Illustrative Interactive Stipple Rendering

Aidong Lu, *Student Member*, *IEEE Computer Society*, Christopher J. Morris, Joe Taylor, David S. Ebert, *Member*, *IEEE*, Charles Hansen, *Member*, *IEEE*, Penny Rheingans, *Member*, *IEEE Computer Society*, and Mark Hartner

Abstract—Simulating hand-drawn illustration can succinctly express information in a manner that is communicative and informative. We present a framework for an interactive direct stipple rendering of volume and surface-based objects. By combining the principles of artistic and scientific illustration, we explore several feature enhancement techniques to create effective, interactive visualizations of scientific and medical data sets. We also introduce a rendering mechanism that generates appropriate point lists at all resolutions during an automatic preprocess and modifies rendering styles through different combinations of these feature enhancements. The new system is an effective way to interactively preview large, complex volume and surface data sets in a concise, meaningful, and illustrative manner. Stippling is effective for many applications and provides a quick and efficient method to investigate both volume and surface models.

Index Terms—Nonphotorealistic rendering, volume rendering, scientific visualization, medical imaging, illustration, stippling.

1 Introduction

Throughout history, archaeologists, surgeons, engineers, and other researchers have sought to represent the important scientific data that they have gathered in a manner that could be understood by others. Illustrations have proven to be an effective means to achieve this goal because they have the capability of displaying information more efficiently by omitting unimportant details. This refinement of the data is accomplished by directing attention to relevant features or details, simplifying complex features, or exposing features that were formerly obscured [36]. This selective inclusion of detail enables illustrations to be more expressive than photographs and provides a feature based compression of the data.

Indeed, many natural science and medical publications use scientific illustrations in place of photographs because of the illustrations' educational, training, and communicative utility [10]. Illustrations can represent a large amount of information in a relatively succinct manner, as shown in Figs. 2 and 3. Frequently, areas of greater emphasis are stippled to show detail, while peripheral areas are simply outlined to give context. The essential object elements (e.g., silhouettes, surface, and interior) can be combined to create a simple, clear, and meaningful image. By controlling the level of detail in this way, the viewer's attention can be directed to particular items in the image. This principle forms the basis of our stipple rendering system.

Stipple drawing is a pen-and-ink illustration technique where dots are deliberately placed on a surface of contrasting color to obtain subtle shifts in value. Traditional stipple drawing is a time-consuming technique. However, points have many attractive features in computer-generated images. Points are the minimum element of all objects and have connatural features that make them suitable for various rendering situations, no matter whether surface or volume, concrete or implicit. Furthermore, points are the simplest and quickest element to render. By mimicking traditional stipple drawing, we can interactively visualize modestly sized simulations.

We previously introduced in [15] a nonphotorealistic rendering (NPR) stipple-based volume rendering system. When initially exploring an unknown volume data set, this system provides an effective means to preview this data and highlight areas of interest in an illustrative fashion. The system creates artistic rendering effects and enhances the general understanding of complex structures. Once these structures are identified, the user may choose additional complementary rendering techniques to generate a more detailed image of these structures. It is the use of NPR techniques that provides the stipple volume renderer with its interactivity and illustrative expressiveness. We refer to this type of NPR technique as *illustrative rendering*.

NPR is a powerful tool for making comprehensible, yet simple images of complicated objects. Over the past decade, the field of NPR has developed numerous techniques to incorporate artistic effects into the rendering process [8], [31]. Various approaches have been used, including pen-and-ink illustration, silhouette edges, and stroke textures. Most of the research in the field of nonphotorealistic illustration has concentrated on strokes, crosshatching, and pen and ink techniques [9], [18], [30] and most of the current research still concentrates on surface renderings, which requires surface geometry. We choose to directly

- *A. Lu, J. Taylor, and D.S. Ebert are with the Purdue University Rendering and Perceptualization Lab, Purdue University, West Lafayette, IN 47907. E-mail: {alu, jtaylor, ebertd}@ecn.purdue.edu.*
- *C.J. Moris is with the IBM T.J. Watson Research Center, 19 Skyline Dr., Hawthorne, NY 10532. E-mail: cjmorris@us.ibm.com.*
- *C. Hansen and M. Hartner are with the Scientific Computing and Image Institute, School of Computing, University of Utah, Salt Lake City, UT 84112. E-mail: hansen@cs.utah.edu, hartner@sci.utah.edu.*
- *P. Rheingans is with the Computer Science and Electrical Engineering Department, University of Maryland-Baltimore County, Baltimore, MD 21250. E-mail: rheingan@csee.umbc.edu.*

Manuscript received 15 Nov. 2002; accepted 3 Dec. 2002.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number S10009-1102.

Fig. 1. This image shows color stipples with both distance color blending and tone shading.

Fig. 2. Traditional manual stipple drawing of idol by artist George Robert Lewis [11].

render volume and surface data sets without any additional analysis of object or structure relationships within the data sets. Direct stippling rendering not only maintains all the advantages of NPR, but it also makes interactive rendering and illustration feasible on useful-sized data sets because of two attributes of points: fast rendering speed and innate simplicity.

For volume data sets, the volume resolution is initially adjusted for optimum stipple pattern rendering and point lists are generated corresponding to the gradient magnitude and direction. In our interactive stippling system, a rendering mechanism is introduced that incorporates several feature enhancements for scientific illustration. These enhancements include a new method for silhouette curve generation, varying point sizes, and stipple resolution adjustments based on distance, transparency, and lighting effects. By combining these feature enhancements, data sets can be rendered in different illustration styles, as can be seen in Fig. 1.

In this paper, we extend our previous work in volume stippling to a direct interactive stipple rendering system for both volumes and surfaces. We present several new features which improve the quality, speed, and usability of the stipple system. As with the volume rendering method, the surface stippling [16] uses actual points as geometry to achieve stipple rendered images. We also describe the use of the latest graphics hardware capabilities to accelerate the rendering by performing the silhouette rendering on the GPU and the stipple density enhancement calculations as a vertex program.

2 Related Work

NPR has been an active area of research, with most of the work concentrating on generating images in various traditional styles. The most common techniques are sketching [34], pen-and-ink illustration [6], [27], [28], [36], silhouette rendering [18], [23], [25], [29], and painterly rendering [1], [4]. Pen-and-ink rendering uses combinations of strokes (i.e., eyelashing and crosshatching) to create textures and shading within the image.

Lines, curves, and strokes are the most popular among existing NPR techniques. Praun et al. [24] presented a real-time system for rendering of hatching strokes over arbitrary surfaces by building a lapped texture parameterization where the overlapping patches align to a curvature-based direction field. Ostromoukhov [21] illustrated some basic techniques for digital facial engraving by a set of black/white and color engravings, showing different features imitating traditional copperplate engraving. Hertzmann [9] presented a method for creating an image with a hand painted appearance from a photograph and an approach to designing styles of illustration. He demonstrated a technique for painting with long, curved brush strokes, aligned to the normals of image gradients, to explore the expressive quality of complex brush stokes. Winkenbach and Salesin [37] presented algorithms and techniques for rendering parametric free-form surfaces in pen and ink.

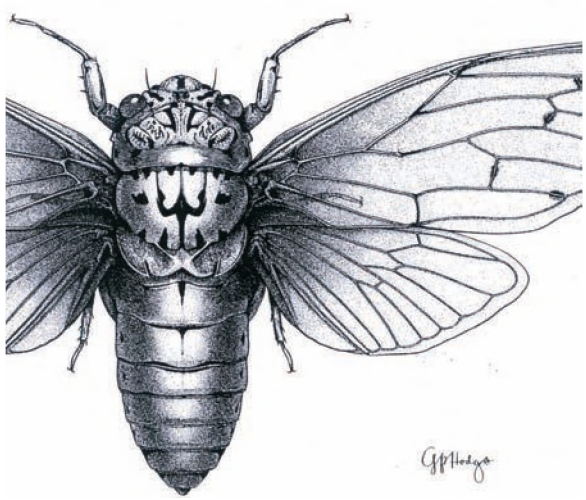

Fig. 3. Traditional manual stipple drawing of cicadidae by artist Gerald P. Hodge [11].

Deussen et al. [5] used points for computer generated pen-and-ink illustrations in simulating the traditional stipple drawing style. Their method first renders polygonal models into a continuous tone image and then converts these target images into a stipple representation. They can illustrate complex surfaces vividly. However, their method does not work for rendering volumes and is too slow for interactive rendering.

NPR techniques have only recently been applied to the visualization of three-dimensional (volume) data. Interrante developed a technique for using three-dimensional line integral convolution (LIC) using principal direction and curvature to effectively illustrate surfaces within a volume model [12]. Treavett et al. also used illustration techniques to render surfaces within volumes [32], [33]. In both cases, the results were compelling, but the techniques are surface-based visualization techniques, rather than direct volume rendering techniques that can show not only surfaces, but also important details of the entire volume.

Several NPR techniques have recently been applied to volume rendering. Ebert and Rheingans [7] showed the power of illustrative rendering techniques for volume data; however, the renderer was based on ray-casting and too slow for interactivity or quick exploration of the data. Our current work builds upon enhancement concepts from that work. Furthermore, interactive volume rendering has garnered a significant amount of attention [19] and NPR methods have been applied to obtain interactive performance while producing effective volume renderings [2], [3]. Treavett et al. [33] implemented artistic procedures in various stages of the volume-rendering pipeline. Techniques such as brush strokes, control volumes, paint splatting, and others were integrated into their rendering system to produce a variety of artistic effects to convey tone, texture, and shape.

However, tone, texture, and shape can be effectively conveyed by simply controlling the placement and density of points. Though not a primary focus in illustrative rendering systems until recently, points have previously been used as rendering primitives. Levoy and Whitted [14] first demonstrated that points could be used as a display primitive and that a discrete array of points arbitrarily displaced in space, using a tabular array of perturbations, could be rendered as a continuous three-dimensional surface. Furthermore, they established that a wide class of geometrically defined objects, including both flat and curved surfaces, could be converted into points. The use of points as surface elements, or "surfels," can produce premium quality images which consist of highly complex shape and shade attributes, at interactive rates [22], [39].

The main difference between previous stipple and point rendering research and ours is that our system interactively renders volumes and surfaces with points instead of just surfaces with points. Within volume rendering, the closest related technique is splatting [38], [26], which traditionally does not incorporate the effectiveness of illustration techniques. More recently, Wilson et al. [35] presented a hybrid technique for interactive volume rendering using a combination of hardware accelerated direct volume rendering (DVR) and point-based rendering. Unlike pure hardware accelerated DVR, the hybrid approach is not limited by the size of the available texture memory. The hybrid technique differs from ours in that it does not use any NPR enhancements and only utilizes points at those locations where the volumetric data contains large amounts of error. In the remainder of this paper, we show the effectiveness of a simple point-based interactive stippling system and describe how a number of illustrative enhancement techniques can be utilized to quickly convey important characteristics for rapid previewing and investigation of both volume and surface models.

3 The Stipple Renderer

The clarity and smoothness displayed by stippling, coupled with the speed of hardware point rendering, makes stippling an effective tool for illustrative rendering. As with all scientific and technical illustration, this system must perform two key tasks. First, it must determine what to show, primarily by identifying features of interest. Second, the system must carry out a method for how to show identified features. The stipple renderer consists of a point-based system architecture that behaves as a volume and surface renderer and visually extracts various features of the data by selective enhancement of certain regions.

For volumes, volume gradients are used to provide structure and feature information. With this gradient information, other features can be extracted, such as the boundary regions of the structure. We can then illustrate these volumes using stippling techniques with a particular set of features in mind.

There are several issues specific to surfaces that must be handled, including back face culling and surface shading. Points alone could be rendered, but this would result in stipples representing both front and back facing surfaces being rendered. To prevent this, stipples that represent back facing surfaces are removed with a depth test by first rendering the entire surface object opaque. Unfortunately, even with back face culling, some geometry which would not have been rendered with hidden surface methods contributes stipples to the final image. Therefore, to achieve correct hidden surface removal, we displace the stipples by a small epsilon along the normal direction. We use the OpenGL polygon offset, which takes into account the orientation of the polygon when calculating this offset.

To effectively generate renderings of both volume and surface data sets at interactive rates, the system has two main components: a preprocessor and an interactive point renderer with feature enhancement.

4 Preprocessing

Before interactive rendering begins, the preprocessor automatically generates an appropriate number of stipple points for each object element based on its characteristics. This preprocessing stage handles a number of calculations that do not depend on viewpoint or enhancement parameters, including the calculation of volume gradient direction and magnitude (surface normals), the initial estimation of stipple density from object resolution, and the generation

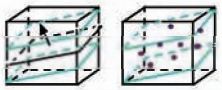

Fig. 4. Initial point generation in voxel using a Poisson disc approximation.

of an initial point distribution. Additionally, for volumes, the voxel values and gradients are all normalized.

4.1 Gradient Processing for Volumes

Gradient magnitude and direction are essential in feature enhancement techniques, especially when rendering CT data [11]. Some feature enhancements are significantly affected by the accuracy of the gradient direction, especially our light enhancement. Noisy volume data can create problems in generating correct gradient directions. Additionally, first and second derivative discontinuity in voxel gradients can affect the accuracy of feature enhancements. Therefore, we use Neumann et al.'s [20] improved gradient estimation method for volume data. Their method approximates the density function in a local neighborhood with a three-dimensional regression hyperplane whose four-dimensional error function is minimized to get the smoothed data set and estimated gradient at the same time.

4.2 Initial Point Generation

In several illustrative applications, units (such as points, particles, or strokes) are distributed evenly after random initialization. Due to constantly changing scenes, these individual units are redistributed in every frame. This process is very time-consuming and leads to issues with frame-to-frame coherence. To alleviate this problem, we approximate a Poisson disc distribution to initially position a maximum number of stipples. After this preprocessing step is performed and the stipple positions are determined, any processing that is subsequently performed (i.e., feature enhancements, motion) simply adjusts either the number of stipples that are drawn within each voxel/polygon or their respective size. The maximum number of stipples for each voxel or polygon is also adjusted per frame based on distance and shading enhancements. We always select the stipples that will be drawn from a pregenerated list of stipples for each voxel or polygon, therefore maintaining frame-to-frame coherence for the points.

For volume models, we generate stipples near the gradient plane for the voxels whose gradient magnitude is above a user specified threshold according to the statistics of the gradient magnitude distribution. We place stipples randomly, around the center of the voxel, between two planes, p1 and p2, that are parallel to the tangent plane, p0, and are separated by a distance chosen by the user, as shown in Fig. 4. Next, we adjust the point locations in this subvolume so that they are relatively equally spaced, approximating the even distribution of points in a stipple drawing.

The surface-based approach uses an analogous method, generating the initial points for each polygon randomly within the polygon and then redistributing them to approximate a Poisson disc distribution.

4.3 Initial Resolution Adjustment

When viewing an entire data set, as the object's size increases, each element's (voxel for volume or polygon for surface) screen projection is reduced. Even if we assign at most one point per element, areas with high gradient magnitude still appear too dark. We define N_{max} as the maximum number of stipples that each element can contain during the rendering process. After reading the data set, we approximately calculate the maximum projection of an element on the screen and set its maximum number of points to be equal to the number of pixels in the projection area. This reduces redundant point generation and improves the stippling pattern in the resulting images. Also, for surfaces, considering the polygon's area in generating the maximum number of stipples ensures that the stipple pattern is independent of the object tessellation.

5 Feature Enhancements

Scientific illustration produces images that are not only decorative, but also serve science [10]. Therefore, the rendering system must produce images accurately and with appropriately directed emphasis. To meet this requirement, we have explored several feature enhancements in an attempt to simulate traditional stipple illustrations. These feature enhancements are based on specific characteristics of a particular element: Whether it is part of a boundary or silhouette, its spatial position in relation to both the entire object and the entire scene, and its level of illumination due to a light source. In particular, silhouette curves (common in stipple drawings) are very useful for producing outlines of boundary regions and significant lines along interior boundaries and features.

To enable the use of all of our feature enhancements, each element has the following information stored in a data structure: number of points, gradient (volumes) or surface normal (surfaces), element scalar data value, point size, and a point list containing the x, y, z location of each point. Our feature enhancements, calculated on a per frame basis, determine a point scaling factor according to the following sequence: boundary, silhouette, resolution, light, distance, and interior. For different data sets, we select a different combination of feature enhancements to achieve the best effect.

The point count per voxel/polygon, T_i, is the product of the maximum point count, N_{max}, and the selected enhancement factors described in the following sections. Except for the boundaries and silhouettes enhancement for volume data sets, all other feature enhancements are subtractive. All feature enhancements are applied to volume models, while all except boundary, silhouette, and interior enhancements are applied to surface models. The equations used for each feature enhancement are shown in Table 1 and a more detailed description of the equations can be found in [15].

5.1 Boundaries and Silhouettes

In traditional stipple drawings, boundaries are usually represented by a high concentration of stipples that cluster on surfaces. In a scalar volume, the gradient of a voxel is a good indication of whether the voxel represents a boundary region. Boundary and silhouette enhancements

TABLE 1
Feature Enhancement Equations

Feature Enhancement	Equations	
	Volume	Surface
Boundary	$T_b = v_i * (k_{gc} + k_{gs} * (\|\nabla \vec{V}_i\|^{k_{gc}}))$	–
Silhouettes	$T_s = v_i * (k_{sc} + k_{ss} * (1 - (\|\nabla \vec{V}_i \cdot \vec{E}\|))^{k_{se}})$	–
Resolution	$T_r = [\frac{(D_{near}+d_i)}{(D_{near}+d_0)}]^{k_{re}} * (\|\nabla \vec{V}_i \cdot \vec{E}\|)^{k_{n_x}}$	$T_r = [\frac{(D_{near}+d_i)}{(D_{near}+d_0)}]^{k_{re}} * (\|\vec{N} \cdot \vec{E}\|)^{k_{n_x}}$
Distance	$T_d = 1 + (\frac{z}{a})^{k_{dc}}$	$T_d = 1 + (\frac{z}{a})^{k_{dc}}$
Interior	$T_t = \|\nabla \vec{V}_i\|^{k_{tc}}$	–
Lighting	$T_l = 1 - (\vec{L} \cdot \nabla \vec{V}_i)^{k_{lc}}$	$T_l = 1 - (\vec{N} \cdot \vec{L})^{k_{lc}}$

v_i is the scalar voxel data value, $\nabla \vec{V}_i$ is the voxel gradient vector, $\vec{E}$ is the eye vector, $\vec{L}$ is the light vector, z is the object's depth coordinate, D_{near} is the near plane depth coordinate, $(-a, a)$ is the depth range in the volume, and ks are user adjustable parameters.

are determined using volume illustration techniques [7]. Several additional user defined factors determine the range and sharpness of the boundary enhancement as well as the relative effect of the data and gradient values on the calculation.

By making the stipple placement denser in voxels of high gradient, boundary features are selectively enhanced. The silhouette enhancement factor is constructed in a manner similar to the boundary enhancement factor in areas oriented orthogonally to the view plane, forming a silhouette edge. The boundary enhancement factor for a voxel is a function of the voxel's scalar data value and the voxel's gradient magnitude. The silhouette enhancement is a function of voxel's scalar data value and the dot product of the voxel's gradient vector and the eye vector. Using the boundary and silhouette enhancement factors, we can effectively render the outline of the features in the volume. Therefore, points are dense on the outline of the objects, while sparse on other boundaries and in the interior. Fig. 5 shows an abdominal CT volume data set rendered with stipples. Boundary areas, particularly those in silhouette, are enhanced, showing these features clearly.

5.2 Resolution

Traditionally, the number of stipples used to shade a given feature depends on the viewed resolution of that feature. By using a resolution factor, we can prevent stipple points from being too dense or sparse. The resolution factor adjusts the number of points in each voxel or polygon and produces the effect that the features become larger and clearer when the object moves closer to the viewpoint. It also helps increase rendering performance by eliminating unnecessary rendering of distant points.

We compute the resolution enhancement differently based on whether an element projects larger than or smaller than a user specified minimum screen coverage area (two to nine pixels). If the element's projection is large enough to generate a stipple pattern, the resolution enhancement is a function of the element's z value and the object's minimum z value, both relative to the near plane. A user defined parameter determines the relative effect of the resolution enhancement. In Fig. 6, the same model is viewed at three different distances, but the resulting stipple density is the same for each. Fig. 7 shows a similar result of resolution enhancement for the polygonal Stanford bunny model.

When the projection of a rendering element becomes too small, creating a stipple pattern is not practical and we must select an appropriate percentage of the elements to achieve a consistent gray-level image as the object decreases in size. We achieve this consistency by slightly adjusting the number of stipples to be drawn. A random probability value is assigned to each voxel or polygon to affect the stipple number. The stipple numbers are calculated by the weighted sum of the current maximum stipple number and the random value. The two weights change, corresponding to the location of the center of the object. The effect is that the current maximum stipple number is always the

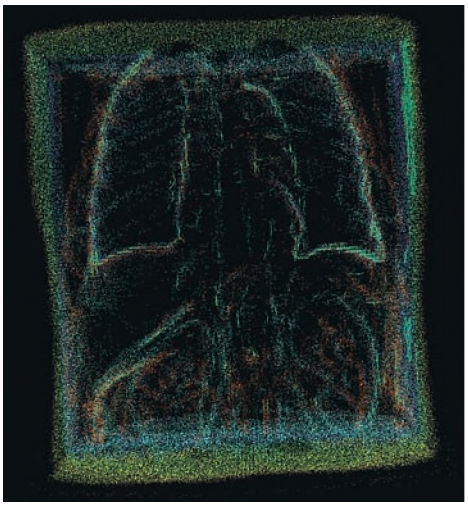

Fig. 5. Abdomen rendered with volume stipple renderer shows boundary and silhouette enhancements.

Fig. 6. Resolution enhancement of the leg volume data set.

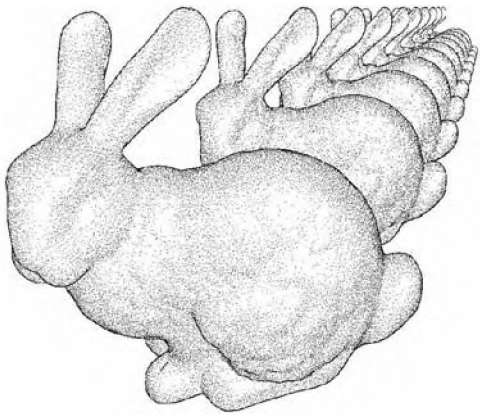

Fig. 7. This image shows the multiscale resolution enhancement where polygons become smaller than one pixel. The system automatically adjusts the stipples to provide a consistent appearance.

dominant factor, but the random value becomes more and more important when the object is moved further away from the view point.

Another technique that is used to avoid frame to frame incoherency is the use of density determined point intensity. We interpret a fractional density value as the intensity of the point to be drawn, whereas the previous coherency technique interpreted such a value as a probability. Using the probabilistic technique, three consecutive elements with densities equal to one third would likely produce two empty elements and one element with a single fully intense stipple. The second technique would produce stipples with intensity equal to one third in all three elements. Fig. 8 and Fig. 9 show the effects of point intensity and probability based resolution enhancement.

5.3 Distance

In resolution enhancement, we use the location of the whole object in the scene. The location of different elements within the overall volume presents a different challenge. Distance is an important factor that helps us understand the relationship between elements within an object. As in traditional illustration, we can enhance depth perception by using the position of an element within the object's bounding box to generate a factor that modifies both the point count and the size of the points. The distance enhancement factor is, therefore, a function of the relative depth of the element within the object's bounding box. A

Fig. 8. Another example of the resolution enhancement applied to a statue model. The first image has no enhancement, the second image has point intensity added for anti-aliasing, the third image has resolution enhancement, and the fourth image is with color.

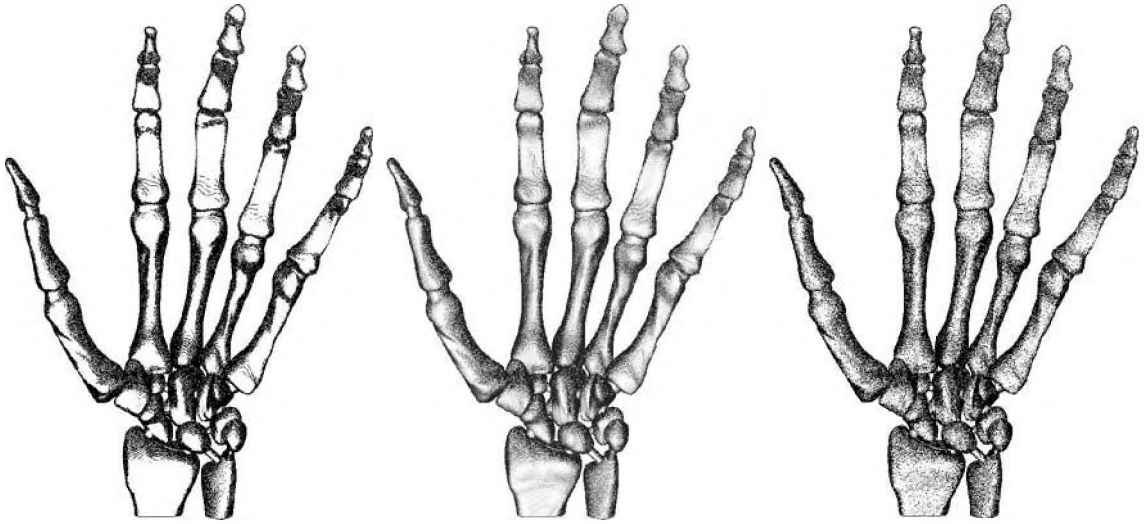

Fig. 9. Another example of the resolution enhancement applied to a hand model. The left hand has no enhancement, the middle image has point intensity added for anti-aliasing, and the right image has resolution enhancement.

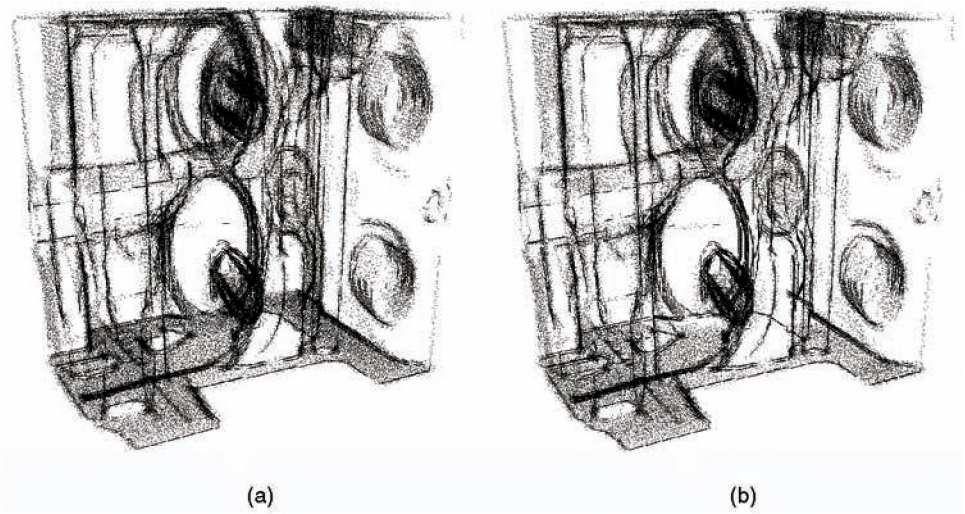

Fig. 10. Engine block rendered with volume stipple renderer: (a) shows boundary and silhouette enhancement, as well as silhouette curves and (b) distance attenuation of the engine block volume.

user defined parameter determines the strength of the effect. Fig. 10 shows an example of distance attenuation. Comparing the right image to the left, it is clear that more distant parts of the volume contain fewer and smaller points. This is most apparent in the back, right section of the engine block.

5.4 Interior

Point rendering is transparent in nature, allowing background objects to show through foreground objects. By doing explicit interior enhancement, we exaggerate this effect, allowing us to observe more detail inside the volume. Generally speaking, the point count of the outer volume elements should be smaller than that of the interior to allow the viewing of interior features. While there are several ways to implement this enhancement, we achieve this effect by varying the number of points based on the gradient magnitude of a voxel, thus achieving a better transparency effect.

The interior enhancement is also a function of a user defined parameter that controls the falloff of the transparency enhancement. With this factor, the voxels with lower gradient magnitude become more transparent. In addition, point sizes are adjusted by the transparency factor. In Fig. 11, the density of the leaves changes from sparse to dense when the gradient magnitude changes from low to high. The structure of the tree is more evident with interior enhancement.

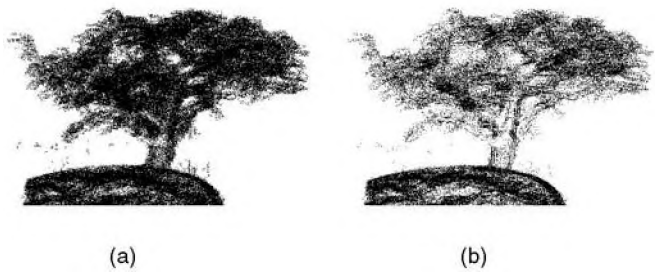

Fig. 11. Stipple rendering of bonsai tree volume. (a) Without interior enhancement, (b) with interior enhancement.

5.5 Lighting

When lighting the volume, only the front oriented voxels (where the gradient direction is within a specified number of degrees of the eye direction) are rendered. The light enhancement factor is a simple modification to the Lambertian diffuse illumination of the Blinn-Phong model. A user defined parameter adjusts the strength of the enhancement. For voxels which are front facing with respect to the light direction, we use the light enhancement factor to adjust the number of stipples to be drawn. For voxels which are back facing with respect to the light direction, we draw the current maximum number of stipples.

Surface objects use an analogous technique. The lighting factor potentially reduces the number of points drawn per polygon for each frame. The reduction in the number of points provides perceived shading based on the number of stipples. Fig. 12 shows the effect of shading on the stipple placement. Note the details which become visible with shading in the dragon on the right.

To achieve compelling lighting effects for the volume data sets presents several challenges. Since structures often overlap in the volume, it can still be difficult to identify to which structure a point belongs in complex scenes. Also, the problem of capturing both the inner and outer surfaces at the same time, while their gradient directions are opposite, must be correctly handled. These issues can all significantly reduce the quality of the lighting effects.

5.6 Silhouette Curves

Manual stipple drawings frequently contain outlines and other curves which supplement the shading cues provided by the stipples. These silhouette curves are generally drawn at two places: the outline of the objects and obvious interior curves. Different silhouette techniques are implemented for volume and surface objects. With volumetric models, searching for potential silhouette curves in the vicinity of each voxel could easily create a performance bottleneck by

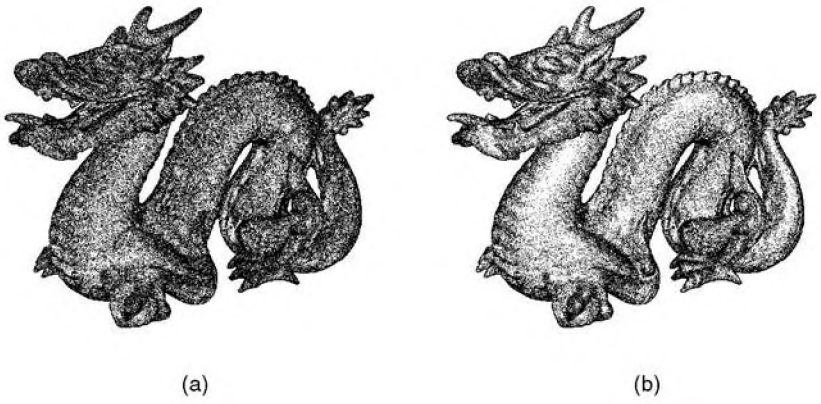

Fig. 12. Stipple rendering of dragon. (a) has no shading, while (b) has lighting enhancement added.

requiring a search in, at least, the $3 \times 3 \times 3$ subspace around each voxel. We have implemented this more exhaustive search, as well as an alternative technique using the Laplacian of Gaussian operator (LoG) as a volumetric edge detection technique.

This LoG edge detection technique provides virtually identical results and simplifies the boundary computation, so it is much faster to calculate per frame. In a preprocess, we compute the LoG value for each voxel, then, during rendering, we determine the silhouette voxels using a criteria based on the scalar data value, the gradient magnitude, the gradient direction, the eye vector, and several user defined parameters. To "sketch" silhouette curves, the voxels that satisfy these conditions have a line segment drawn through the center of the voxel in the direction of cross product of the voxel gradient and the eye vector ($\nabla \vec{V_i} \times \vec{E}$). Silhouette curves can be rendered at 20 to 30 frames per second and significantly improve image quality. Fig. 13 shows the effectiveness of silhouette curves in highlighting structures, especially the bones of the foot.

Silhouette rendering for surface-based objects should be handled differently. We have incorporated two different approaches to silhouette rendering in our stippling system. The first approach is based on Raskar and Cohen's method for interactive silhouette lines [25]. They describe a simple method for displaying image-based silhouettes using standard OpenGL hardware. Their method involves rendering front-facing polygons into the depth buffer, while back-facing polygons are rendered in wireframe mode with a slight offset and a depth test set to equal, yielding silhouette lines of uniform thickness.

When the object occupies a large portion of the image, thicker silhouette lines give a better result. However, when the object is moved away from the viewpoint, the silhouette lines should become thinner to give a consistent appearance. We use the object's bounding sphere to scale the silhouette line width based on the object's distance from the camera. The radius, which is orthogonal to the viewing direction, is projected onto the screen. The relative size provides a scaling factor for the silhouette thickness. Fig. 14 shows an example of the Stanford bunny rendered without silhouettes (left) and with silhouettes (right). The two smaller silhouette bunnies show the effects of thinning the silhouettes. The top-most has the thinning enabled, whereas the bottom smaller bunny uses the same pixel size as the larger object.

Our second approach is to use a standard "toon" shader silhouette approach and the vertex programmability extension to OpenGL. Surfaces whose normals are nearly perpendicular to the eye vector will have intensity near 1.0, thus producing a silhouette. Fig. 15 shows an example of the Stanford bunny rendered using this technique with silhouette curves.

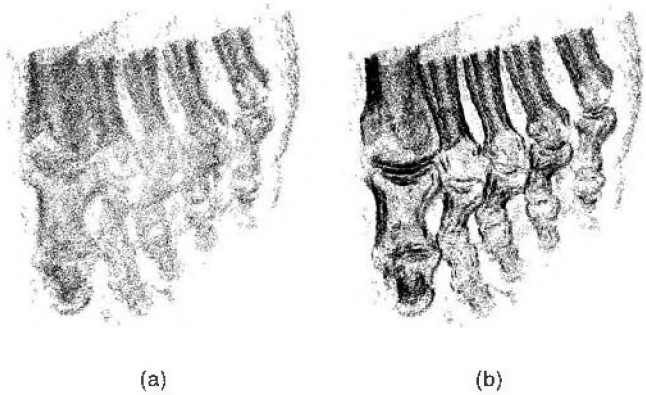

Fig. 13. Stipple rendering of the foot volume. (a) Without silhouette curves, (b) with silhouette curves.

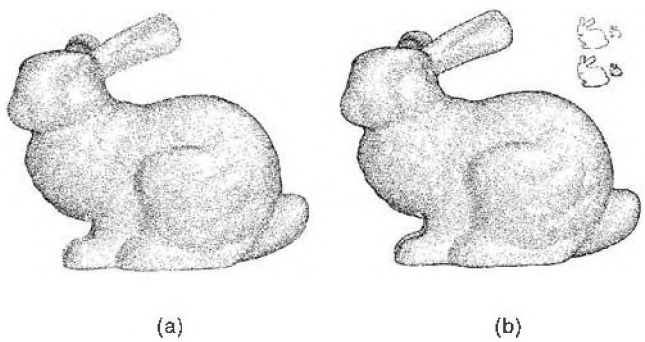

Fig. 14. (a) shows stipples without silhouette edges. (b) has image-based silhouettes enabled. The smaller bunnies demonstrate the silhouette thinning method.

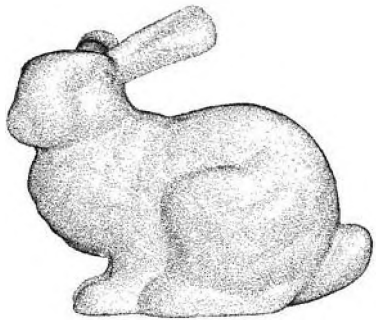

Fig. 15. This image shows our toon shading vertex program silhouette method.

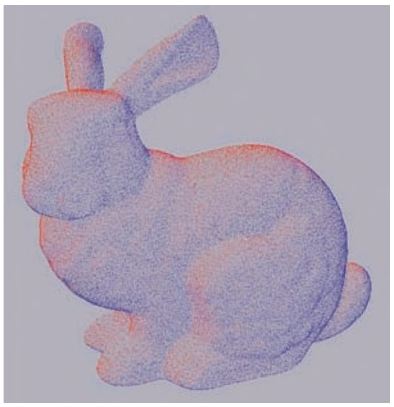

Fig. 16. This image shows color rabbit with tone shading.

5.7 Color

We have extended the monochrome pen and ink technique of stippling to use colored points, similar to the traditional painting technique of pointillism. For incorporating tone shading and distance color blending [7], we use the HLS (hue, lightness, and saturation) color model [13] to blend between warm and cool colors according to the element's orientation with respect to the light direction and position with respect to the viewer. We use an HLS to RGB color palette and a texture palette, which stores indices into the color palette, similar to the techniques in Lum and Ma [17] to improve the performance. In Fig. 1, the skin and the bones of the foot are rendered with different colors. To increase the contrast, silhouette curves were rendered in white. Fig. 16 shows the color blending from red to blue.

6 User Interaction

We have developed a two-level user-interface for the stipple rendering system. Expert users can control each of the parameters of the enhancement equations directly to achieve their desired effect. To simplify the use of the system for nonexpert users, we have also developed a more intuitive, higher-level user interface with several sliders that each affect multiple feature enhancement parameters. For example, the **gradient slider** allows the user to control the emphasis placed on boundary regions within the data. Adjusting this slider changes the amount of the enhancements that depend on the gradient magnitude. The **sharpness slider** allows the user to select the sharpness of the feature enhancements by controlling the exponent of the feature enhancement equations. The **orientation slider** allows the user to choose how much enhancement to include based on the orientation of the object (light and silhouette enhancement). The **distance slider**, **interior slider**, and **color slider** are self-explanatory.

7 Hardware Acceleration

We have implemented GPU-based stipple rendering enhancements for both volume and surface models as well as the previously mentioned silhouette enhancement for surface-based objects and will continue to explore moving more of our calculations to the vertex processing unit of the GPU. We have been able to increase the performance of the stipple renderer by implementing all of the stipple enhancements on the GPU using the vertex program extension of OpenGL on the Nvidia GeForce3. Our volume model implementation was written using Nvidia's Cg shading language, while the surface model implementation was directly written using the hardware instruction set. The conversion of the enhancement functions to the vertex program hardware instruction set is simple, with the standard trick for converting conditionals to vertex programs (compute both paths and select one of the results). The vertex programs currently use a relatively small number of instructions, which allows for future enhancements for stippling to be computed with vertex programs. Compared to software stipple rendering only, the current performance increase ranges from 0 percent to almost 100 percent. The larger the model, the greater the speedup. With silhouette rendering added, depending on the view, the vertex program enhancements can be as much as 50 percent faster than software stipple rendering (with hardware silhouettes).

When the computations are done in software, the density for a given element is calculated and only the number of stipples (vertices) that are to be drawn are sent to the hardware. When the computations are done in the vertex program, every potential stipple (vertex) for a given element is sent to the GPU and the density calculation is performed per stipple. Stipples rejected by any of the enhancements are eliminated from further processing by adding a large offset to the homogeneous z coordinate of the vertex and depth culling them. Although this vertex program calculation is faster for reasonable values of N_{max}, it requires redundant calculations compared to the software (CPU) implementation because the density calculations are performed once per stipple (vertex) instead of once per element (voxel or triangle). This redundancy, however, removes the dependency within the set of potential stipples for a given element on a single calculation (density calculation). Independent stipple calculation can be utilized for even greater performance on the latest, and future, generations of graphics boards that feature multiple parallel vertex processing pipelines.

8 Performance

We are able to interactively render reasonably sized volume and surface data sets using illustrative enhancement with

TABLE 2
Running Times (Frames per Second) for Different Data Sets

Dataset Name	*Resolution/ Polygons*	*Stipple Only*		*Stipple & Silhouette*	
		No Vertex Program (fps)	*VP Speedup (percent)*	*No Vertex Program (fps)*	*VP Speedup (percent)*
iron	64x64x64	30.0	57.5%	29.9	54.3%
head	256x256x113	4.0	69.6%	3.5	57.3%
engine	256x256x128	4.0	90.5%	3.6	71.5%
leg	341x341x93	5.0	56.3%	4.6	45.9%
lobster	301x324x56	8.7	11.1%	7.5	10.0%
foot	256x256x256	5.9	54.7%	5.0	46.4%
aneurysm	256x256x256	15.1	34.1%	12.1	26.2%
bonsai	256x256x256	4.3	8.0%	3.9	8.0%
horse	13,352	60.39	0.0%	30.19	0.0%
low-res dragon	45106	30.4	5.4%	15.24	2.4%
bunny	69451	20.07	50.1%	11.89	1.2%
skeleton1	124018	12.08	71.0%	7.53	34.9%
skeleton2	522567	2.60	78.8%	1.50	29.3%
hand	654666	2.07	93.2%	1.32	42.4%
dragon	871414	1.50	99.3%	0.91	44.0%

our system on modern PCs. Performance results of our stipple system are presented in Table 2. These running times were gathered from a dual processor Intel Xeon 2.2 GHz computer with a Geforce 3 Ti 500 display card. The preprocessing time varies from seconds to a minute. The total number of point primitives in a typical volume data set ranges from 5,000 to 2,000,000 and the silhouette curves range from 1,000 to 300,000.

For surface models, silhouette rendering currently occupies a significant portion of the rendering time. Vertex programs can speed up the stipple rendering by up to 100 percent, depending on the model. Both of our silhouette techniques for surface objects enhance the quality of the final images and show a range of artistic styles that can be conveyed with the stipple rendering system. The frame rates can be improved by further reducing cache exchange and floating-point operations.

Nonetheless, the measured frame rate does provide the user with a level of interactivity necessary for exploring and illustrating various regions of interest within the volume data sets. Through the use of sliders, the user is able to quickly adjust the parameters to select the desired feature enhancement and its appropriate level. The user is able to rotate, translate, and zoom in or out of the object while maintaining consistent shading. The system has very good temporal rendering coherence with only very subtle temporal aliasing occurring during rotation near silhouette edges and illuminated boundaries as new points are added based on the silhouette and illumination enhancement factor. We have implemented a simple partial opacity point rendering to fade the points, alleviating this problem.

9 Conclusions and Future Work

We have developed an interactive stippling system that combines the advantages of point based rendering with the expressiveness of the stippling illustration style into an effective interactive illustration system for both volumes and surfaces, as can be seen in Fig. 17. This system utilizes techniques from both hand drawn illustration and volume rendering to create a powerful new environment in which to visualize and interact with data. Our system demonstrates that stippling effectively illustrates complex volume and surface data in a simple, informative manner that is valuable, especially for initial object investigation and data previewing. For these situations, the stipple renderer can be used to determine and illustrate regions of interest. These illustrated regions can then be highlighted when traditional rendering methods are later used for more detailed exploration and analysis. Initial feedback from medical researchers shows they are enthusiastic about the usefulness of the system for generating images for medical education and teaching anatomy and its relation to mathematics and geometry to children.

Many new capabilities have recently become available on modern graphics hardware that could significantly improve the performance of our system. Programmable vertex shaders allow us to move many of our feature enhancements onto the graphics card. This is especially true for those that are view dependent. Preprocessed points can be stored as display lists or vertex arrays in the graphics card's memory, which avoids the expensive vertex download each

Fig. 17. Head volume with silhouette, boundary, and distance enhancement and silhouette curves.

time a frame is rendered. Vertex programs can be used to evaluate the thresholds of feature enhancements by taking advantage of the fact that we are using vertices rather than polygons. Thresholding involves simple formulae and can be easily implemented in a vertex program. When a vertex falls below the enhancement threshold its coordinates can be modified to a position off screen, effectively culling it. This culling technique is not possible, in general, for polygons since there is currently no connectivity information available in vertex programs.

We plan to extend our work to improve the interactivity of the system and compare the performance to other NPR volume renderers to assess the effectiveness of using a point-based rendering system. We will continue to explore additional feature enhancement techniques. Additionally, it may be interesting to investigate the implementation of a stipple renderer using a texture-based volume rendering architecture which modulates the alpha values per-pixel in the fragment shader portion of the pipeline. The stipple volume renderer is available at: http://shay.ecn.purdue.edu/purpl/projects/dl_stipple.htm.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under grants: NSF ACI-0081581, NSF ACI-0121288, NSF IIS-0098443, NSF ACI-9978032, NSF MRI-9977218, NSF ACR-9978099, and the US Department of Energy's VIEWS program. Some data sets used in this paper come from the web page: http://www.gris.uni-tuebingen.de/areas/scivis/volren/datasets/datasets.html.

References

[1] E. Akelman, "Implicit Surface Painting," *Proc. Implicit Surfaces '98*, pp. 63-68, 1998.

[2] B. Csébfalvi and M.E. Gröller, "Interactive Volume Rendering Based on a 'Bubble Model'," *Proc. GI 2001*, pp. 209-216, June 2001.

[3] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M.E. Gröller, "Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering," *Computer Graphics Forum*, vol. 20, no. 3, pp. 452-460, Sept. 2001.

[4] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin, "Computer-Generated Watercolor," *Proc. SIGGRAPH 1997, Computer Graphics Proc., Ann. Conf. Series*, pp. 421-430, Aug. 1997.

[5] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte, "Floating Points: A Method for Computing Stipple Drawings," *Computer Graphics Forum*, vol. 19, no. 3, Aug. 2000.

[6] O. Deussen and T. Strothotte, "Computer-Generated Pen-and-Ink Illustration of Trees," *Proc. ACM SIGGRAPH 2000, Computer Graphics Proc., Ann. Conf. Series*, pp. 13-18, July 2000.

[7] D. Ebert and P. Rheingans, "Volume Illustration: Non-Photorealistic Rendering of Volume Models," *Proc. IEEE Visualization 2000*, pp. 195-202, Oct. 2000.

[8] B. Gooch and A. Gooch, *Non-Photorealistic Rendering*. A.K. Peters, 2001.

[9] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Proc. SIGGRAPH 98, Computer Graphics Proc., Ann. Conf. Series*, pp. 453-460, July 1998.

[10] *The Guild Handbook of Scientific Illustration*. E. Hodges, ed. John Wiley & Sons, 1989.

[11] K. Höhne and R. Bernstein, "Shading 3D-Images from CT Using Gray Level Gradients," *IEEE Trans. Medical Imaging*, vol. 5, no. 1, pp. 45-47, Oct. 1986.

[12] V. Interrante, "Illustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution," *Proc. SIGGRAPH '97, Computer Graphics Proc., Ann. Conf. Series*, pp. 109-116, Aug. 1997.

[13] S. Feiner, J. Foley, A. van Dam, and J. Hughes, *Computer Graphics, Principles and Practice*. Addison-Wesley, 1990.

[14] M. Levoy and T. Whitted, "The Use of Points as a Display Primitive," Technical Report 85-022, Univ. of North Carolina-Chapel Hill Computer Science Dept., Jan. 1985.

[15] A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen, "Non-Photorealistic Volume Rendering Using Stippling Techniques," *Proc. IEEE Visualization 2002*, pp. 211-218, Oct. 2002.

[16] A. Lu, J. Taylor, M. Hartner, D. Ebert, and C. Hansen, "Hardware-Accelerated Interactive Illustrative Stipple Drawing of Polygonal Objects," *Proc. VMV2002: Vision, Modeling, and Visualization*, pp. 61-68, Nov. 2002.

[17] E. Lum and K. Ma, "Hardware-Accelerated Parallel Non-Photorealistic Volume Rendering," *Proc. Second Int'l Symp. Non-Photorealistic Animation and Rendering*, pp. 67-ff, 2002.

[18] K. Ma and V. Interrante, "Extracting Feature Lines from 3D Unstructured Grids," *Proc. IEEE Visualization '97*, pp. 285-292, Nov. 1997.

[19] L. Mroz and H. Hauser, "RTVR—A Flexible Java Library for Interactive Volume Rendering," *Proc. IEEE Visualization 2001*, pp. 279-286, Oct. 2001.

[20] L. Neumann, B. Csébfalvi, A. König, and E. Gröller, "Gradient Estimation in Volume Data Using 4D Linear Regression," *Computer Graphics Forum*, vol. 19, no. 3, pp. 351-358, Aug. 2000.

[21] V. Ostromoukhov, "Digital Facial Engraving," *Proc. SIGGRAPH '99, Computer Graphics Proc., Ann. Conf. Series*, pp. 417-424, Aug. 1999.

[22] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface Elements as Rendering Primitives," *Proc. ACM SIGGRAPH 2000, Computer Graphics Proc., Ann. Con. Series*, pp. 335-342, July 2000.

[23] M. Pop, C. Duncan, G. Barequet, M. Goodrich, W. Huang, and S. Kumar, "Efficient Perspective-Accurate Silhouette Computation and Applications," *Proc. 17th Ann. Symp. Computational Geometry*, pp. 60-68, 2001.

[24] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-Time Hatching," *Proc. ACM SIGGRAPH 2001, Computer Graphics Proc., Ann. Conf. Series*, pp. 579-584, Aug. 2001.

[25] R. Raskar and M. Cohen, "Image Precision Silhouette Edges," *Proc. 1999 ACM Symp. Interactive 3D Graphics*, pp. 135-140, Apr. 1999.

[26] S. Rusinkiewicz and M. Levoy, "QSplat: A Multiresolution Point Rendering System for Large Meshes," *Proc. SIGGRAPH 2000*, pp. 343-352, 2000.

[27] M. Salisbury, C. Anderson, D. Lischinski, and D. Salesin, "Scale-Dependent Reproduction of Pen-and-Ink Illustrations," *Proc. SIGGRAPH '96, Computer Graphics Proc., Ann. Conf. Series*, pp. 461-468, Aug. 1996.

[28] M. Salisbury, M. Wong, J. Hughes, and D. Salesin, "Orientable Textures for Image-Based Pen-and-Ink Illustration," *Proc. SIGGRAPH '97, Computer Graphics Proc., Ann. Conf. Series*, pp. 401-406, Aug. 1997.

[29] P. Sander, X. Gu, S. Gortler, H. Hoppe, and J. Snyder, "Silhouette Clipping," *Proc. SIGGRAPH '00, Computer Graphics Proc., Ann. Conf. Series*, pp. 327-334, July 2000.

[30] S. Strassmann, "Hairy Brushes," *Proc. SIGGRAPH '86, Computer Graphics Proc., Ann. Conf. Series*, pp. 225-232, Aug. 1986.

[31] T. Strothotte and S. Schlechtweg, *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. San Francisco: Morgan Kaufmann, 2002.

[32] S. Treavett and M. Chen, "Pen-and-Ink Rendering in Volume Visualisation," *Proc. IEEE Visualization 2000*, pp. 203-210, Oct. 2000.

[33] S. Treavett, M. Chen, R. Satherley, and M. Jones, "Volumes of Expression: Artistic Modelling and Rendering of Volume Datasets," *Proc. Computer Graphics Int'l 2001*, pp. 99-106, July 2001.

[34] M. Visvalingam, "Sketch-Based Evaluation of Line Filtering Algorithms," *Proc. GI Science*, Oct. 2000.

[35] B. Wilson, K. Ma, and P. McCormick, "A Hardware-Assisted Hybrid Rendering Technique for Interactive Volume Visualization," *Proc. Volume Visualization and Graphics Symp. 2002*, Oct. 2002.

[36] G. Winkenbach and D. Salesin, "Computer-Generated Pen-and-Ink Illustration," *Proc. SIGGRAPH '94, Computer Graphics Proc., Ann. Conf. Series*, pp. 91-100, July 1994.

[37] G. Winkenbach and D. Salesin, "Rendering Parametric Surfaces in Pen and Ink," *Proc. SIGGRAPH '96, Computer Graphics Proc., Ann. Conf. Series*, pp. 469-476, Aug. 1996.
[38] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "EWA Volume Splatting," *Proc. IEEE Visualization 2001*, pp. 29-36, Oct. 2001.
[39] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface Splatting," *Proc. SIGGRAPH 2001, Computer Graphics Proc., Ann. Conf. Series*, pp. 371-378, Aug. 2001.

Aidong Lu received the BS and MS degrees in computer science from Tsinghua University in 1999 and 2001, respectively. Currently, she is a graduate student at Purdue University. Her research interests are computer graphics and visualization. She is a student member of the IEEE Computer Society.

Christopher J. Morris received the MS degree in computer science from the University of Maryland-Baltimore County in 2001, where his focus was in scientific visualization. Previously, he received the MS in mechanical engineering from Stanford University in 1998, where his interests included robotics and engineering design. He is an employee in the Visual Technology Group at the IBM T.J. Watson Research Center. His current research interests include volume rendering, nonphotorealistic rendering, and parallel systems.

Joe Taylor received the BS degree in computer engineering from Purdue University in 2001. He is currently a graduate student at Purdue University. His research interests are computer graphics and visualization.

David S. Ebert received the PhD degree from the Computer and Information Science Department at The Ohio State University in 1991. He is an associate professor in the School of Electrical and Computer Engineering at Purdue University. His research interests are scientific, medical, and information visualization, computer graphics, animation, and procedural techniques. Dr. Ebert performs research in volume rendering, nonphotorealistic visualization, minimally immersive visualization, realistic rendering, procedural texturing, modeling, and animation, modeling natural phenomena, and volumetric display software. He has also been very active in the graphics community, teaching courses, presenting papers, chairing the ACM SIGGRAPH '97 Sketches program, cochairing the IEEE Visualization '98 and '99 Papers program, serving on the ACM SIGGRAPH Executive Committe and serving as an associate editor for the *IEEE Transactions on Visualization and Computer Graphics*. He is a member of the IEEE and the IEEE Computer Society.

Charles Hansen received the BS degree in computer science from Memphis State University in 1981 and the PhD degree in computer science from the University of Utah in 1987. He is an associate professor of computer science at the University of Utah. From 1997 to 1999, he was a research associate professor in computer science at Utah. From 1989 to 1997, he was a technical staff member in the Advanced Computing Laboratory (ACL) located at Los Alamos National Laboratory, where he formed and directed the visualization efforts in the ACL. He was a Bourse de Chateaubriand PostDoc Fellow at INRIA, Rocquencourt, France, in 1987 and 1988. His research interests include large-scale scientific visualization, parallel computer graphics algorithms, massively parallel processing, 3D shape representation, and computer vision. He is a member of the IEEE and the IEEE Computer Society.

Penny Rheingans received the PhD degree in computer science from the University of North Carolina, Chapel Hill, and the AB degree in computer science from Harvard University. She is an assistant professor of computer science at the University of Maryland Baltimore County. Her current research interests include uncertainty in visualization, multivariate visualization, volume visualization, information visualization, perceptual and illustration issues in graphics, dynamic and interactive representations and interfaces, and the experimental validation of visualization techniques. She is a member of the IEEE Computer Society.

Mark Hartner graduated in the summer of 2002 with the BS degree in computer engineering. He is a researcher for the Scientific Computing and Imaging Institute at the University of Utah. His primary interests include high performance computing and scientific visualization.