EUROGRAPHICS 2003 / P. Brunet and D. Fellner
(Guest Editors)

Volume 22 (2003), Number 3

Precise Ink Drawing of 3D Models

Mario Costa Sousa Kevin Foster Brian Wyvill Faramarz Samavati

Department of Computer Science
University of Calgary, Canada

Abstract

Drawings made with precise pen strokes accurately reveal the geometric forms that give subjects their characteristic shape. We present a system for non-photorealistic rendering of precise drawing strokes over dense 3D triangle meshes with arbitrary topology. During an automatic pre-process, we construct an extended version of the edge-buffer data structure to allow the calculation of shape measures at each mesh edge, by adapting numerical methods used in geomorphology. At runtime, feature edges related to shape measures are extracted and rendered as strokes with varying thickness and pen marking styles. Stroke thickness is automatically adjusted by considering surface curvature. Pen marking styles and visual effects of ink distribution are both controlled by the user. We demonstrate precise drawing strokes over complex meshes revealing a variety of shape characteristics.

1. Introduction

Precise drawings are usually the best way for understanding structures and details of various subject matters. The primary purpose of such drawings is to make shape characteristics of the subject visible, to lessen the possibility of misinterpretation, and to please the eye in terms of proper artistic handling of the subject [1, 2]. Traditionally, precise drawings are produced in three main steps: 1) Shape characteristics (i.e. folding regions, surfaces areas, volumes, curvatures) must first be accurately identified and measured; 2) regions related to the shape measures are then lightly outlined using pencils, and 3) filled with pen strokes, with a gesture that conveys a careful constructed look [3, 1, 4].

This paper presents a new non-photorealistic rendering (NPR) object-space method for revealing shape features of highly-tessellated 3D triangle mesh models with procedurally generated pen strokes. Key goals include good rendering rates, efficient scheme for shape measures calculation and visual quality resembling traditional precise pen stroke drawings (figs. 1 and 2).

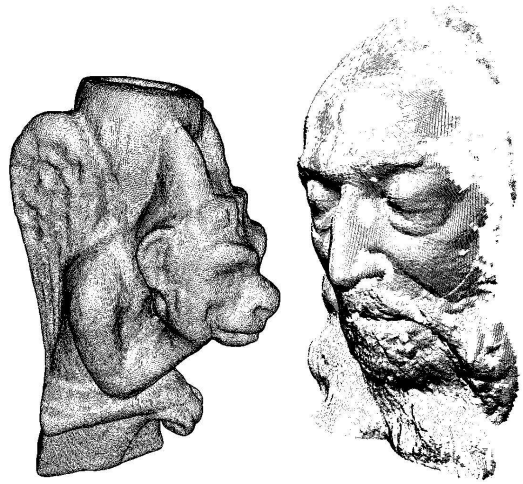

Figure 1: *Models rendered with our system.*

One of the main reasons for the use of pen strokes in the production of precise drawings is that they can represent virtually any shape if used properly [3, 1] (fig. 2). As an individual primitive, pen strokes are either lines or dots, black, narrow, and consistent in thickness. In clusters, they create a cumulative effect resulting in tone values which helps to ensure proper revealing of the subject's shape characteristics. To this end, the main challenge facing the illustrator is on representing tri-dimensionality in drawings, given the fact that strokes are essentially 2D marks placed in a plane. To overcome this limitation, illustrators: (a) adjust the thickness of the strokes by making them thicker at certain curvatures,

Figure 2: *Real precise drawings: (left) Portion of* American Portrait *by William Henson* [3]*, (middle) illustration of a hip* [5]*; (right) archeological artifact by Emily S. Damstra. Refer to the accompanying CD-ROM for additional images.*

junctions, creases [1, 4], and (b) control the cumulative effects of strokes by carefully adjusting the spacing among them.

Typical NPR approaches to procedurally generate strokes as individual primitives and to placing them directly on 3D models involve processes that can be costly. These include distributing strokes across the surfaces, evaluating hand-gestures functions (i.e. pressure, slanting, waviness), linking strokes in chains, fitting curves to stroke sequences, among others. Our approach embodies three main strategies:

i. One stroke per mesh edge: Each stroke has the same length and location of its corresponding edge, and is modeled and rendered individually (i.e. no chaining). This strategy provides rendering at reasonable rates with temporal coherence, as the strokes are fixed to their edges on the model, and are not redistributed for each frame.

ii. Edge-based shape measures: Our method calculates shape measures at every mesh edge, using only information from its two adjacent faces. This is achieved by extending the edge-buffer data structure [6] and by adapting shape measure calculation schemes from geomorphology [7].

iii. Pen stroke thickness and styles: Our system automatically adjusts the thickness of each stroke as a function of surface curvature estimated at the edge; the user controls the parameters of stroke style for placing different types of pen marks and for achieving ink distribution visual effects.

1.1. Related work

Our work is related to three main lines of investigation, which as a whole focus on calculating and drawing stylized feature edges of 3D models:

(1) Precise ink-based illustration : Different approaches for placing small pen-and-ink primitives over 3D models have been proposed. Winkenbach and Salesin [8] introduce "stroke textures" allowing procedural accumulation of strokes for stippling and other textures made with small lines such as "grass". Elber [9] presented a technique for uniformly spreading small strokes across freeform surfaces. Deussen and Strothotte [10], used a particle-based distribution to render clusters of leaves as small line primitives. Building on the previous technology of stroke textures [8], Praun et al. [11] introduced "tonal art maps" which organizes pre-rendered strokes as a sequence of mip-mapped images. Recent investigation on precise stippling has focused on the geometric relation between the stippled dots [12] and on interactive direct volume illustration systems [13]. Secord [14] developed a fast probabilistic method that places small arbitrarily-shaped primitives, including stippling. More recently, Pastor et al. [15] present an approach where stipple particles are attached to the surface of the model, using a point hierarchy to control the stipple shading density. Notice that the focus of all of these works is for creating tone and texture on 3D models. Our approach is to use pen marks to reveal shape features, instead, without considering any lighting or material information. Strokes are also distributed on a fixed basis across the mesh (i.e. one stroke per mesh edge).

(2) Shape measures in mesh models: Researchers have explored the use of lighting parameters/equations for measuring and rendering models in the context of technical and scientific illustration [16, 17, 18]. Our approach is to extract and render shape features by calculating local shape measures directly at the 3D mesh, with no need for either illumination or surface reflectance information. For polygonal meshes, shape measures are usually estimated at every vertex of the model, taking into account some local properties of the adjacent faces to each vertex, such as triangle areas, angles, and edge lengths [19, 20, 21]. In NPR, the focus has been mainly on shape measures related to principal direction of curvature to guide the stroke placement process [21, 22, 11, 23]. Our approach is to use geomorphological shape measure calculation schemes [7, 24, 25, 26]. They provide a large and computationally stable collection of shape measures. We adapt those methods to work directly with 3D triangle meshes, organized in an edge-buffer data structure [6]; the main resulting benefit is that now various shape measures can be directly calculated at each edge of the model using only the information of its two adjacent faces. The edge-buffer was primarily designed for efficient extraction and rendering of silhouette edges. By adapting geomorphological methods, the edge-buffer is now able to efficiently extract and render edges related to different shape measures as well.

(3) Line thickness: Proper adjustment of line thickness (also known as weight) greatly improves the quality of the final stroke. Existing approaches scale the thickness of strokes based on depth [27, 28, 29] or tone [27]. Our approach is to automatically adjust stroke thickness according to surface curvature and then apply different pen marking styles.

1.2. Algorithm overview

In a pre-processing stage, a single 3D triangle mesh is read, with no need for either illumination or surface reflectance information. In a single-pass pre-processing stage, an edge-buffer data structure is then constructed with automatic calculation of shape measures directly at each edge (sec. 2), by adapting numerical techniques used in digital terrain analysis (geomorphology) (sec. 3). At run-time, the edge-buffer is traversed, carrying user information on 1) which shape measures to display, 2) threshold values for the shape measures, and 3) parameters to adjust stroke style attributes. Each edge is then modeled and rendered as a single stroke, with a specific thickness and style (sec. 4). Stroke thickness is automatically adjusted by the pre-computed surface curvature measure associated with the edge (subsec. 4.1). Stroke styles are provided by an interactive stroke model, which reproduces traditional pen marks (subsec. 4.2), and visual effects of ink-distribution (subsec. 4.3).

2. The edge-buffer revisited

In our implementation, an edge buffer [6] consists of a vertex-indexed array $V[1..n]$ of pointers, where n is the total number of vertices in the mesh. Every ith element of V points to a linked list of the edges incident on vertex v_i. Each node in this linked list holds: 1) j, the id of vertex v_j adjacent to v_i; 2) (F,B,F_a,B_a), the bit fields for *front face, back face, front absolute, back absolute*, respectively, that are used for extracting feature edges at run-time [6]; 3) f_1, the index of the face sharing edge (v_i,v_j) that is first visited during construction of the edge-buffer, and 4) $(D,GA,A0,H)$, the descriptors of shape measures for dihedral angle, slope steepness, slope aspect, and mean curvature, respectively (sec. 3). Note that this linked list is sorted in increasing order of vertex ids j starting from i, and that each pair (v_i,v_j) forms an edge. The edge-buffer is then constructed by the following algorithms:

```
BUILD-EDGE-BUFFER(mesh)
1  for all triangles f ∈ mesh
2    do remove vertex indices (i,j,k) ∈ f
3       (i',j',k') ← SortIndices(i,j,k), {i' < j' < k'}
4       call InsertEdge(i',j',f)
5       call InsertEdge(i',k',f)
6       call InsertEdge(j',k',f)
```

```
INSERT-EDGE(a,b,f)
1  v ← V[a].SearchForVertex(b)
2  if v = NULL
3    then construct new v with (j,f1) = (b,f)
4         V[a].Insert(v)
5    else b = v.j, F1 = v.f1, F2 = f
6         call ShapeMeasures(a,b,F1,F2)  (sec. 3)
```

In lines 2-4 of $InsertEdge()$, the edge ab is visited for the first time, so we store the index of the face f. For the second visit (lines 5 and 6) we retrieve the face id stored at the first visit and together with the current face id, compute the shape measures associated with edge ab (sec. 3).

Finally, at run-time, feature edges are ready for extraction and display. We consider feature edges as being silhouettes, boundaries and interior edges (within the outlined form of the object) depicting a particular shape measure (sec. 3). For each new camera view, every triangle in the mesh is first determined to be either front- or back-facing. Then, the bit fields FBF_aB_a for each of the three edges of the triangle under consideration are updated with FB being XORed and F_aB_a being ORed accordingly (refer to Buchanan and Sousa [6] for more details). After processing every triangle, the edge-buffer is then traversed carrying on user given information about **1)** feature edges to be rendered, **2)** threshold values for the shape measures, and **3)** parameters values for modeling and rendering pen strokes (sec. 4). Specific feature edges are then identified by simply checking the bit field configuration: $FB = 11$ indicates silhouettes, $FB = 10$ or 01 indicates boundaries, and $FBF_aB_a = 0010$ indicates front-facing interior edges that will be rendered if one or more of their corresponding shape measures are selected for display and if the value of each of the shape measures is within its corresponding threshold range. Note that we do not use the bit field A for *artist* edges as originally defined by Buchanan and Sousa [6]. In our implementation, *artist* edges correspond exactly to the interior edges that are rendered depicting a particular shape measure.

3. Edge-based shape measures

This section describes the computations involved in function *ShapeMeasures()* (called from *InsertEdge()*, line 6), where shape measures are calculated directly at every edge ab, using only information from its two adjacent faces F_1 and F_2. One obvious measure is the **dihedral angle** D between the normals N of faces F_1 and F_2, defined as $D = \arccos(N_{F_1} \cdot N_{F_2})$. This measure depicts creases, which are edges whose dihedral angle is within threshold limits (min,max) specified by the user, assuming that $0 \leq min \leq D \leq max \leq 180$ (fig. 5(a)). In addition to dihedral angle, we would like to have a more general scheme to allow the calculation of a larger collection of shape measures to reveal more features about the shape of the model. Our approach is to investigate shape measure methods from geomorphology (the study of landforms and the processes that produce them), in particular techniques for characterizing form (geomorphometry) by means of morphometric variables [7, 24, 25, 26].

Morphometric Variables

Surface geomorphometry is most commonly modeled as Digital Elevation Models (DEMs), which are defined as col-

lections of elevation points sampled above some datum describing a terrain surface. Elevation coordinates are usually organized in a regular grid with points equally spaced in X and Y regardless of the shape of the terrain. Parameters characterizing the terrain surface, known as Morphometric Variables (MVs), are then produced from the DEMs. MVs include local shape measures such as slope, aspect, profile curvature, plan curvature, tangential curvature, flow path length, among others [7, 25, 26]. One common aspect of MVs is that they can be calculated using methods based on the approximation of differential operators by finite differences, expressed via first and second derivatives computed at every point of the DEM. Moving and evaluating a finite difference template along the regular DEM compute these derivatives. This template is derived from Taylor series expansion usually as a 9-point scheme (a 3 by 3 grid) [30].

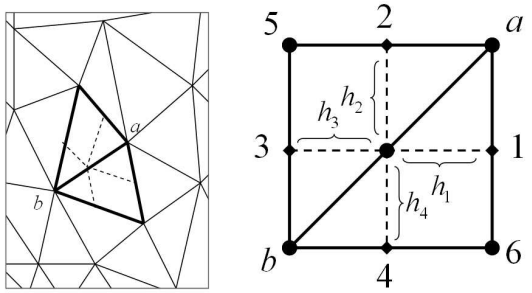

Figure 3: *Discrete diagram for a 9-point finite difference scheme for numerical differentiation (right), constructed from pair of mesh faces (F_1, F_2), shared by edge ab (left). Point 0 is the midpoint of ab. Faces F_1 and F_2 are defined by the vertices $(a,5,b)$ and $(a,b,6)$, respectively. The spaces along x and y are given by the values of h.*

We adapt the geomorphometry approach for calculating shape measures by using the local-coordinates (x,y,z) of our mesh models as points in DEMs, with z values $(0 \leq z \leq 1.0)$ being the elevations and (x,y) being the ground's coordinates. We just consider those geomorphomtrics that can evaluate locally and in a very small neighborhood, since they can be applied to arbitrary surfaces. During construction of the edge-buffer, each pair of mesh faces (F_1, F_2) defines four elevation points, from which a 9-point finite difference scheme for numerical differentiation is then constructed. The discrete diagram for this situation is illustrated in Figure 3(right): points $\{a,5,b,6\}$ correspond to the local-coordinates of F_1 and F_2 combined, and middle points $\{0,1,2,3,4\}$ are computed along edges $\{ab, 6a, a5, 5b, b6\}$ respectively. Note that because our meshes may have non-uniform edge lengths, our 3x3 template fits in the general case wherein each spacing along x (h_1, h_3) and y (h_2, h_4) can be different [30]. We can now estimate first and second partial derivatives of elevation z by plan coordinates x and y at point 0, the middle-point of the edge ab under consideration. The value of the partial derivatives f at location 0 is then formulated using central difference with variable steps. The notation $z_\#$ means "elevation z at point #" and $h_\#$ means "distance h from point 0 to #" (refer to fig. 3):

$$(f_{0,x}, f_{0,y}) = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right)_0 = \left(\frac{z_1 - z_3}{h_1 + h_3}, \frac{z_2 - z_4}{h_2 + h_4}\right) \quad (1)$$

$$f_{0,xx} = \left(\frac{\partial^2 z}{\partial x^2}\right)_0 = 2\left(\frac{h_1 z_3 - (h_3 + h_1) z_0 + h_3 z_1}{h_3 h_1 (h_3 + h_1)}\right) \quad (2)$$

$$f_{0,yy} = \left(\frac{\partial^2 z}{\partial y^2}\right)_0 = 2\left(\frac{h_2 z_4 - (h_2 + h_4) z_0 + h_4 z_2}{h_2 h_4 (h_2 + h_4)}\right) \quad (3)$$

$$f_{0,xy} = \left(\frac{\partial^2 z}{\partial x \partial y}\right)_0 = \frac{(z_a + z_b) - (z_5 + z_6)}{(h_3 + h_1)(h_2 + h_4)} \quad (4)$$

Figure 4: *Shaded relief images of* Venus *model (5,584 faces) with (1) lighter yellow referring to greater slope steepness, and with (2) darker blue and red referring to regions of greater convexity and concavity, respectively.*

The next step is to decide which MVs best represent shape measures mostly used in the production of precise drawings. We considered fundamental shape measures used by illustrators for highlighting structures of the subject, which include slant variations (in length and direction) and convex/concave formations across the shape of the subject [4, 31] †. After studying complete systems of MVs [7, 24, 25, 26], we observed that slant variations can be adequately represented by

† Mathematically, surface geometry can be classified into six categories defined by the values of the principal curvatures: elliptical (convex when both curvatures are positive or concave when both curvatures are negative), hyperbolic (convex in one direction and concave in the other), flat (zero curvature in both directions), and cylindrical (convex when curvature is positive in the non-flat direction or concave otherwise). Based on visual observations, we noted that illustrators are most interested on revealing elliptical surface patches [4, 31].

(a) Dihedral Angle D(min, max)=(20,140) *(b) Slope Steepness GA(0.48,1.0)* *(c) Slope Aspect A0(0.63,1.0)* *(d) Mean Curvature (+) +H(0.54,1.0)* *(e) Mean Curvature (-) -H(-0.5,-0.01)*

Figure 5: *Feature edges displaying thresholded shape measures within (min,max) range.*

MVs of slope steepness and aspect, and that convex and concave formations can be properly measured with mean curvature. Those MVs are calculated at the middle point 0 of the edge *ab* (fig. 3) as described next.

Slope Steepness (GA) is a measure of the rate of change of elevation. Geometrically, it is an angle ($0 \leq GA \leq 90$) between a horizontal plane and a tangential to land surface plane at the same point . This angle can be represented through first partial derivatives using the formula [32]: $GA = \arctan(f_{0,x}^2 + f_{0,y}^2)^{0.5}$ (in radians). At large scales, slope steepness can be used as an isotropic variant of *shaded relief* map image, which is used to highlight structure within a DEM [25, 7]. Illustrators also use this measure for highlighting steep variations in the subject (fig. 2). Figures 4 (in yellow) and 5(b) illustrate this effect as computed by our system.

Slope Aspect (A0) is a measure of the direction that a slope faces. It identifies the steepest downslope direction at a location on a surface. Geometrically, it is the angular distance (counted clockwise) between the directions of a fixed point and of the slope. This angle ($-90 \leq A0 \leq 90$) can be represented through first partial derivatives using the formula [25]: $A0 = \arctan(f_{0,y}/f_{0,x})$ (in radians). Artistically, slope aspect is useful for indicating hatching marks as clusters of parallel lines following a particular drawing direction (fig. 5(c)).

Mean Curvature (H) The mean curvature of a surface at a point is one half the sum of the principal curvatures at that point. Geomorphologically, negative H values describe mean-concave land forms, while positive H values refer to mean-convex ones. Technically, it can be represented through first, second partial derivatives using the formula [7, 25, 33]:

$$H = -\frac{(1+f_{0,y}^2)f_{0,xx} - 2f_{0,x}f_{0,y}f_{0,xy} + (1+f_{0,x}^2)f_{0,yy}}{2(1+f_{0,x}^2+f_{0,y}^2)^{3/2}} \quad (5)$$

This formula provides good discrete approximations of mean curvature, allowing proper detection of convex and concave formations across the mesh (figs. 4 and 5(d,e)).

4. Pen strokes

Figure 5 (a-e) illustrate the case in which thresholded feature edges are rendered as single thickness lines, revealing specific shape measures. In traditional production of precise drawings, this corresponds to the stage in which the illustrator has accurately measured and lightly delineated the shape characteristics across the surface of the subject (sec. 1). We can improve the 3D perception of the shape measures by adjusting the thickness of the strokes and by rendering them with different marking styles.

4.1. Thickness adjustment

Our approach is to create, for each edge to be displayed, a 3D ribbon (in world-space) by extruding its vertices (a,b) in the direction of their normals (n_a,n_b), resulting in a new pair of vertices $(a',b') = (a+n_a\rho, b+n_b\rho)$, with the amount of extrusion given by ρ. Intuitively, this amount of extrusion corresponds to the amount of ink placed at the stroke (fig. 7, left). In traditional illustration, the ink flow is precisely controlled to make very heavy to very fine lines in order to depict regions of high and low curvature, respectively [4, 31]. We therefore define $\rho = H_{ab}\phi_{ab}$, where H_{ab} is the mean curvature estimated at edge *ab* (eq. 5), and ϕ_{ab} is a user-defined positive scaling factor, which gives further control over the stroke thickness adjustment process. Note that for values of $H_{ab} < 0$ (concave regions), $\rho_{ab} = H_{ab}(-1.0)\phi_{ab}$ to guarantee that the extrusion from (a,b) moves in the outward direction of the normals (n_a,n_b).

4.2. Ink marking styles

Two styles are provided: filled and serrated. **Filled marks** are implemented by simply rendering the stroke defined by

the ribbon $\{a,b,b',a'\}$ in black (fig. 7, left). **Serrated marks** are modeled by distributing marks with different directions and lengths within the ribbon $\{a,b,a',b'\}$ (fig. 7, right). As shown in Figure 6 and in the algorithm below, a stroke s with vertices (p,q) is defined at each parametric distance t along edge ab. The algorithm for generating serrate strokes is as follows:

SERRATED-STROKE$(a,b,a',b',res,l_1,l_2,d_1,d_2)$
1 $res \leftarrow res\ (|ab|/l_{max})$
2 **for** $t \leftarrow 0$ *to* 1, $step \leftarrow 1/res$
3 **do** $(\delta_1,\delta_2) \leftarrow (l_1/2res, l_2/2res)$
4 $(t_1,t_2) = (t-\delta_1,\ t+\delta_1)$
5 $(t'_1,t'_2) = (t-\delta_2,\ t+\delta_2)$
6 $v = a + unif(t_1,t_2)(b-a)$
7 $v' = a' + unif(t'_1,t'_2)(b'-a')$
8 $p = v + d_1(v'-v)$
9 $q = p + d_2(v'-v)$
10 $DrawLine(p,q)$

In *SerratedStroke()*, the variable *res* corresponds to the user-defined number of marks to be placed along edge *ab*. Line 1 adjusts *res* considering the length of edge *ab* and the maximum edge length l_{max} in the mesh (both lengths related to the current view location *eye*). Lines 3-5 compute how much stroke s (fig. 6) can vary to the left and right of t, with $(l_1,l_2) \in [0,1]$. Lines 6-7 compute the stroke coordinates (v,v'). The function $unif(i,j) = i + (j-i)rand()$ returns a uniformly distributed real number between i and j, $(i < j)$, and $rand()$ returns a pseudo-random real number uniformly distributed between 0 and 1. Notice that v and v' can be anywhere along (t_1,t_2) and (t'_1,t'_2), respectively, thus resulting in orthogonal or diagonal strokes along *ab*. Finally, lines 8-9 adjusts the length of s, with $0 \le d_1 < d_2 \le 1$.

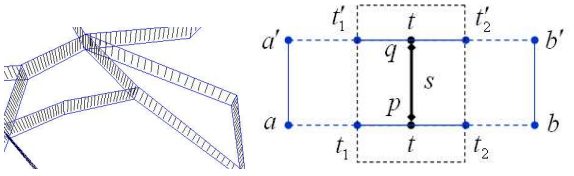

Figure 6: *Left: serrated marks placed at mesh edges. Right: A single mark s, defined by vertices (p,q), can vary in location and length within (t_1,t_2,t'_1,t'_2).*

Figure 7 (right), shows an example of serrated marks. Notice that the edge $a'b'$ is not displayed, only edge *ab* and the serrated marks s along it. Also notice the fur-like effect on the bunny due to a reduction on the amount of marks (variable *res* in the above algorithm) and to an increase in the thickness of the strokes by adjusting variable ϕ_{ab} (subsec. 4.1).

4.3. Ink distribution effects

Ink distribution is an intuitive way of referring to stroke thickness distribution across the mesh. As described in section 4, the thickness of every front-facing interior edge *ab* (sec. 2) is determined by extruding *ab* in world-space by some ρ amount in (n_a,n_b). This results in larger perceived stroke thickness as (n_a,n_b) becomes more orthogonal to the view vector. Different visual effects can be achieved by simply adjusting the amount of extrusion ρ. We implemented two effects dependent on the parametric view depth distance d_{ab} of each edge *ab*, given by $d_{ab} = 0.5(\ (|\vec{v}| + |\vec{w}| - 2z_{min})\ /\ (z_{max} - z_{min})\ )$, where $(\vec{v},\vec{w}) = (a-eye, b-eye)$, and (z_{min},z_{max}) are the minimum and maximum view depth distances, respectively. The **first effect** is given by adjusting $\rho_{ab} = \rho_{ab}(1.0 - d_{ab})$, which slightly increases the thickness of strokes as they gets closer to the viewer. The overall resulting effect is a more balanced distribution of stroke thickness across the mesh (fig. 8, Hand). The **second effect** is given by $\rho_{ab} = \rho_{ab}(2.0 - d_{ab})$, where strokes farther from the viewer will have their thickness scaled down, resulting in better depth cues (fig. 8, Mt. St. Helens).

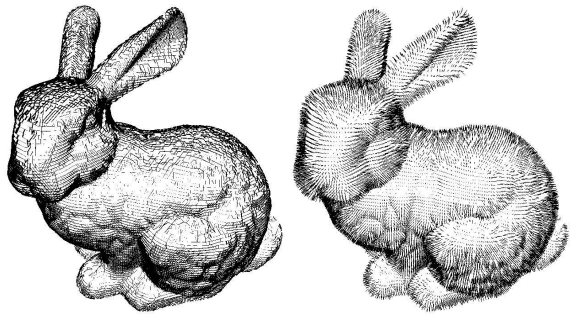

Figure 7: ***Stanford bunny*** *(built from laser scans), with slope steepness $GA(0.81,1.0)$ and positive mean curvature $+H(0.57,1.0)$, rendered with filled (left) and serrated (right) pen marks (res = 15, see subsec. 4.2).* Model source: Stanford University Computer Graphics Lab.

4.4. Rendering

In our system, rendering is performed by OpenGL calls. For thickness adjustment (subsec. 4.1) notice that (a',b') is defined in 3D, and then projected by OpenGL. This means that the thickness of a line in the image space is defined by the projection of the normal of the vertex on the picture plane. The visibility computation uses the Z-buffer, where triangles are first rendered in background color (white) and stroke ribbons are then rendered in different styles. Since the Z-buffer computation is necessarily imprecise, some edges *ab* may get clipped or hidden by the mesh. Our approach to this problem is to translate edges *ab* some small amount into the direction of the normals (n_a,n_b), prior to stroke thickness adjustment.

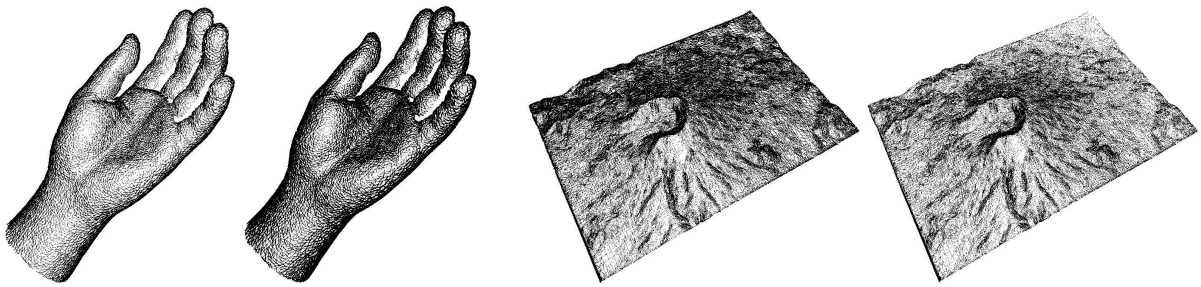

Figure 8: *Two visual effects of stroke thickness distribution:* ***Hand*** *with normal stroke thickness distribution (left) and with increase of thickness as strokes gets closer to the viewer (right).* ***Mount St. Helens*** *with normal stroke thickness distribution (left) and with strokes farther from the viewer having their thickness scaled down (right).* Model provided by Delcam at www.millwizard.com/Main.htm

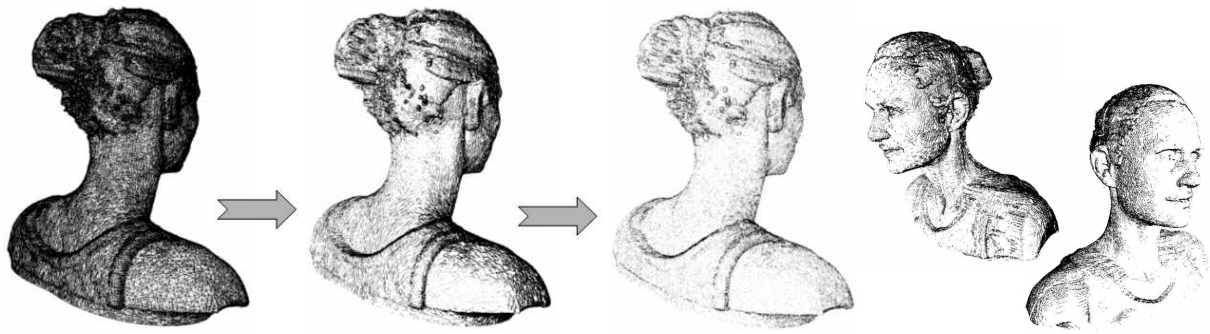

Figure 9: ***Female head*** *(built from laser scans);* Left: *starting with the wire-frame mesh, we place filled strokes at slope steepness* $GA(0.7, 1.0)$*, and positive mean curvature* $+H(0.47, 1.0)$*; Notice how various anatomic and hair features are revealed. Next, we replace filled with serrated marks, resulting in a soft stippling effect.* Rightmost images: *Side views of model rendered with filled strokes using the exact setup of image right to the mesh.* Model source: Cyberware.

Model	$\triangle$s	*edges*	*preproc.*	*render*
Hand	26,373	13,339	2	0.25
Mt St Helens	65,932	33,822	4	1
Bunny	69,451	36,742	4	1
Killeroo	92,092	46,351	5	1
Rihard Jakopič	108,507	56,610	6	1
Broken Adze	118,676	59,339	6	1
Female Head	154,650	78,560	8	1
Gargoyle	206,982	103,740	11	2
Hip	265,084	136,099	14	2
Igea Artifact	268,686	136,483	14	2
Fossil Skull	284,458	146,123	15	4
Preformed Adze	401,060	200,529	22	4
Hammerstone	725,828	362,915	235	7

Table 1: *Average times (in seconds) for pre-processing and rendering the models presented in this paper.*

5. Results and discussion

Our system achieved fast computation rates including pre-processing (building the edge-buffer and calculating shape measures) and rendering (automatic stroke thickness adjustment and interactive pen marking). Figures 1 and 7-16 show results using our system in 13 meshes (Table 1). Running times were gathered from a 2.65 GHz Pentium IV with OpenGL/ATI Radeon 9700 graphics. We recommend printing the results at 200dpi on a 600dpi HP LaserJet, or better. At the figure caption of each result, we described the effects achieved for different shape measures selection, thresholds, and pen-marking styles. Concerning the user-defined extra thickness scale ϕ_{ab} (subsec. 4.1), we use values between 0.005 and 0.01 for all the results presented here. The measured frame rate does provide the user with an acceptable level of interactivity for exploring and illustrating various shape measures on mesh models. The user is able to quickly select and threshold clusters of feature edges related to certain shape measures, to adjust the parameters of stroke styles and thickness distribution effects. The system has good temporal rendering coherence with only some temporal aliasing occurring at the silhouette edges as new strokes are added based on the silhouette extraction and automatic thickness adjustment. We also noted three important points:

Input Mesh: In our system, any triangle mesh can serve as input. However, we noted two important constraints. First,

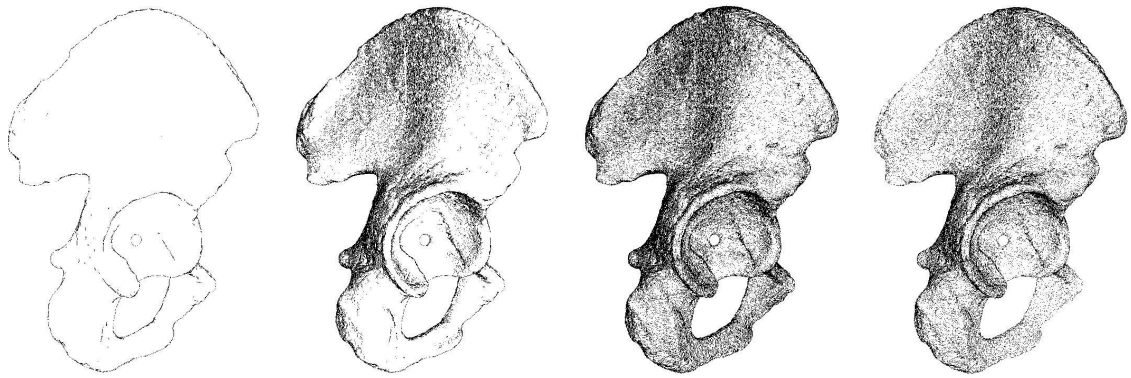

Figure 10: *Using our system for medical illustration of a* hip *model (built from laser scans) (a) First we render silhouettes; Notice the thickness variation as a function of the curvature; (b) Concave formations are then revealed by placing filled strokes in locations with negative mean curvature $-H(-0.459, -0.001)$; (c) Creases are then delineated with $D(155, 170)$, and convex formations are revealed by placing filled strokes in locations where positive mean curvature $+H(0.56, 1.0)$; (d) Finally, the view-depth effect is applied, improving the depth perception of the hip. Compare results with the real precise drawing of a hip in Figure 2.* Model source: Cyberware.

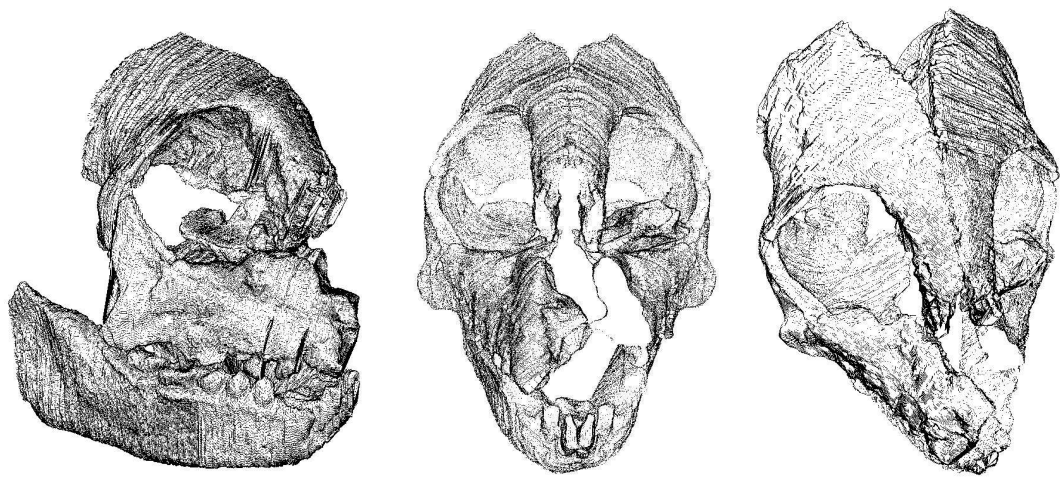

Figure 11: *Using our system for archaeological illustration of primate specimen* ***Homunculus Patagonicus*** *(built from laser scans); Left: covering the whole model with filled strokes; Middle: Replacing filled with serrated marks and applying the view-depth effect; Notice the improvement in depth perception; Right: placing filled marks to reveal slope steepness and serrated marks for slope aspect, keeping view-depth effect on.* Model source: Smithsonian Bio Visualization Lab.

our method can be successfully used on models defined by very dense meshes. Second, certain types of models made of many flat-surfaces (i.e. modern buildings), would either be under-tessellated for our algorithm, or be composed of mostly regular 'quads', either of which could produce unwanted artifacts in placement of the strokes. For instance, in Figure 8, Mount St. Helens comprises many visible quads in the final rendering. These quads may have an effect on the aesthetic of the image, but nevertheless the shape is revealed adequately.

Level-of-Detail: Our system allows for some level of control when the object is viewed from far away and in close-up. The user can adjust the style parameters of the pen marks (subsec. 4.2) and also select depth-dependent ink distribution effects (subsec 4.3). It is important to also consider other level-of-details controls, including measures for omitting or adding edges while preserving local shape measures and rendering effects.

Line Directions: Our goal was to experiment with morphometric variables, with line direction given by slope aspect, which results in the shape being properly revealed. Observe, for instance, Figures 1 and 12(right) (the Artist's mask) has

edges aligned along the direction of the nose. The killeroo (Figure 16), has lines following curvature on its back that reveal shape. The broken preformed adze of Figure 15 also has short directional strokes revealing the irregularity of rock formations. A larger selection of line directions calculation schemes may improve visual results even further, including principal directions of curvature [21, 22, 23, 11] and other types of line directions found in geomorphology [24, 25, 7, 26].

6. Conclusion and future work

We have developed a 3D NPR system that reproduces the traditional technique of precise drawings, where short pen marks are used to depict the geometric forms that give 3D objects their characteristic shape. Our system utilizes techniques from geomorphology to calculate shape measures across the surface of the models. Pen strokes are then modeled and rendered at each edge on the model with automatic thickness adjustment and interactive control over pen marking styles. Our system demonstrates that precise drawing effectively illustrates complex mesh models in a simple, informative manner that is valuable, especially for illustrating regions of interest while maintaining shape perception. Initial feedback from illustrators is very positive. They are enthusiastic about the usefulness of the system for generating images, in particular for natural science subjects. It was observed that illustrators usually spend far more time to generate traditional precise drawings for similar subjects presented in this paper.

We plan to extend our work to improve the interactivity of the system and compare the performance to other NPR stroke-based renderers to assess the effectiveness of using a precise drawing system, including a more formal evaluation of the system working together with art and science illustrators. Furthermore, we will continue to explore additional shape measure techniques and pen marking styles.

Acknowledgments

We would like to thank Peter A. Shary, Beei-Huan Chao, Mohammad A. Rajabi, and Ali El-Zafrany for their discussions on geomorphology and related numerical methods. Many thanks to Emily S. Damstra and Humberto Costa Sousa for providing real precise drawings and for their valuable input on traditional illustration techniques. We also thank the researchers, labs. and companies who made the models available for use in this paper. Finally we thank Patricia Rebolo Medici for her useful advice and the anonymous reviewers for their substantive and valuable comments and suggestions. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

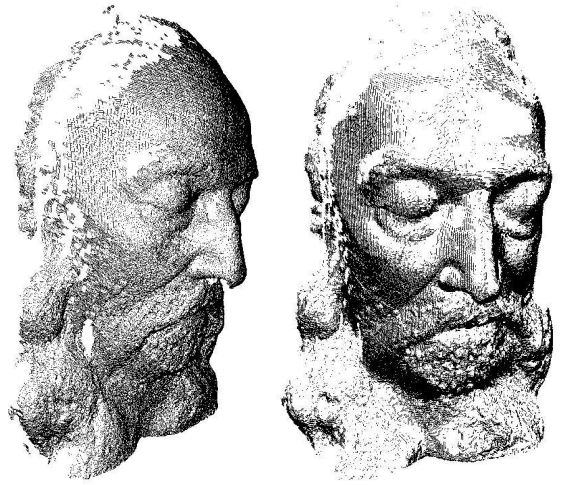

Figure 12: ***Mask*** *of artist Rihard Jakopič (built from range images); Left: Covering the model with filled strokes on locations of negative mean curvature* $-H(-1.0,-0.01)$ *and positive mean curvature* $+H(0.001,1.0)$*; Right: placing filled strokes with slope steepness* $GA(0.8,1.0)$ *and slope aspect* $A0(0.9,1.0)$*. Notice how facial features are properly revealed. Compare with real precise drawing of a man's face in Figure 2.* Model provided by Danijel Skočaj, University of Ljubljana Computer Vision Lab [34].

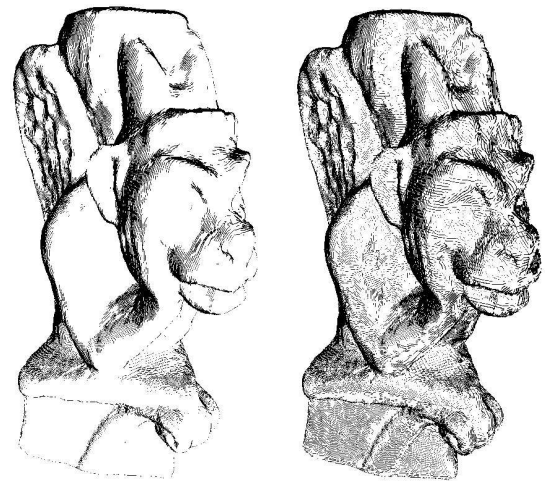

Figure 13: ***Gargoyle*** *(built from range images) Left: Initially we place filled strokes for slope steepness measure only. Right: Placing strokes for all other shape measures, with emphasis on slope aspect. Notice how the directional variation of strokes reveal the curved shapes of the statue.* Model source: Rich Pito, University of Pennsylvania GRASP Lab.

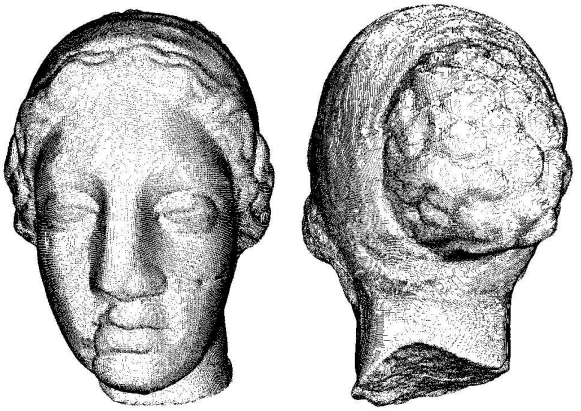

Figure 14: ***Igea artifact*** *(built from laser scans); Front head: placing filled strokes with slope steepness* $GA(0.81, 1.0)$*, and positive mean curvature* $H(0.57, 1.0)$*; Back head: placing serrated strokes with 3 marks per edge and with view depth effect, revealing shape measures of slope steepness* $GA(0.73, 1.0)$*, and negative mean curvature* $-H(-0.5, -0.48)$*. Notice that hair and anatomic features are clearly revealed, which has special significance for this model in particular, given that the original artifact has various degrees of erosion.* Model source: Cyberware.

Figure 15: ***Killeroo*** *with filled strokes and slope steepness, aspect and positive mean curvature.* Model provided by headus.com.au

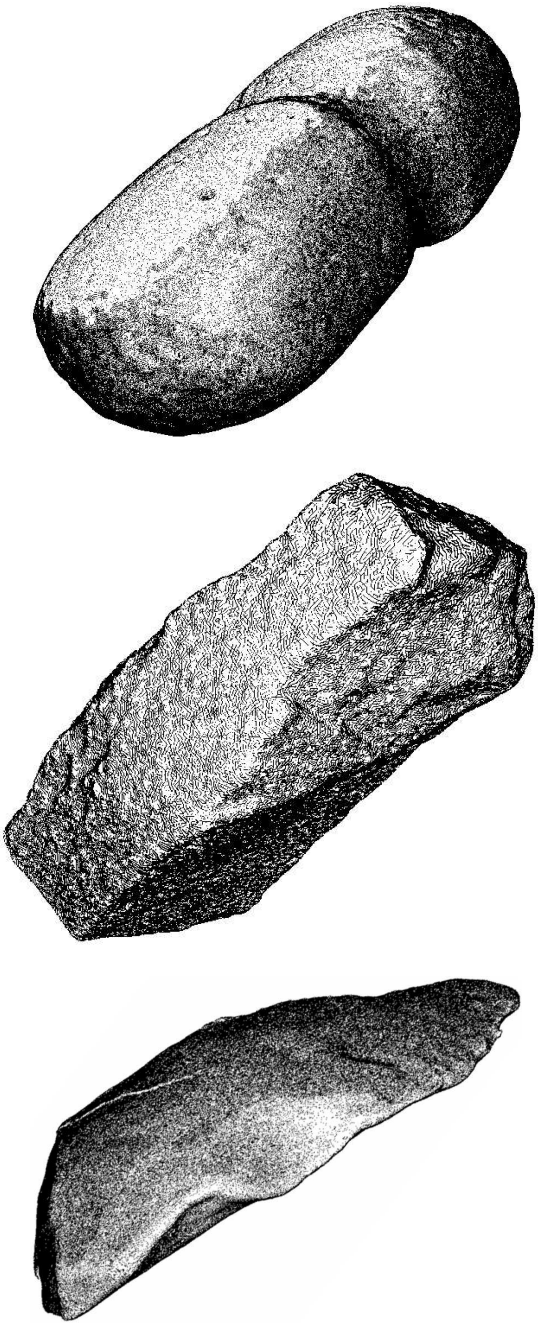

Figure 16: *Archeological illustrations of artifacts built from laser scans.* Top: ***large oval hammerstone*** *with filled strokes revealing slope steepness and few locations with negative mean curvature.* Middle: ***broken preformed adze*** *rendered with filled strokes for slope aspect; Notice the the variations of stroke directions, revealing the irregularity of the shape structure.* Bottom: ***preformed adze*** *with filled strokes for positive and negative mean curvature and by applying the effect of increasing the stroke thickness as they gets closer to the viewer. Compare all three results with real precise drawing of the artifact in Figure 2.* Models provided by Dr. Jeff Clark, North Dakota State University Archaeology Technologies Lab.

References

1. E. Hodges, *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, (1989).
2. C. Papp, *Scientific Illustration: Theory and Practice*. W.M. C. Brown Company Publishers, (1968).
3. G. Simmons, *The Technical Pen*. Watson-Guptill Publications, (1992).
4. P. Rawson, *Drawing*. University of Pennsylvania Press, (1987).
5. I. Cornwall, *Bones for the Archeologist*. Phoenix House, London, (1956).
6. J. Buchanan and M. Sousa, "The edge buffer: a data structure for easy silhouette rendering", in *Proc. of NPAR '00*, pp. 39–42, (2000).
7. P. Shary, L. Sharaya, and A. Mitusov, "Fundamental quantitative methods of land surface analysis", *Geoderma*, **107**(1-2), pp. 1–32 (2002).
8. G. Winkenbach and D. H. Salesin, "Computer-generated pen-and-ink illustration", in *Proc. of SIGGRAPH '94*, pp. 91–100, (1994).
9. G. Elber, "Interactive line art rendering of freeform surfaces", in *Proc. of Eurographics '99*, pp. 1–12, (1999).
10. O. Deussen and T. Strothotte, "Computer-generated pen-and-ink illustration of trees", in *Proc. of SIGGRAPH '00*, pp. 13–18, (2000).
11. E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching", in *Proc. of SIGGRAPH '01*, pp. 581–586, (2001).
12. O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte, "Floating points: A method for computing stipple drawings", in *Proc. of Eurographics '00*, pp. 40–51, (2000).
13. A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen, "Non-photorealistic volume rendering using stippling techniques", in *Proc. of IEEE Visualization '02*, pp. 211–218, (2002).
14. A. Secord, "Random marks on paper: Non-photorealistic rendering with small primitives", Master's thesis, Department of Computer Science, University of British Columbia, (Oct. 2002).
15. O. E. M. Pastor, B. Freudenberg, and T. Strotthote, "Animated, real-time stippling", *IEEE Computer Graphics and Applications (Special Issue on Non-Photorealistic Rendering), July/August*, (2003).
16. A. Gooch, B. Gooch, P. Shirley, and E. Cohen, "A non-photorealistic lighting model for automatic technical illustration", in *Proc. of SIGGRAPH '98*, pp. 447–452, (1998).
17. G. Gooch, P.-P. J. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld, "Interactive technical illustration", in *Proc. of the Conference on the 1999 Symposium on Interactive 3D Graphics*, pp. 31–38, (1999).
18. J. Hamel, *Alternative Lighting Methods for Comuter Generated Line Drawings*. PhD thesis, University of Magdeburg, (2000).
19. E. Boix, *Approximation lineaire des surfaces de* $\Re^3$ *et applications*. PhD thesis, Ecole Polytechnique, France, (1995).
20. P. Yuen, N. Khalili, and F. Mokhtarian, "Curvature estimation on smoothed 3-d meshes", in *Proc. of British Machine Vision Conference '99*, pp. 133–142, (1999).
21. A. Girshick, V. Interrante, S. Haker, and T. S. Lemone, "Line direction matters: An argument for the use of principal directions in 3d line drawings", in *Proc. of NPAR '00*, pp. 43–52, (2000).
22. A. Hertzmann and D. Zorin, "Illustrating smooth surfaces", in *Proc. of SIGGRAPH '00*, pp. 517–526, (2000).
23. C. Rossl, L. Kobbelt, and H. Seidel, "Line art rendering of triangulated surfaces using discrete lines of curvature", in *Proc. of WSCG '00*, pp. 168–175, (2000).
24. I. Evans, "General geomorphometry, derivatives of altitude and descriptive statistics", *Spatial analysis in geomorphology*, pp. 17–90 (1972).
25. H. Mitasova and J. Hofierka, "Interpolation by regularized spline with tension: II. application to terrain modeling and surface geometry analysis", *Mathematical Geology*, **25**, pp. 657–669 (1993).
26. J. Wood, *The Geomorphological Characterisation of Digital Elevation Models*. PhD thesis, University of Leicester, UK, (1996).
27. M. Kaplan, B. Gooch, and E. Cohen, "Interactive artistic rendering", in *Proc. of NPAR '00*, pp. 67–74, (2000).
28. J. Northrup and L. Markosian, "Artistic silhouettes: A hybrid approach", in *Proc. of NPAR '00*, pp. 31–37, (2000).
29. L. Markosian, B. J. Meier, M. A. Kowalski, L. S. Holden, J. D. Northrup, and J. F. Hughes, "Art-based rendering with continuous levels of detail", in *Proc. of NPAR '00*, pp. 59–64, (2000).
30. L. Lapidus and G. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*. John Wiley Sons, (1982).
31. W. Crane, *Line Form*. George Bell Sons, London, (1900).
32. J. Krcho, "Morphometric analysis of relief on the basis of geometric aspect of field theory", *Geographica Universitatis Comenianae, Geographico-Physica*, pp. 7–233 (1973).
33. A. Gray, "The gaussian and mean curvatures", *§16.5 in Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed., Boca Raton, FL, CRC Press*, pp. 373–380 (1997).
34. F. Solina, "Internet based art installations", *Informatica*, **24**(4), pp. 459–466 (2000).