions [do Carmo 1976]. Of particular relevance in this work is the *radial curvature* $\kappa_r(\mathbf{p})$ [Koenderink 1984], which is the normal curvature of the surface in the direction of $\mathbf{w}$ defined as the (unnormalized) projection of the view vector $\mathbf{v}$ onto the tangent plane at $\mathbf{p}$. See Figure 3. We are extending Koenderink's definition—κ_r was originally defined only along the contour generator, where $\mathbf{v}$ already sits in the tangent plane (so that $\mathbf{w} = \mathbf{v}$). The radial curvature remains undefined wherever $\mathbf{v}$ and $\mathbf{n}$ are parallel (as $\mathbf{w} = 0$), but these surface locations are not of concern in this work.

2 Suggestive contours

Informally, suggestive contours are curves along which the radial curvature is zero and where the surface bends away from the viewer (as opposed to bending towards them). Equivalently, they are those locations at which the surface is *almost* in contour from the original viewpoint—locations at which the dot product in (1) is a positive local minimum rather than a zero. They also correspond to true contours in relatively nearby viewpoints. In this section, we define suggestive contours formally in terms of a *suggestive contour generator*, which sits on the surface. Figure 4 illustrates this. Figure 4(a) overlays the suggestive contours drawn in blue on the image of Figure 2(a), while Figure 4(b) presents the suggestive contour generator in a topographic view; the portion that projects to the suggestive contour is drawn solid.

[2] While this is the opposite convention from do Carmo [1976], it corresponds to outward-pointing surface normals.

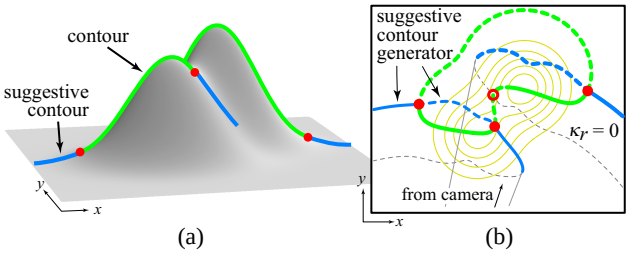


Figure 4: (a) Suggestive contours (shown in blue) extend the actual contours of the surface (shown in green). (b) A topographic view showing how the suggestive contour generators cross contours at the ending contours. The portion of the suggestive contour generator that projects to the suggestive contour is drawn solid.

The situation illustrated in Figure 5 helps to motivate our definitions. In this case, a viewpoint at **c** sees a contour at **q**, and in a nearby view **c**′ the contour has "slid" along the surface, to **q**′. Point **p** is different, however. This point is a contour when viewed from **c**′, but in no other viewpoint closer to **c**. Rather than sliding along the surface as the camera position varies between **c** and **c**′, it suddenly appears. When viewed from **c**, **p** makes a suggestive contour, while **q**′ does not. Any definition of suggestive contour generators must pick out the location of **p** in Figure 5 from among all the points on the surface.

In the remainder of this section, we offer a primary definition of suggestive contour generators, two equivalent definitions, and some further descriptive results. The context for all three definitions comes from Section 1.2: the surface S is viewed from a point **c**. Defined at every point $\mathbf{p} \in S$ is the unit surface normal $\mathbf{n}(\mathbf{p})$, the view vector $\mathbf{v}(\mathbf{p}) = \mathbf{c} - \mathbf{p}$, and $\mathbf{w}(\mathbf{p})$ which is the projection of $\mathbf{v}(\mathbf{p})$ onto the tangent plane at **p**. We exclude from this discussion all locations where **n** and **v** are parallel (where $\mathbf{w} = 0$), as the radial plane is not defined there. When we define the suggestive contour generator, we are defining the suggestive contour generator of S from the viewpoint **c**.

2.1 Definition: Zeros of radial curvature

Recall that the radial curvature is measured along a curve in the radial plane, as diagrammed in Figure 3(b). Suggestive contours are points of inflection on these curves when viewed from the convex side, along which a contour will eventually first appear. See Figure 6(a). These points of inflection occur where the radial curvature changes sign, and the radial curvature is increasing towards the camera. Thus we have:

> **Definition I:** The suggestive contour generator is the set of points on the surface at which its radial curvature κ_r is 0, and the directional derivative of κ_r in the direction of **w** is positive:
>
> $$D_{\mathbf{w}}\kappa_r > 0 \qquad (2)$$

Here, the directional derivative $D_{\mathbf{w}}\kappa_r$ is defined as the differential [do Carmo 1976] of $\kappa_r(\mathbf{p})$ applied to **w**, or $d\kappa_r(\mathbf{w})$.

Returning to the situation in Figure 5, we see there are actually two inflection points on the curve: **p** and **r**. Of these, only **p** satisfies (2) and hence is part of the suggestive contour generator.

The solution of $\kappa_r = 0$ is a set of closed loops on the surface (from a generic viewpoint), just as the contours are. We see the portions of these loops as dictated by (2) drawn in blue in Figure 4(b); the view vector is tangent to those locations where (2) cuts these loops. Furthermore, we see that contours and suggestive contours meet at the ending contours. This is because the radial curvature at ending contours is zero [Koenderink 1984]. In fact, Hertzmann and Zorin [2000] solve $\kappa_r = 0$ (which they call the cusp function) along the contour generator to locate the points of ending contour. This leads to renderings of contours with accurate visibility.

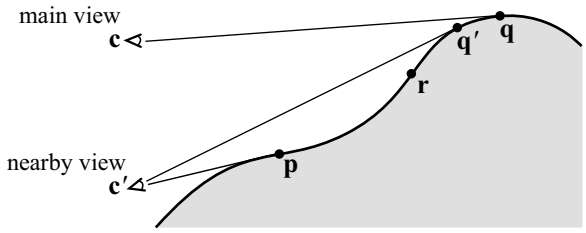


Figure 5: A situation showing both contours (**q**) and suggestive contours (**p**), as seen by the main viewpoint **c**. As the viewpoint moves to **c**′, a contour suddenly appears at **p**, while the contour at **q**′ slides along the surface from **q**.

2.2 Equivalent definitions

2.2.1 "Almost contours": minima of n·v

Our second definition of suggestive contours is those locations at which the surface is almost a contour.[3] At these locations $\mathbf{n}\cdot\mathbf{v}$ does not reach zero, but is a local minimum along **w**. Thus we have:

> **Definition II:** The suggestive contour generator is the set of minima of $\mathbf{n}\cdot\mathbf{v}$ in the direction of **w**.

Equivalence of I and II: We note that local minima of $\mathbf{n}\cdot\mathbf{v}$ in the direction of **w** correspond to zeros of the directional derivative of $\mathbf{n}\cdot\mathbf{v}$ in the direction of **w** where the second directional derivative in **w** is positive. Specifically:

$$D_{\mathbf{w}}(\mathbf{n}\cdot\mathbf{v}) = 0, \quad \text{and} \qquad (3)$$

$$D_{\mathbf{w}}(D_{\mathbf{w}}(\mathbf{n}\cdot\mathbf{v})) > 0 \qquad (4)$$

In rewriting (3) as:

$$D_{\mathbf{w}}(\mathbf{n}\cdot\mathbf{v}) = D_{\mathbf{w}}\mathbf{n}\cdot\mathbf{v} + \mathbf{n}\cdot D_{\mathbf{w}}\mathbf{v}$$

we note that **w** can replace **v** in the first term, as $D_{\mathbf{w}}\mathbf{n}$ lies in the tangent plane because **n** maintains unit length. Also, it follows from $\mathbf{v} = \mathbf{c} - \mathbf{p}$ for $\mathbf{p} \in S$ that derivatives of **v** lie in the tangent plane; so the second term is zero because $D_{\mathbf{w}}\mathbf{v}$ is perpendicular to **n**. So:

$$D_{\mathbf{w}}(\mathbf{n}\cdot\mathbf{v}) = D_{\mathbf{w}}\mathbf{n}\cdot\mathbf{w} = -II(\mathbf{w},\mathbf{w}) = (\mathbf{w}\cdot\mathbf{w})\kappa_r$$

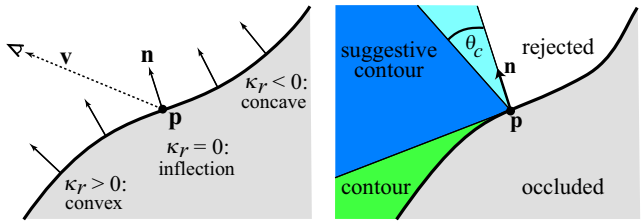
shows that $D_{\mathbf{w}}(\mathbf{n}\cdot\mathbf{v})$ is by definition the (negated) second fundamental form $-II$ [do Carmo 1976] which is a positive scaling of the radial curvature κ_r, and has the same zeros as κ_r. Computing $D_{\mathbf{w}}\left((\mathbf{w}\cdot\mathbf{w})\kappa_r\right)$ at $\kappa_r = 0$ shows the equivalence of (2) and (4).

2.2.2 Contours in nearby viewpoints

Recall that **q**′ in Figure 5 is not a suggestive contour because it slid along the surface from the contour **q** as the viewpoint changed. However, **q**′ is still a contour in a nearby viewpoint. This indicates that our informal definition of suggestive contours as simply those points that are contours in nearby views is incomplete. We refine this definition of suggestive contours to be those points that are contours in "nearby" viewpoints, but do not have "corresponding" contours in any closer views.

A measure of distance to nearby viewpoints as well as the correspondence induced by changing the viewpoint are defined in terms of the radial plane—see Figure 3(b). The *radial distance* at **p** from the main viewpoint **c** to an alternative view **c**′ is the angle formed at **p** by the points **c**, **p** and the projection of **c**′ onto the radial plane

[3] Koenderink uses the term *almost contour* in passing when describing the geometry of a specific shape: [Koenderink 1990], p. 304.



(a) (b)

Figure 6: A portion of a surface around the point **p**, as seen in the radial plane. (a) A suggestive contour appears at the inflection point on the radial curve. (b) If the location of the camera is in the dark blue region, the surface will have a suggestive contour at **p**.

of **c**. Similarly, correspondence between contours across infinitesimal changes in view is defined in terms of differential motion along the surface in the radial plane. (This differs from correspondences using epipolar geometry, used when estimating shape from contours [Cipolla and Giblin 2000].) These preliminaries give us:

> **Definition III:** The suggestive contour generator is the set of points on the contour generator of a nearby viewpoint (of radial distance less than 90 degrees) that are not in radial correspondence with points on the contour generators of any (radially) closer viewpoint.

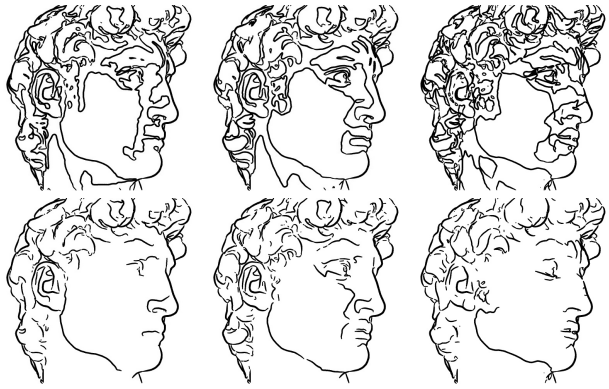
Figure 6(a) motivates the relationship between definitions **II** and **III**. Point **p** has the surface normal that is a local minimum of $\mathbf{n}\cdot\mathbf{v}$, so it would be a suggestive contour by definition **II**. As the viewpoint is moved to the left, eventually real contours begin to appear on the surface. They appear when the camera moves below the tangent plane (i.e. in Figure 6(b), moves from the blue to the green region), and for no closer viewpoints. Additionally, points to the left of **p** only produce contours at more distant viewpoints, and are in radial correspondence with **p** in the sense of definition **III**.

Equivalence of II and III: In general, consider a point **p** on the surface, a designated camera view **c**, and consider a radially nearby view $\mathbf{c}'$ that has **p** in a contour generator. The nearby view must be tangent to the surface at **p**. Suppose **p** is a local minimum of $\mathbf{n}\cdot\mathbf{v}$ in the direction of **w**. Since **n** is always closer to the camera direction near **p**, there cannot be a correspondence with a point on a contour generator in a nearer view than $\mathbf{c}'$; so **p** is a suggestive contour by **III**. Conversely, if **p** is not a local minimum of $\mathbf{n}\cdot\mathbf{v}$, then a point near **p** and in radial correspondence with **p** will become a contour before **p** does. So **p** is not a suggestive contour. It follows that definitions **II** and **III** are equivalent.

2.3 Viewpoint dependence and stability

For the diagram in Figure 6(b), consider all possible viewpoint locations for which **p** is visible—the points above its tangent plane and outside the surface. Of these, **p** is a suggestive contour generator only for those shaded in (light or dark) blue. Below this are areas where there can be a contour at or nearby **p**, or **p** is occluded.

When viewed from the other side of the normal vector, nothing is drawn at **p** due to (2). Applying this test is crucial, as the entire closed loops of the solution to $\kappa_r = 0$ do not resemble what an artist would draw. Figure 7(a) compares a rendering of the full set of zeros of κ_r to the suggestive contours. We note that (2) may be applied to other lines on the surface, such as zeros of the mean (H) or Gaussian curvature (K). Figure 7(b) shows the zeros of the mean curvature both with and without $D_{\mathbf{w}}H > 0$ applied, and (c) shows the analogous drawings for Gaussian curvature. Note, however, that the curves in (b) and (c) are attached to the surface; their



(a) (b) (c)

Figure 7: This figure demonstrates the importance of performing the derivative test, even for different families of curves. On the top are the entire solutions to (a) $\kappa_r = 0$, (b) $H = 0$ (the mean curvature), and (c) $K = 0$ (the Gaussian curvature). Below, the appropriate derivative test has been applied: (a) $D_{\mathbf{w}}\kappa_r > 0$, (b) $D_{\mathbf{w}}H > 0$, and (c) $D_{\mathbf{w}}K > 0$. In each case, the contours are also drawn.

view-dependence comes through their dependence of the derivative test on **w**. Drawing curvature zeros is a practiced visualization tool—drawing the parabolic lines (the complete zeros of the Gaussian curvature) is perhaps most common—but these efforts have not connected with line-drawing more generally.[4] Given the dramatic effects of (2), this is hardly surprising.

As the viewpoint starts to approach a head-on view of a surface location, the suggestive contour generators there can change drastically with small changes in the viewpoint (as **w** changes direction quickly). At such a viewpoint, which would fall in the light blue region in Figure 6(b), these parts of the suggestive contour generator are too unstable to give meaningful information about shape. Therefore, we additionally enforce:

$$0 < \theta_c < \cos^{-1}\left(\frac{\mathbf{n}(\mathbf{p})\cdot\mathbf{v}(\mathbf{p})}{\|\mathbf{v}(\mathbf{p})\|}\right) \tag{5}$$

This, in addition to the visibility of **p**, define the suggestive contours. This threshold θ_c places a tighter bound (than **III** does) on how nearby a contour-producing viewpoint can be: the radial distance of the viewpoints considered is at most $90 - \theta_c$ degrees.

2.4 Smoothness properties

For suggestive contours to be effective at conveying shape, they have to merge seamlessly with genuine contour information from the same viewpoint. In fact, suggestive contours are related to contours in one of two ways. In the first case, the suggestive contour *anticipates* a new loop in the contour generator that is present only in nearby views. The suggestive contour appears as a new line, away from any contour, which portrays an unoccluded shape feature. See the top of Figure 8, which depicts contours in green and suggestive contours in blue.

In the second case, an ending contour lengthens when seen from a nearby view. Away from the ending contour, contour generator points slide to their corresponding point on the contour generator in the nearby view; the points **q** and $\mathbf{q}'$ in Figure 5 are related this way. But no radial correspondence is defined at the ending contours. So here we get a suggestive contour that *extends* the ending contour. See the bottom of Figure 8.

[4] Famously, Felix Klein had the parabolic lines drawn on a copy of the Apollo of Belvedere [Hilbert and Cohn-Vossen 1932; Koenderink 1990], but was discouraged when the lines had no obvious aesthetic significance.

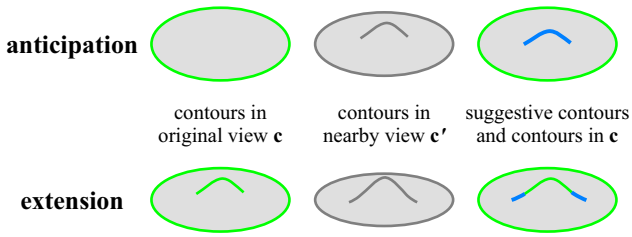


Figure 8: We classify suggestive contours as either *anticipation*, resulting from new contours that appear in nearby views, or *extension*, resulting from contours that increase in length in nearby views.

In either case, as the viewpoint approaches a place where a contour appears (or perhaps lengthens), the suggestive contour joins smoothly with the contour—with G^1 continuity. This is critical for producing comprehensible images. The presence of suggestive contours should not be distracting—they should look just like the true contours (and in fact, it should be difficult to tell them apart, except in the most obvious cases).

In establishing this smoothness property, we have to distinguish the curves on the surface from their projections in the image. When a suggestive contour generator crosses a contour generator at an ending contour, the curves are not lined up on the surface (and in fact, they can be perpendicular). Thus we focus on the projection into the image of the tangent plane at the ending contour; this is a line L. *Any* curve on the surface that arrives at the ending contour will do so in the tangent plane; its tangent vector will lie along L provided it doesn't project to a point. The tangent to the suggestive contour generator cannot project to a point, because that would imply that $D_{\mathbf{w}}\kappa_r = 0$ which contradicts (2). While the tangent to the contour does project to a point, it still lines up with L [Cipolla and Giblin 2000]. Finally, visible curves on the surface must approach the ending contour along L from the visible side, so the suggestive contour cannot meet the contour at a cusp. It follows that the suggestive contours line up with the contours with G^1 continuity.

3 Computing suggestive contours

The extraction of suggestive contours can draw upon any of the three definitions from Section 2. We expect most implementations would use **I** or **II**. However, one can think of the algorithm for formulated silhouettes [Whelan 2001] as a partial solution to **III**.

This section describes two implemented systems that detect suggestive contours working from triangular meshes. The first computes them directly on the mesh by solving **I**. The second renders a diffusely shaded image which is processed to find the location of the suggestive contours. Although it is only an approximate solution to **II**, we have found the behavior to be similar to that of the other method. Furthermore, it so simple to implement that it is valuable as a prototyping method.

3.1 Object-space algorithm

Our algorithm proceeds first by finding the solution to $\kappa_r = 0$ by searching over the entire mesh. The suggestive contours are computed by trimming this solution using (2) and (5). We next detect contours on the mesh, and render all the lines with visibility assessed using the method described by Kalnins et al. [2002].

Detection: Working directly on the polygon mesh, we follow Hertzmann and Zorin [2000] in representing the suggestive contour generator as a contiguous path consisting of line segments that cross triangles. Solving **I**, we find zero crossings of κ_r exhaustively by testing each triangle. While the exhaustive algorithm presented by Hertzmann and Zorin [2000] is used to find zero crossings of $\mathbf{n}\cdot\mathbf{v}$, it is suitable for finding zero crossings of any function sampled at the vertices. This requires values of κ_r at each vertex, which are computed using the Euler formula for normal curvature from the principal curvatures [do Carmo 1976]:

$$\kappa_r(\mathbf{p}) = \kappa_1(\mathbf{p})\cos^2\phi + \kappa_2(\mathbf{p})\sin^2\phi \tag{6}$$

which takes ϕ as the angle between $\mathbf{w}(\mathbf{p})$ and the principal curvature direction corresponding to κ_1.

The principal curvatures and directions are computed using established techniques. First, the meshes are filtered using Taubin smoothing [Taubin 1995a]. The curvature information is then estimated from this smoothed geometry using the T-algorithm presented by Taubin [1995b], with the modifications suggested by Hameiri and Shimshoni [2002] (except for the inverse distance weighting to accumulate information from mesh neighborhoods).

This yields a set of loops on the surface, which are subject to (2) and (5). This requires computing $D_{\mathbf{w}}\kappa_r$, which proceeds first by computing a constant gradient vector within in each triangle, and then averaging these at the vertices (weighted by angles).

Filtering: There are additional difficulties caused by errors in the curvature estimation. For instance, when the minimum principal curvature is close to zero and $\mathbf{w}$ roughly looks down its corresponding principal direction, spurious zero crossings are present that result from noise. Marr and Hildreth [1980] encountered an analogous problem when detecting edges in images as zero crossings—their solution checks that the derivative magnitude is large. Similarly, we apply a small positive threshold to $D_{\mathbf{w}}\kappa_r$ to avoid those situations which are likely to produce spurious zero crossings:

$$t_d < \frac{D_{\mathbf{w}}\kappa_r}{\|\mathbf{w}\|} \tag{7}$$

The thresholds t_d and θ_c are most useful when they work consistently for a range of objects and viewpoints. As a first step, we normalize t_d for object size. Then, we use the idea of hysteresis thresholding [Canny 1986] to increase granularity. Our method proceeds first by filtering with (5) and (7). Then, segments that were discarded are introduced again if they meet a second (less stringent) set of thresholds t_d' and θ_c', and neighbor a segment that was not discarded. This process is repeated until convergence. Finally, chains of segments shorter than t_s are discarded.

3.2 Image-space algorithm

Instead of identifying locations on the mesh, it is possible to determine suggestive contours directly from a rendered image. There are many possible approaches; one could, for instance, develop a successful image-based analogue of our object-space algorithm, which renders the polygon mesh colored by κ_r. Instead, we present a simple yet effective approximation to **II**.

Our algorithm proceeds as follows. The dot product in (1) is approximated for the entire image using the graphics hardware by rendering a smoothly shaded image with a diffuse light source placed at the camera origin. Pearson and Robinson [1985] motivate their approach using this same configuration. In this rendered image, we detect suggestive contours as steep valleys in intensity—this finds stable minima of $\mathbf{n}\cdot\mathbf{v}/\|\mathbf{v}\|$. (Actually, this will also find contours, when the resolution is high enough and what's behind is bright.) The normalization in the lighting computation only approximates $\mathbf{n}\cdot\mathbf{v}$—it is more accurate when the viewpoint is distant. In addition, since one cannot determine the radial direction on the mesh from the rendered image, we must find all valleys, not just those minima in the radial direction.

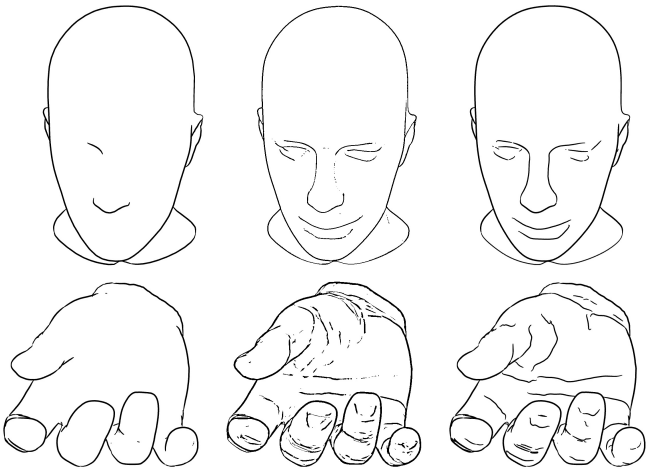


Figure 9: Comparison of image- and object-space algorithms for computing suggestive contours. Left: contours (computed in object space). Center: image-space algorithm. Right: object-space algorithm.

There are many successful image processing algorithms for detecting valleys [Iverson and Zucker 1995; Steger 1999]. However, they are typically complex—the presence of noise in captured images is largely at fault for this. Given we are working with rendered images that are also smooth (away from occlusion boundaries), we can use a much simpler algorithm.

The basic idea is as follows. While a pixel in a valley is not necessarily the minimum intensity value in a neighborhood, it will be among a thin set of dark pixels that cuts across the neighborhood. If the valley is steep, the neighborhood will also contain significantly brighter pixels away from the valley; to implement (5), we can require a sufficient intensity difference that the surface must be turned meaningfully away. So to test pixel i with intensity p_i our algorithm collects all the pixel intensities within a radius r centered around it; across this circular window the greatest intensity is $p_{\max}$. We label i a valley if two conditions are met: no more than a certain percentage s of the pixels in this disk are strictly darker than p_i; and $p_{\max} - p_i$ exceeds a fixed threshold d. (To minimize the effects of discretization, it is convenient to scale s as r varies by $\left(1 - \frac{1}{r}\right)$ and d by r.) As a final step, we remove small irregularities with a median filter of radius r.

4 Results

This section demonstrates results from our image- and object-space algorithms, then briefly compares suggestive contours with other effects. As seen in Figure 9, the image- and object-space algorithms produce very similar results, but the object-space algorithm has the advantage of generating continuous stroke paths on the surface. We believe that the image-space method, in contrast, will not be able to produce nicely-rendered strokes in a hand-drawn style. Because the algorithm produces a discrete set of pixels, good image quality is dependent on heavy post-processing, such as median filtering. Even so, its avoidance of higher-order derivatives makes it much less susceptible to problems resulting from noise in the geometry—this accounts for much of the differences in the locations or presence of the suggestive contours on the hand. The object-space algorithm is generally more efficient for larger images and medium-sized (50–100K polygon) models, aside from the initial preprocessing to compute curvatures (which takes several seconds for a medium-sized polygon model).

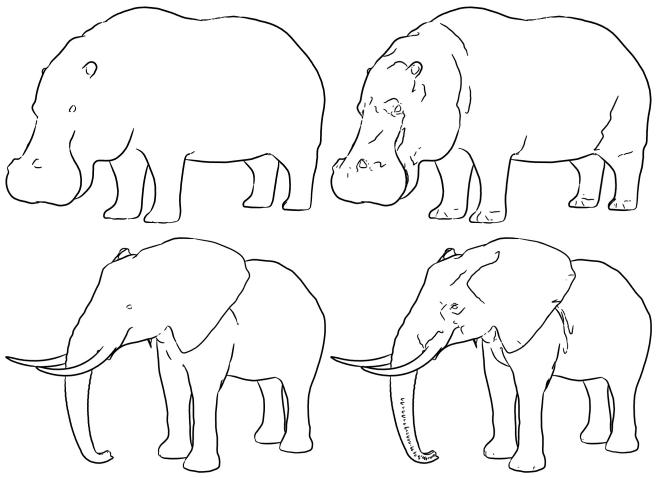


Figure 10: Results of the object-space algorithm (Section 3.1). Left: only contours; Right: both contours and suggestive contours.

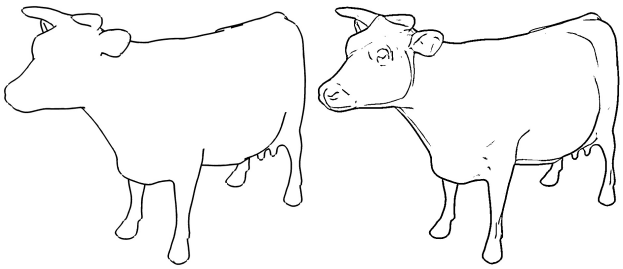


Figure 11: Results of the image-space algorithm (Section 3.2). Left: only contours; Right: both contours and suggestive contours.

Results of our object-space algorithm are shown in Figure 10, as well as Figure 1 and the center of Figure 12. In each case, the left image displays a drawing with only the contours. In these examples, the values of θ_c ranged from 20 to 30 degrees and t_d from 0.02 to 0.08; with $t_s = 2$, $\theta_c' = 0$ and $t_d' = 0$ for all examples. Computing the contours and suggestive contours on a 50K polygon mesh takes 0.15 seconds. (Available contour extraction techniques [Markosian et al. 1997; Gooch et al. 1999; Hertzmann and Zorin 2000; Sander et al. 2000] can certainly be adapted to improve our performance by not checking every polygon.) Figure 11 and the right of Figure 12 show results of our image-space algorithm for finding suggestive contours, using the parameter values $r = 4.0$, $s = 0.2$ and $d = 0.25$. Computation lasted 0.4 seconds for a 640x480 image (the time is dominated by the image operations for these examples).

The image-space rendering at right in Figure 12 uses a version of the David at a substantially finer scale than the object-space rendering in the center. The substantial noise in the higher-resolution mesh posed a difficulty in applying (2) successfully; while the strokes were placed reasonably, they were excessively fragmented.

In each of these examples, we selected viewpoints that produce compelling results; not all viewpoints are created equal (as artists already know). For the object-space algorithm, θ_c and t_d were adjusted to cull only those lines that convey shape most effectively.

From renderings of contour alone, the limited shape information that is available seems to present only an undifferentiated, smooth and round relief. Suggestive contours enrich and differentiate the conveyed shape. As in the lines at the head and shoulders of the figures, suggestive contours define bulges and allow the viewer to localize their convexity more precisely. As in the lines on the rump

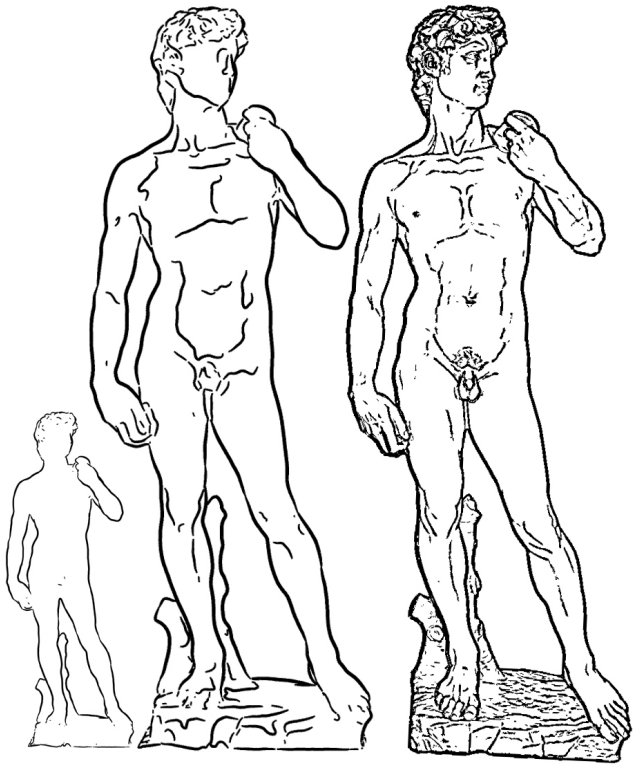


Figure 12: Left: only contours; Center: results of the object-space algorithm for the David (80K polygons), drawn using thicker lines (for stylistic reasons); Right: results of the image-space algorithm for the David (500K polygons).

of the cow and hippo, or the torso and knees of the David, suggestive contours highlight hollows and dimples that the viewer would not otherwise have suspected. Suggestive contours convey small shape features as well as large ones: in the eyes or toes of the figures, in the elephant's trunk, in the texturing of the bunny (in Figure 13, right). These bumps and indentations show up in much the same manner as an artist would depict them. Suggestive contours can also convey folds in the surface when they are deep enough to contain an inflection point, as are the wrinkles in the hand.

Comparisons to other strategies We now present comparisons to two methods whose goal is also to draw lines that are visual cues for shape. First, inspired by the work of Saito and Takahashi [1990], we implemented an algorithm that finds discontinuities in the depth buffer using a Canny edge detector [Canny 1986], and renders these lines. Next, we implemented an algorithm for detecting ridge and valley lines. We render just the valleys here; in our experience, we have found that rendering both ridges and valleys yields a cluttered result. (For many models, it is possible to locate ridges and valleys as simply the creases in the model—edges along which the dihedral angle is small. For more highly-tessellated models, however, this definition is inadequate, and we instead define ridges and valleys as the local maxima or minima, respectively, of the larger (in magnitude) principal curvature, in the corresponding principal direction [Interrante et al. 1995; Lengagne et al. 1996]. In order to compute these robustly, we find the principal curvatures and directions and perform non-maximum suppression and hysteresis thresholding operations similar to those used in the Canny edge detector [Canny 1986]. In particular, for each vertex of the mesh we locate the two neighboring vertices with the most positive and negative projections onto the first principal direction, and discard the

Figure 13: Comparison of several visual effects (two views of each): Left: edge detection applied to the depth map to extract depth discontinuities. Center: contours with valley lines computed as local maxima of curvature. Right: contours with suggestive contours computed using the object-space algorithm.

vertex unless its first principal curvature is more extreme than both. Then, we connect all remaining vertices with chains of edges, keeping only the chains for which each vertex has curvature above some low threshold and at least one vertex has curvature above some high threshold.)

It's clear from Figure 13 that computing edges of the depth map (left) conveys little more than contour does. In fact, we needed to adjust the edge detector thresholds to be very sensitive to get any additional lines at all. Valleys (center) sometimes highlight important shape features, as with the nose, eyes, ears and feet of the bunny; unlike suggestive contours, these features are localized on the surface, and as the view changes to wash out the relief of these features, they start to look like surface markings. Elsewhere, as at the haunches, the valleys can reduce to a distracting network, evocative of terrain. Finally, the suggestive contours (right) convey the shape effectively as in the earlier examples. But the many lines portraying the texture of the bunny give the image a somewhat disorganized appearance, without highlighting structural features, for example on the face, as explicitly as the valleys do.

5 Discussion and Conclusion

In this paper, we have explored drawings that portray objects' suggestive contours. We have characterized these suggestive contours simultaneously in terms of the geometry of the object under the current view, and in terms of the appearance of the object in nearby views. On this characterization, suggestive contours are a way of exaggerating a rendering, by using lines that are almost contours to give added visual evidence of surface geometry. Our perceptual abilities make us sensitive to the reliable visual information in such renderings, and they thus can portray richer and more detailed shape information than true contours do alone. We have supported this intuitive account of suggestive contours with a mathematical formalization which substantiates the compatibility of true contours and suggestive contours, and which enables us to describe a variety of algorithms, including some familiar ones, that can extract and stabilize suggestive contours.

Our initial progress with suggestive contours motivates more substantial perceptual, aesthetic and computational investigations. Perceptual research that establishes how people perceive drawings

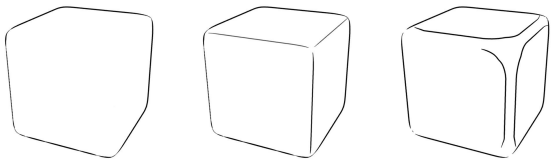


Figure 14: A rounded cube has no suggestive contours (left). Ridges can mark its features (center). The effect on the right works from zero crossings obtained by offsetting all of the curvatures, so that the planar faces registered with negative curvature.

with suggestive contours, as well as a comparative study of suggestive contours with lines that people actually draw, would help to make precise the visual information that suggestive contours convey. Perhaps most relevant is a study by Koenderink et al. [1996] which compared the perception of shape given (external) silhouettes, hand-made line drawings (which resemble the results here), and shaded images. One conclusion is that the interpretation of the line drawings was very close to that of shaded figures. In particular, internal lines (which without context could easily be interpreted as contours) do not necessarily cause people to believe that the surface is perpendicular to the view direction. The fact that suggestive contours fuse seamlessly with true contours does not mean that viewers will perceive the shape incorrectly from them.

Aesthetic challenges include generalizing the account to apply across levels of detail and across new classes of shapes. Level-of-detail for object-space contour algorithms is an open problem, but is particularly important for suggestive contours since they are so rich. There are a number of established strategies for level of detail, and comparison of the two renderings of the David in Figure 12 and the head of David in Figure 1 leaves us optimistic about them.

Objects without concavities have no suggestive contours; the rounded cube portrayed in Figure 14 is such a shape. Ridges can be useful in situations like this, as seen in the center. Nevertheless, variations on suggestive contours may still help depict their shape; the image on the right was constructed by instead solving $\kappa_r = \frac{1}{2}$ (the particular value was chosen by hand).

Finally, on the computational side, greater robustness is required in computing suggestive contours in object space. Animation also presents a key challenge. The behavior of suggestive contours through viewpoint changes is encouraging. Because suggestive contours anticipate and extend true contours, they can eliminate flickering for contours that would otherwise appear only briefly. On the other hand, further methods to select and stabilize suggestive contours are required for effective animation. In early experiments we have found that while some suggestive contours are very stable, others move. We are optimistic that meaningful progress will be possible, because the unstable lines seem to be the least important for conveying shape, and tend to be caused by shallow surface features and noise.

Acknowledgments

Thanks to Jordan Ellenberg, Stephen Greenfield, Spike Hughes, Michael Kazhdan, Michael Kowalski, Michael Leyton, Peter Meer, Ilan Shimshoni, Manish Singh, Matthew Stone, and especially Lee Markosian, Rob Kalnins, and Philip Davidson who participated in the discussion that led to this work and in its implementation.

This research was supported by the NSF (SGER 0227337) and Intel Labs. The David model is from the archives of the Digital Michelangelo Project, the bunny model is from the Stanford 3D Scanning Repository, and the hand model is from Viewpoint.

References

APPEL, A. 1967. The notion of quantitative invisibility and the machine rendering of solids. *Proceedings of ACM National Meeting*, 387–393.

CANNY, J. 1986. A computational approach to edge detection. *IEEE PAMI 8*, 6, 679–698.

CIPOLLA, R., AND GIBLIN, P. J. 2000. *Visual Motion of Curves and Surfaces*. Cambridge Univ. Press.

DO CARMO, M. P. 1976. *Differential Geoemtry of Curves and Surfaces*. Prentice-Hall.

GOOCH, B., SLOAN, P., GOOCH, A., SHIRLEY, P., AND RIESENFELD, R. 1999. Interactive technical illustration. In *Proc. of the 1999 symposium on Interactive 3D graphics*, 31–38.

HAMEIRI, E., AND SHIMSHONI, I. 2002. Estimating the principal curvatures and the darboux frame from real 3D range data. In *International Symposium on 3D Data Processing Visualization and Transmission*, 258–267.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *SIGGRAPH 2000*, 517–526.

HILBERT, D., AND COHN-VOSSEN, S. 1932. *Geometry and the Imagination*. Springer.

INTERRANTE, V., FUCHS, H., AND PIZER, S. 1995. Enhancing transparent skin surfaces with ridge and valley lines. *IEEE Visualization 1995*, 221–228.

IVERSON, L. A., AND ZUCKER, S. W. 1995. Logical/linear operators for image curves. *IEEE PAMI 17*, 10, 982–996.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. WYSIWYG NPR: Drawing strokes directly on 3D models. In *SIGGRAPH 2002*, 755–762.

KOENDERINK, J. J., VAN DOORN, A., CHRISTOU, C., AND LAPPIN, J. 1996. Shape constancy in pictorial relief. *Perception 25*, 155–164.

KOENDERINK, J. J. 1984. What does the occluding contour tell us about solid shape? *Perception 13*, 321–330.

KOENDERINK, J. J. 1990. *Solid Shape*. MIT Press.

LENGAGNE, R., FUA, P., AND MONGA, O. 1996. Using crest lines to guide surface reconstruction from stereo. In *ICPR '96*, 9–13.

MARKOSIAN, L., KOWALSKI, M. A., TRYCHIN, S. J., BOURDEV, L. D., GOLDSTEIN, D., AND HUGHES, J. F. 1997. Real-time nonphotorealistic rendering. In *SIGGRAPH 97*, 415–420.

MARR, D., AND HILDRETH, E. 1980. Theory of edge detection. *Proc. Royal Soc. London B-207*, 187–217.

PEARSON, D., AND ROBINSON, J. 1985. Visual communication at very low data rates. *Proc. IEEE 4* (Apr.), 795–812.

RASKAR, R. 2001. Hardware support for non-photorealistic rendering. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 41–46.

SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3-D shapes. In *SIGGRAPH 90*, 197–206.

SANDER, P. V., GU, X., GORTLER, S. J., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *SIGGRAPH 2000*, 327–334.

STEGER, C. 1999. Subpixel-precise extraction of watersheds. In *ICCV '99*, vol. II, 884–890.

TAUBIN, G. 1995. Curve and surface smoothing without shrinkage. In *ICCV '95*, 852–857.

TAUBIN, G. 1995. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95*, 902–907.

WHELAN, J. C. 2001. *Beyond Factual to Formulated Silhouettes*. PhD thesis, University of Hull; supervised by M. Visvalingam.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer-generated pen-and-ink illustration. In *SIGGRAPH 94*, 91–100.