# Depicting Shape Features with Directional Strokes and Spotlighting

Mario Costa Sousa Faramarz Samavati Meru Brunn

University of Calgary
Department of Computer Science
Calgary, AB, Canada T2N 1N4
{mario,samavati,brunn}@cpsc.ucalgary.ca

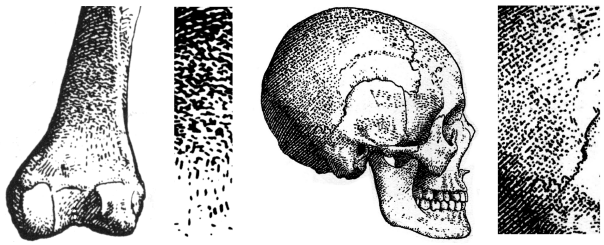
## Abstract

*This paper presents a new algorithm and technique for rendering triangular surfaces in pen-and-ink edge-based strokes. Our technique integrates two very important illustration strategies for depicting shape features: selection of drawing direction and the use of light. Drawing direction is given by four stroke directional fields. For lighting, we introduce the idea of "spotlight silhouettes" for fast illumination computation, with target tone matched by adaptive stroke length adjustment. Stroke style is achieved by path perturbation and noise-based weight control. Our technique also allows visual effects of reverse tone values and depth cueing. Examples with models from anatomy and archeology demonstrate the capabilities of our system.*

## 1. Introduction

Creating convincing impressions of 3D forms on paper is a demanding perceptual challenge for traditional artists in the area of scientific illustration. The accurate depiction of form is essential in making the drawings meaningful both aesthetically and scientifically. Additionally, the final rendering of the subjects must convincingly depict solid masses in space, show their various structural conditions, and suggest their different surface characteristics. To achieve these goals in traditional drawings, illustrators typically use two strategies: lines arranged following particular surface directions and the effect of light cast upon the surface, showing contrast and attenuation variations [7, 16].

This paper presents a new algorithm and technique that implements the two illustration strategies above, directly rendering triangle-mesh surfaces in pen-and-ink style strokes. Our goal is to reproduce a particular traditional technique where the strokes used are short, straight (sometimes showing very small hand gesture perturbations), and with very little width variation. Tone values are mainly matched by adjusting the length of the strokes. This technique can evoke powerful expressive meaning and is often used in the context of precise artistic drawings and science illustrations [18, 35] (see figure below). We also consider other important goals including interactive control over viewing and lighting conditions and a balanced level of interactivity, giving the user some degree of artistic freedom.



Our algorithm follows an edge-based stroke placement approach. At each edge of the mesh, a stroke is mapped directly to one of four possible directional fields. For lighting, we introduce the idea of "spotlight silhouettes," which consider only the regions on the mesh that are visible to the spotlight cone, resulting in fast illumination computations. Target tone is matched by adjusting the stroke length adaptively from parameters computed directly from the mesh, without the need for tone value charts or pre-generated stroke textures. Each stroke can be rendered as straight single width lines or stylized by a noise-based approach for weight distribution and path perturbation. Our technique also allows the visual effects of reverse tone values and depth cueing. Examples with models from technical and scientific subjects of anatomy and archeology demonstrate the capabilities of our system.

## 2. Related work

Our work is related to four main research directions in NPR: short ink marks-based illustrations, lighting models, stroke directional fields, and ink stroke stylization.

**1) Short ink marks-based illustrations:** Winkenbach and Salesin [34] introduced *stroke textures* which procedurally accumulate strokes for patterns made with short marks. Praun et al. [25] extended this approach introducing *tonal art maps* which organize pre-rendered strokes as a sequence of mip-mapped images. Some researchers have focused on the geometric relation between the stipple marks [3, 15]. A particle-based distribution approach was used by Deussen and Strothotte [4] to place marks for tree foliage rendering. A voxel-based approach has been proposed by Lu et al. [19] with an interactive direct volume stippling illustration system. Secord et al. [29] develop a fast probabilistic method that places small arbitrarily-shaped primitives, including stippling. More recently, Pastor et al. [23] present an approach where stipple particles are attached to the surface of the model, using a point hierarchy to control the stipple shading density. In [32], a system is described for rendering large, detailed meshes, using extracted geometric shape features to guide the placement of strokes on each mesh edge.

We extended the work in [32] to include stroke directional fields and the evaluation of "spotlight silhouettes" directly at each edge in the model. All of these extensions are placed into a new version of the edge-buffer data structure [2]. Also, we match target tones by adaptively adjusting the stroke length directly at each edge. Our method also allows for efficient visualizations of tone value reversals across the model.

**2) NPR lighting**: Compared to photorealistic rendering algorithms, lighting in NPR does not generally attempt to simulate or approximate a physical model of light-surface interaction. Instead, the goal is often to extract additional information from the scene to help generate a stylized rendering. Martin and Torres [20] use multiple interactively-placed diffuse or specular *virtual lights* to generate silhouette outlines and shape lines for cartoon animation-style renderings of 3D objects. Akers et al. [1] interactively composite multiple differently-lit images of an object to reproduce the artistic technique of using different lighting conditions for different feature areas of an object. Hall [12] introduces a technique to visualize the shape of precisely colored 3D surfaces without color distortion from lighting. Gooch et al. [8] (extended in [9]) introduce an NPR lighting model that mimics the artistic style of traditional technical illustrations. Hamel [13] presents a new lighting model, derived from traditional scientific illustration principles, that allows both photorealistic and artistic illumination techniques such as rim shadows, curvature shading and transparency.

Our approach is similar in principle to the idea presented in [20]. In our work, we first extract all mesh regions visible to the spotlight rays and then calculate the light intensity at each edge in the edge-buffer. The reason we select a spotlight is because (1) it provides coverage of selective areas in the model with (2) attenuated light intensity. Both aspects are very useful to create different visualizations of the shape features.

**3) Directional strokes**: In mesh-based NPR systems, the focus has been mainly on estimating principal direction of curvature (usually at each mesh vertex) to guide the stroke placement process [6, 14, 25, 27].

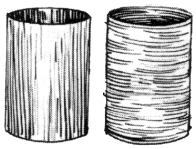
In our system we estimate four types of stroke directional fields, providing alternate visualizations of shape features. We use two calculation methods. The first method estimates the principal directions of curvature at each vertex with subsequent edge-buffer updates. The second method estimates two directional fields directly at each edge by considering the normals of the two triangles adjacent to it.

**4) Ink stroke stylization**: Stroke stylization helps both in evaluating surface shape and achieving expressive effect typical of artistic drawings. Stroke textures [34] use slight perturbations of the stroke's curvature, length, and direction to give a less mechanical look to pen-and-ink renderings. Kaplan et al. [17] vary line thickness of silhouette strokes based on light intensity. Northrup and Markosian [22] allow the user to specify a combination of various stylization such as tapering, flaring, and wiggles to achieve a desired look for silhouette edge strokes. In [32], surface curvature metrics are used to automatically adjust stroke thickness. The strokes are then drawn with different pen marking styles.

Our stroke stylizer combines path perturbation [28] with a new way of distributing specific weight patterns using Perlin noise [24].

## 3. Illustration techniques

In this section, we briefly review two key principles of traditional illustration used together for depicting shape features: the selection of drawing direction and the use of light; much more detailed studies can be found in a number of texts [7, 11, 16, 35].



**Drawing direction** In addition to drawing the silhouettes and boundaries of the model, groups of parallel lines should be arranged in a hatching style flowing in consistent directions to produce optical tones which help interpret the form. There are no strict rules for selecting a particular direction. Many artists of very different aesthetic goals have used various line directions as an effective means for interpreting form in original ways [7, 26, 31]. It is generally accepted that the choice of the direction of the strokes should be one that supports issues such as compositional or expressive ones. Usually, hatched lines drawn as moving around a form or as riding

upon its surface undulations tell us more about a shape's form than lines that go in other directions on it. Therefore, a common strategy is to establish two main directional fields, one orthogonal to the other. In the cylinder drawings [7] (previous page), the lines of the left cylinder direct our eyes along the length of the body suggesting a more forceful move in a direction, while those in the right cylinder move around it, telling us a little more about its volume [7].

**Light on form** The placement of lines in particular directions is not the only way to describe form. Light on form is a very important and complementary technique for conveying shape features and overall form of subjects. Tone values can do more than establish local tones, build forms and suggest light. They can help the artist to add expressiveness to the subject [7, 11, 16]. Light not only can be made to explain and unify forms, it can provide important visual and expressive meanings. Without a study of the subject's tonalities, important structural clues are missed. Traditional illustration commonly tend to have a "dramatic" light source (i.e. light off to one side of the model), to experiment with tone value studies. Also, abrupt changes in value are used to suggest angular forms and gradual changes for rounded forms.

## 4. Algorithm overview

Our data structures and algorithms were selected to provide interactive rates for visualization of complex meshes. For efficiency, our technique uses two stages: we separate the single-pass pre-processing from the user-adjustable interactive controls over the final rendering. In the preprocessing phase, we construct an edge buffer [2] from the mesh and simultaneously compute and store the four stroke direction fields at each edge (sec. 5). During run-time spotlight silhouettes are extracted and illumination values are computed at each edge (sec. 6). Strokes are then rendered from the direction and illumination values stored with each edge. To match the target tones from the attenuated spotlighting, the length of each stroke is adjusted proportionally, based on the stored intensity (sec. 6). Finally, the stroke is stylized with path perturbation and ink weight distribution, imparting a less uniform look to strokes rendered on regular meshes (sec. 7).

## 5. Stroke directional fields

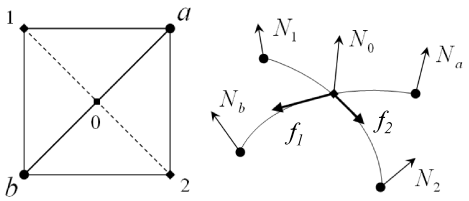
Based on the techniques of traditional illustration, we consider two orthogonal directions for strokes (sec. 3). In order to achieve an efficient technique, these directions are pre-computed and saved in the data structure. Consequently, in our case, the directions are associated with the edges of the 3D mesh.

The important problem is to determine the appropriate directions from the given 3D mesh. Although there is no clear artistic rule for having a unique stroke direction, it is possible to find reasonable solutions that can depict the shape of the object. We consider two solutions.

**Method 1: principal directions of curvature** Principal directions of curvature are classical geometric measures that can visualize the local shape of the surface. The maximal curvature $c_1$ and minimal curvature $c_2$ are called principal curvatures, the associated tangent vectors $e_1$ and $e_2$ are called principal directions. Therefore, moving along $e_1$ produces the highest and moving along $e_2$ causes the lowest variation for the normal at the surface. These two directions are mutually orthogonal to each other and form an orthonormal basis for the tangent space. Although there is a rich theory in differential geometry about curvature of differentiable surfaces, the resulting methods are not appropriate for 3D meshes as non-differentiable surfaces. In contrast, methods from discrete differential geometry are preferred for these kind of objects [5, 21, 33]. In this approach, principal curvatures and directions are estimated by evaluating the eigenvalues and eigenvectors of a $3 \times 3$ matrix that is closely related to the tensor of curvature. This matrix can be estimated by difference of the normals at a vertex and its neighbors [5, 21, 33]. For an edge-based data structure such as the edge-buffer, it is beneficial if the directions are assigned directly to the edges. We use the average of $e_1$ at two adjacent vertices of the edge as the first direction of the edge. Analogously, the average of $e_2$ at two adjacent vertices is used as the second direction.

**Method 2: simple tangent space directions** Although the principal directions $e_1$ and $e_2$ are good selections for depicting the shape features, they are not the only ones. We can achieve some degree of depiction by using any directional field. For example, the wire-frame representation of a surface is a possible result of using "edge vector" as the stroke direction. In this case, the second direction can be created as a vector that is orthogonal to the edge and also to the normal of the surface at the edge. Although calculation of these directions is very simple, they can not produce satisfactory results for most applications. Based on the curvature around the edge, we correct these directions to enhance the level of depiction while the resulting method remains simple. In order to perform this correction, we estimate two specific curvature values at the midpoint of edges. Consider the figure on the next page, where $(a, b)$ is the current edge, 0 is the midpoint, 1 and 2 are two remaining vertices of the triangles that have $(a, b)$ as a shared edge. In this simple neighborhood, we can measure the curvature in the direction $(a, b)$ and $(1, 2)$ by computing the variation of normals. Cross product can measure these variations as $\xi_{ab} = N_a \times N_b$ and $\xi_{12} = N_1 \times N_2$.

We pick the direction of the largest vector ($\xi_{ab}$ and $\xi_{12}$) as the first direction denoted by $f_1$. We estimate $N_0$, the normal vector at the "phantom" vertex 0 (midpoint of edge $ab$), by $N_0 = 0.5(N_a + N_b)$. Note that $f_1$ lies in the tangent plane at 0. Therefore, we can find the second direction by $f_2 = f_1 \times N_0$. These vectors form an orthonormal ba-




sis for the tangent plane. Vector $f_2$ represents a direction that the normal varies rapidly and vector $f_1$ represents a direction that the normal varies slowly. The effect of the cross product causes the change of the order. Although they are not principal directions of the curvature, they can depict shape features pleasantly in a very simple and inexpensive way. Figure 3 show the effectiveness of these two directions.

## 6. Spotlight illumination

In our system, the evaluation of a light source is based on the traditional illustration technique of first outlining regions in light to guide the subsequent placement of pen strokes for tone matching [16, 26, 30]. We select a spotlight because it provides coverage of selective areas in the model with attenuated light intensity. We observed that these aspects are very useful to create different visualizations of the shape features. Our approach consists of extracting all mesh regions visible to the spotlight rays, discarding faces not visible to them and calculating the light intensity at each edge midpoint of the remaining faces visible to the light. This process requires only simple visibility tests and updates in the edge-buffer, resulting in efficient illumination computation and allowing the user to visualize the "light silhouettes" as in traditional illustrations.

**Light silhouettes** The key to our spotlight rendering is the edge-buffer [2], which is updated for every new camera view and spotlight parameters. We extend the edge buffer by including $I_0$, (the spotlight intensity at the middle point of the edge) at each node in the edge buffer. We first reset all the edge buffer bit fields to zero. Then, for each triangle of the mesh, we determine whether it is front or back facing in relation to the camera, and update its associated edges, setting their initial light intensity to $I_0 = -1.0$. Algorithm $UpdateEB()$ below performs this update, setting the bit fields $(F, Fa)$ to indicate a front-facing silhouette edge, or $(B, Ba)$ for a back-facing one [2]. We denote $EB(i, j)$ as referencing a $node$ from the adjacency list at $EB[i]$ where

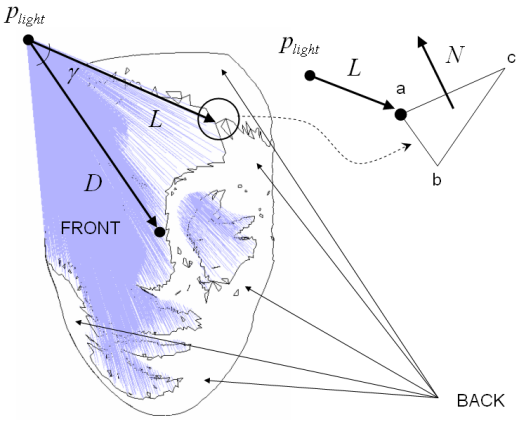



**Figure 1. Key elements of $LightSilhouettes()$, resulting in mesh regions that are visible (FRONT) and invisible (BACK) to the spot light. Notice the silhouette lines formed by the spotlight rays.**

$node.id = j$. Refer to Buchanan and Sousa [2] for more details on the bit-fields update.

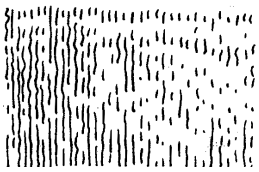
```
UPDATEEB(i, j, facing, I_L)
1  if facing = FRONT
2    then update bit fields EB(i, j).(F, Fa)
3         if I_L > −1.0
4           then EB(i, j).I_0 ← EB(i, j).I_0 + I_L
5    else  update bit fields EB(i, j).(B, Ba)
```

Next, for every new light position/parameters, algorithm $LightSilhouettes()$ below is called, updating the edge buffer to hold the spotlight's silhouette upon the mesh (fig. 1). Its parameters are the mesh, the origin, a spotlight source $p_{light}$, and cone radius angle $\theta$. For each triangle in the mesh, we determine whether it is within the circle of illumination formed by the distance from the light source and attenuation angle (lines 3 - 6). If it is, we update the edge buffer with the intensity of the light at those edges, and mark each edge as front-facing (lines 7 - 11).

```
LIGHTSILHOUETTES(mesh, origin, p_light, θ)
 1  for each triangle T ∈ mesh
 2    do extract vertex indices (a, b, c) of T
 3       D ← origin − p_light
 4       L ← v_a − p_light
 5       γ = cos^−1(D · L)
 6       if (γ ≤ θ) and (L.N < 0.0)
 7         then compute I_L
 8              I_L ← I_L/2.0
 9              UpdateEB(a, b, FRONT, I_L)
10              UpdateEB(b, c, FRONT, I_L)
11              UpdateEB(a, c, FRONT, I_L)
```

**Target tone** In our system the spotlight intensity (the target tone) is computed as $I_L = I_0(D \cdot L)/(k_c + k_l d + k_q d^2)$, where $I_0$ is the user-defined light intensity, $d$ is the distance from $P_{light}$ to triangle vertex $a$, and $(k_c, k_l, k_q)$ are the (user-defined) constant, linear, and quadratic attenuation factors, respectively. Notice that the light intensity of the edge is equal to the averaged intensities of its two adjacent triangles. This is achieved by first dividing the original intensity by two (line 8 in $LightSilhouettes()$) and then adding it to $I_0$ in the edge-buffer (line 4 in $UpdateEB()$).

**Value gradation** One technique used by illustrators to grade target tone variations is to enlarge the space between strokes, as the tone gets closer to full light intensity [18] (figure below).



We reproduce this technique by simply adjusting the $length$ of the stroke directly at each mesh edge, thus creating those space variations. This procedure works as follows: the edge-buffer is traversed, and for every edge $ab$, a single pen mark is modeled by first determining its stroke base line endpoints $(p_0, p_1)$, where $p_0$ is the midpoint of $ab$ and $p_1 = p_0 + \rho\phi\vec{d}$. The proportion of ink to be deposited is given by $\rho = 1.0 - I_0$, with $I_0$ being the light intensity at the midpoint of edge $ab$; $\phi$ is a user-defined length value for the stroke, and $\vec{d}$ is the stroke direction equal to $e_1$, $e_2$, $f_1$ or $f_2$ depending on which directional field the user selects (sec. 5).

**Value reversal** In our system, the effect of *tone value reversal* (or *negative film*) can be computed by simply assigning $\rho = I_0$, which reverses the original light intensity. This effect allows fast visualizations of different contrast tone combinations, an important illustration composition technique used to create focus of attention at specific shape features of the subject [10]. Figure 3 illustrates the results of using this effect in our system.

## 7. Stroke qualities

At this stage, we include attributes along the stroke base line $p_0p_1$ to approximate the visual qualities of traditional ink strokes. Our system supports the attributes of weight distribution and path perturbation.

**Weight distribution** Consists of applying the line quality of weight, or how much ink is distributed along the path. Our goal is to control the mark weight in a way to reproduce specific ink distributions typical of short stroke-based ink drawings (algorithm $StrokeWeight()$ and fig. 2). We observed that the 1D Perlin noise function [24] allows for a controllable simulation of such weight distributions.

STROKEWEIGHT$(p_1, p_2, t_1, t_2, s, \alpha, \beta, N)$
1 $s \leftarrow s * (|ab|/lmax)$
2 **for** $pt \leftarrow 0$ *to* $1$, $step \leftarrow 1/res$
3 **do** $nib \leftarrow p_1 + pt * (p_2 - p_1)$
4 $t \leftarrow t_1 + pt * (t_2 - t_1)$
5 $r \leftarrow |PerlinNoise1D(t, \alpha, \beta, N)|$
6 $fill\ nib\ circle\ at\ (x, y, z)\ with\ radius\ r$

The first four parameters of $StrokeWeight()$ are determined by $StrokePath()$ (below); parameter $s$ corresponds to the user-defined number of nib stamps to be placed along $p_1p_2$. The remaining parameters are related to $PerlinNoise1D()$ function, where $\alpha$ is the weight when the sum is formed, $\beta$ is the harmonic scaling/spacing, and N is number of harmonics. In $StrokeWeight()$, line 1 adjusts $s$ considering the length of $p_1p_2$ and the maximum edge length $l_{max}$ in the mesh (both lengths related to the current view location eye). Lines 3 and 4 compute the nib location and parametric distance $t$ along $p_1p_2$, respectively. The radius $r$ of the nib determines the local weight of the mark, which is calculated by Perlin's 1D noise function. In Figure 2 (middle row), $N = 9$, $\alpha = 0.77$ and $\beta = 0.7, 0.8, 0.9, 1.0$ for (1, 2, 3, 4), respectively.

**Path perturbation** A path is defined along baseline $p_0p_1$ which simulates the hand gesture trace of the pen nib on the drawing paper. The algorithm for generating the path is as follows:

STROKEPATH$(p_0, p_1, m, \omega, c, s, \alpha, \beta, N)$
1 $(chain, np) \leftarrow PerturbLine(p_0, p_1, m, \omega, c)$
2 **for** $i \leftarrow 0$; $i < np - 1$; $i \leftarrow i + 1$
3 **do** $(u_1, u_2) \leftarrow (chain[i], chain[i + 1])$
4 $(t_1, t_2) \leftarrow (i/(np\text{-}2), (i + 1)/(np\text{-}2))$
5 $StrokeWeight(u_1, u_2, t_1, t_2, s, \alpha, \beta, N)$

The first two parameters in $StrokePath()$, $(p_0, p_1)$, are the endpoints of the stroke baseline. The next three parameters are used by function $PerturbLine()$ (line 1) for simulating the effect of small hand gesture variations along $p_0p_1$. We adapted the function $PerturbedLineSegment()$, introduced by Salisbury et al [28]: $m$ is the magnitude and $\omega$ the base frequency of waviness; $c$ is the magnitude of curviness. The remaining four parameters are used by algorithm $StrokeWeight()$ to stylize the perturbed stroke with different weight values. In line 1 of $StrokePath()$, the line $p_0p_1$ is perturbed by waviness functions by calling $PerturbLine()$ which returns an array $chain$ of $np$ points; then, successive pairs of points $(u_1, u_2)$ are extracted from the $chain$ array; in line 4, $(t_1, t_2) \in [0, 1]$ correspond to the scalar distances along the chain indicating where ink should start and end being placed, respectively. Each line segment $u_1u_2$ is then rendered as a single stylized mark by function $StrokeWeight()$. In Figure 2 (bottom row), $m \in [0, 0.002], \omega \in [0.0, 0.04], c \in [0, 0.001]$.

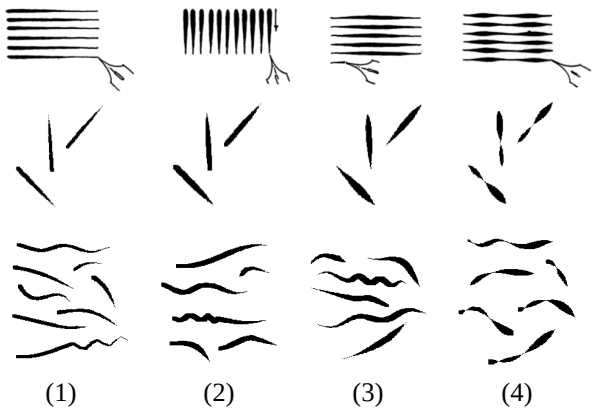



**Figure 2.** *Top row:* **Real samples of ink distribution patterns along a straight pen stroke with (1, 2) decreasing and (3, 4) varied pressure of the pen nib [11].** *Middle row:* **approximations to the real samples by applying algorithm** $StrokeWeight()$ **to a single line segment.** *Bottom row:* **suggesting hand gesture variations by applying algorithm** $StrokePath()$ **to the weighted lines in middle row.**

## 8. Results and discussion

The table below shows the average times (in seconds) for pre-processing and rendering (without light) all the strokes of the models presented in this paper. Running times were gathered from a 2.65 GHz Pentium IV with OpenGL/ATI Radeon 9700 graphics.

| $\triangle$ mesh | $edges$ | *PP* | *L* | *W* | *P* | *W+P* |
|---|---|---|---|---|---|---|
| Inner ear | 16,373 | $< 1$ | $< 1$ | 1 | 3 | 10 |
| Pelvis/hands | 42,231 | 1 | 1 | 7 | 14 | 27 |
| Skull | 49,133 | 1 | 1 | 10 | 27 | 45 |
| Dental arcade | 118,999 | 2 | 2 | 20 | 42 | 70 |
| Igea artifact | 136,484 | 2 | 2 | 25 | 57 | 94 |

Pre-processing (*PP*) refers to construction of the edge buffer and calculation of stroke directional fields. The remaining columns refer to rendering a single straight line (*L*), adding weight to *L*, perturbation (*P*), and lastly adding both weight and perturbation to *L*. In our current system, models can be interactively visualized and composed (with spotlight) with just simple lines ($L$) and then rendered fairly quickly (but not interactively) with full stylization ($W$, $P$, $W + P$) to get a better artistic effect. As expected, rendering times decrease depending on the spotlight cone angle selected, due to the discarding of triangles not visible to the light rays.

We observed that each of the four directional fields result in good visualizations of shape features, thus allowing different interpretations about the form of the subject. In

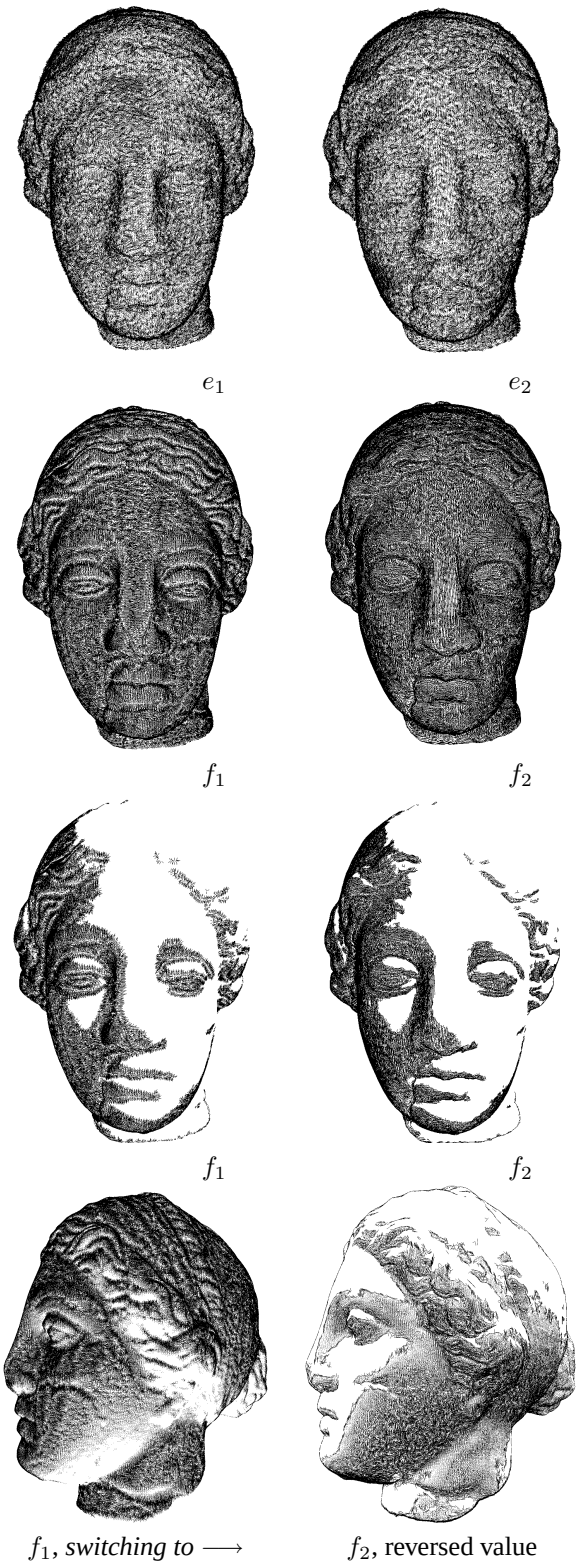
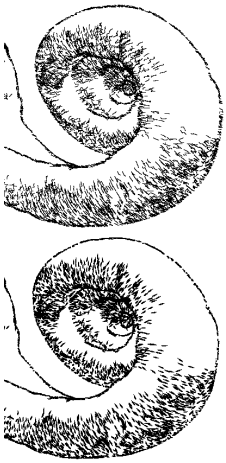



**Figure 3. The Igea artifact (268,686 $\triangle$s).** *From top to bottom:* **the four stroke directional fields (rows 1 and 2); placing a high-contrast (row 3) and an attenuated spot light (row 4).**

the Igea artifact (fig. 3), $f_1$ and $f_2$ clearly provide better depiction of shape features and overall form. Notice how the two directional fields allow different interpretations of the same facial features. Direction $f_1$ seems to depict the volume of facial features (i.e. lips, nose), whereas $f_2$ seems to direct the visualization along the length of the face. The same figure shows the effect of high-contrast lighting and value reversal after an attenuated spotlighting. Notice the material wear patterns revealed at the facial regions (fig. 3, bottom-most row).



The figure in the left shows part of the human inner ear model (32,700 $\triangle$s) rendered in $f_1$ with perturbed (top) and weighted (bottom) strokes. In Figure 4, the skull is rendered with weighted strokes under a highly attenuated spotlight, revealing different shape features. For the pelvis and hands, we place a spotlight with little attenuation and a spread angle $\theta = 40^0$. Notice the overall form depiction and shape features. Finally, the dental arcade model is rendered under an attenuated spotlight and with depth-cueing, using the approach described in [32], where the length of the strokes is re-scaled based on maximum depth values of the edges.

## 9. Conclusions and future work

This paper presents a new algorithm following an edge-based stroke placement approach. Our technique integrates two illustration strategies for depicting shape features: selection of drawing direction and the use of light. At each edge of the mesh, a stroke is mapped directly to one of four possible directional fields. For lighting, we introduce the idea of "spotlight silhouettes," allowing fast illumination computation, with target tone matched by adaptive stroke length adjustment. Each stroke can be rendered as straight single width lines or stylized by a noise-based approach for weight distribution and path perturbation. Our technique also allows visual effects of reverse tone values and depth cueing. Examples with models from scientific subjects demonstrate the capabilities of our system. We are encouraged by the results and preliminary feedback received from scientific illustrators. We have also identified areas in our system that needs further improvement, as given below.

**The planar case.** On planar mesh areas the cross products for $\xi_{ab}$ and $\xi_{12}$ (sec. 4) will be zero, leaving us with no directions when using our tangent space method. This is not an issue for the scanned anatomical and archaeological models we have used in our examples, as these generally have no large completely flat areas. For other meshes (i.e. terrains, buildings), this may be an issue. To handle these situations, it would be beneficial to have a method of picking directions without relying on the normal cross products, so we can still render strokes over those areas.

**Mesh dependency of our simple technique.** We noted that the direction field $(f_1, f_2)$ from the simple tangent space technique is visibly better for shape feature determination than the $(e_1, e_2)$ field for the Igea artifact (fig. 3). Due to the edge-based nature of that technique, this result is influenced to some degree by the nature of the mesh itself. Further investigation of the results on differently-tesselated meshes would give a better idea of the cases in which $(f_1, f_2)$ has an advantage over $(e_1, e_2)$, and vice-versa.

**Efficiency of the system.** We would like to refine our approach to achieve real-time refresh rates for the stylized renderings. Currently, our system is best used to interactively inspect the model with spot light adjustment and simple stroke line rendering, and then separately render an artistic final image with full stylized strokes. Possible strategies would involve more advanced mesh culling - or perhaps not rendering a stroke on every edge, just every few edges for very dense meshes.

**User-defined parameters.** Currently, several user-defined parameters are used for creating our pen-and-ink style renderings. It would be convenient to end-users of system if these parameters were determined from the mesh itself.

## Acknowledgments

Many thanks to Emily S. Damstra for her valuable input on scientific illustration techniques and to Patricia Rebolo Medici and the anonymous reviewers for their useful comments and suggestions. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

## References

[1] D. Akers, F. Losasso, J. Klingner, M. Agrawala, P. Hanrahan, and J. Rick. Conveying shape and features with image-based relighting. In *Proc. of IEEE Visualization '03*, pages 349–354, 2003.

[2] J. Buchanan and M. Sousa. The edge buffer: a data structure for easy silhouette rendering. In *Proc. of NPAR '00*, pages 39–42, 2000.

[3] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000.

[4] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proc. of SIGGRAPH '00*, pages 13–18, 2000.

[5] D. H. Eberly. *3D Game Engine Design : A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann, 2000.

**Figure 4. Bones of the skull (98,192 $\triangle$s), pelvis and hands (84,670 $\triangle$s) with weighted strokes in direction $f_2$ under attenuated spotlight. Dental arcade (233,204 $\triangle$s) with attenuated spotlight, single-width, in $f_2$, with depth-cueing.**


[6] 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*, pages 43–52, 2000.

[7] N. Goldstein. *The Art of Responsive Drawing*. Prentice-Hall, 1999.

[8] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proc. of SIGGRAPH '98*, pages 447–452, 1998.

[9] B. Gooch, P.-P. J. Sloan, A. A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. In *Proc. of Symposium on Interactive 3D Graphics*, pages 31–38, 1999.

[10] A. L. Guptill. *Drawing with Pen and Ink*. Van Nostrand Reinhold Company, 1961.

[11] A. L. Guptill. *Freehand Drawing Self-Taught, With Emphasis on the Techniques of Different Media*. Watson-Guptill, 1980.

[12] P. Hall. Non-photorealistic shape cues for visualization. In *Proc. of WSCG '95*, pages 113–122, 1995.

[13] J. Hamel. *Alternative Lighting Methods for Comuter Generated Line Drawings*. PhD thesis, University of Magdeburg, 2000.

[14] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proc. of SIGGRAPH '00*, pages 517–526, 2000.

[15] S. Hiller, H. Hellwig, and O. Deussen. Beyond stippling: methods for distributing objects in the plane. *Computer Graphics Forum*, 22(3):515–522, 2003.

[16] E. Hodges. *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, 1989.

[17] M. Kaplan, B. Gooch, and E. Cohen. Interactive artistic rendering. In *Proc. of NPAR '00*, pages 67–74, 2000.

[18] F. Lohan. *Pen and Ink Techniques*. Contemporary Books, 1978.

[19] A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proc. of IEEE Visualization '02*, pages 211–218, 2002.

[20] D. Martin and J. C. Torres. Rendering silhouettes with virtual lights. *Computer Graphics Forum*, 20(4):271–282, 2001.

[21] M. Meyer, M. Desbrun, P. Schröder, and A. Barr. Discrete differential geometry operators for triangulated 2-manifolds. In *Proc. of Vismath '02*, 2002.

[22] J. Northrup and L. Markosian. Artistic silhouettes: A hybrid approach. In *Proc. of NPAR '00*, pages 31–37, 2000.

[23] O. E. M. Pastor, B. Freudenberg, and T. Strotthote. Real-time animated stippling. *IEEE CGA*, 23(4):62–68, 2003.

[24] K. Perlin. An image synthesizer. In *Proc. of SIGGRAPH '85*, pages 287–296, 1985.

[25] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proc. of SIGGRAPH '01*, pages 579–584, 2001.

[26] P. Rawson. *Drawing*. University of Pennsylvania Press, 1987.

[27] C. Rossl, L. Kobbelt, and H.-P. Seidel. Line art rendering of triangulated surfaces using discrete lines of curvature. In *Proc. of WSCG '00*, pages 168–175, 2000.

[28] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *Proc. of SIGGRAPH '94*, pages 101–108, 1994.

[29] A. Secord, W. Heidrich, and L. M. Streit. Fast primitive distribution for illustration. In *EG Rendering Workshop*, pages 215–226, 2002.

[30] G. Simmons. *The Technical Pen*. Watson-Guptill Publications, 1992.

[31] J. A. Smith. *The Pen and Ink Book: Materials and Techniques for Today's Artist*. Watson-Guptill Publications, 1992.

[32] M. C. Sousa, K. Foster, B. Wyvill, and F. Samavati. Precise ink drawing of 3d models. *Computer Graphics Forum*, 22(3):369–379, 2003.

[33] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. of ICCV '95*, pages 902–907, 1995.

[34] G. Winkenbach and D. H. David H. Salesin. Computer-generated pen-and-ink illustration. In *Proc. of SIGGRAPH '94*, pages 91–100, 1994.

[35] E. Wolff. *Anatomy for Artists: Being and Explanation of Surface Form*. H. K. Lewis and Co., 1958.