Non-Photorealistic Volume Rendering Using Stippling Techniques

Aidong Lu* Purdue University
Christopher J. Morris† IBM TJ Watson Research Center
David S. Ebert‡ Purdue University
Penny Rheingans§ University of Maryland-Baltimore County
Charles Hansen¶ University of Utah

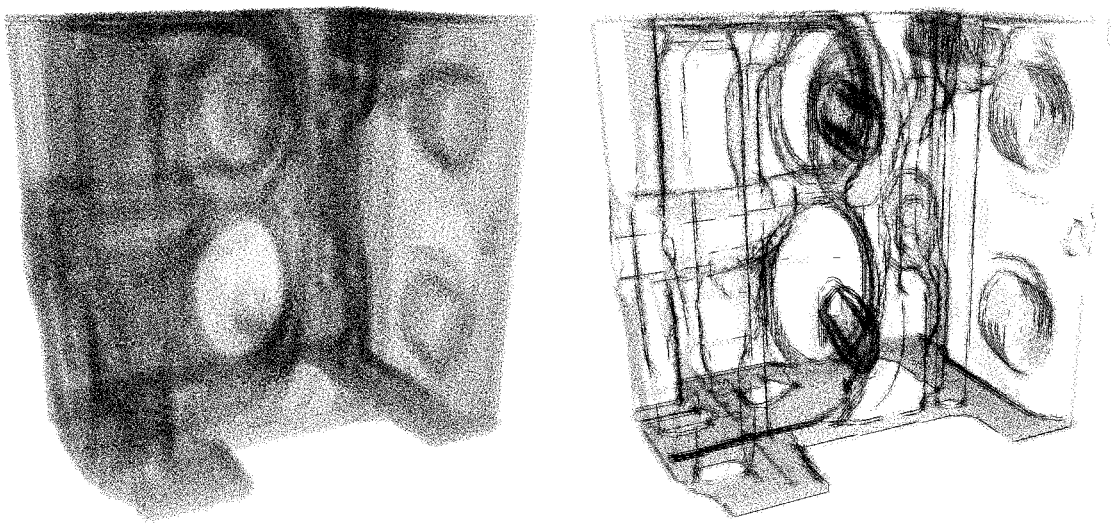


Figure 1: Engine block rendered with volume stipple renderer.($k_{gc} = 0.39, k_{gs} = 0.51, k_{ge} = 1.0$). (a) is the default rendering and (b) shows boundary and silhouette enhancement, as well as silhouette curves.($k_{sc} = 0.279, k_{ss} = 0.45, k_{se} = 1.11$)

ABSTRACT

Simulating hand-drawn illustration techniques can succinctly express information in a manner that is communicative and informative. We present a framework for an interactive direct volume illustration system that simulates traditional stipple drawing. By combining the principles of artistic and scientific illustration, we explore several feature enhancement techniques to create effective, interactive visualizations of scientific and medical datasets. 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 datasets in a concise, meaningful, and illustrative manner. Volume stippling is effective for many applications and provides a quick and efficient method to investigate volume models.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—color, shading, and texture

Keywords: non-photorealistic rendering, volume rendering, scientific visualization, medical imaging

*e-mail: alu@ecn.purdue.edu
†e-mail:cjmorris@us.ibm.com
‡e-mail:ebertd@ecn.purdue.edu
§email:rheingan@cs.umbc.edu
¶email:hansen@cs.utah.edu

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 to display 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 [31]. This selective inclusion of detail enables illustrations to be more expressive than photographs.

Indeed, many natural science and medical publications use scientific illustrations in place of photographs because of the illustrations' educational and communicative ability [10]. Illustrations take advantage of human visual acuity and can represent a large amount of information in a relatively succinct manner, as shown in Figures 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 achieve 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

Figure 2: Idol stipple drawing by George Robert Lewis [10].

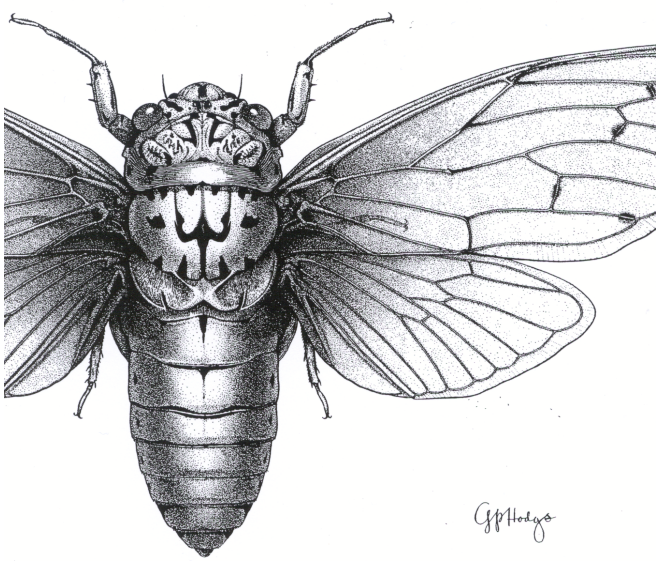


Figure 3: Cicadidae stipple drawing by Gerald P. Hodge [10].

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. When initially exploring an unknown volume dataset, 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 another volume rendering technique to generate a more detailed image of these structures. It is the use of non-photorealistic rendering (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, 27]. Various approaches have been used, including pen-and-ink illustration, silhouette edges, and stroke textures. Most of the research in the field of non-photorealistic illustration has concentrated on strokes, crosshatching, and pen and ink techniques [9, 14, 26] and most of the current research still concentrates on surface renderings, which requires surface geometry. We chose to directly render volume datasets without any additional analysis of object or structure relationships within the volume. Volume stippling not only maintains all the advantages of NPR, but it also makes interactive rendering and illustration feasible on useful-sized datasets because of two attributes of points: fast rendering speed and innate simplicity.

In our system, the volume resolution is initially adjusted for optimum stipple pattern rendering, and point lists are generated corresponding to the gradient magnitude and direction. Next, 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, datasets can be rendered in different illustration styles.

2 Related Work

Non-photorealistic rendering 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 [30], pen-and-ink illustration [6, 23, 24, 31],silhouette rendering [14, 19, 21, 25], 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. [20] 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 [17] 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 et al. [9] presented a method for creating an image with a hand painted appearance from a photograph and an approach to designing styles of illustration. They 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 [32] presented algorithms and techniques for rendering parametric free-form surfaces in pen and ink.

Deussen et al. [5] used points for computer generated pen-and-ink illustrations in simulating the traditional stipple drawing style. Their method is to first render polygonal models into a continuous tone image and then convert these target images into a stipple representation. They can illustrate complex surfaces very vividly, but their method is for surface rendering, not 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 and Chen also used

illustration techniques to render surfaces within volumes [28, 29]. 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 et al. [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 yet. Furthermore, interactive volume rendering has garnered a significant amount of attention [15] and NPR methods have been applied to obtain interactive performance while producing effective volume renderings [2, 3]. Treavett et al. [29] 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 been used as rendering primitives before. Levoy and Whitted [13] 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 [18, 34].

The main difference between previous stipple and point rendering research and ours is that our system interactively renders volumes with points instead of just surfaces with points. Within volume rendering, the closest related technique is splatting [33, 22], which traditionally does not incorporate the effectiveness of illustration techniques. In the remainder of this paper, we show the effectiveness of a simple point-based interactive volume stippling system and describe how a number of illustrative enhancement techniques can be utilized to quickly convey important volume characteristics for rapid previewing and investigation of volume models.

3 The Stipple Volume Renderer

The clarity and smoothness displayed by stippling, coupled with the speed of hardware point rendering, makes volume stippling an effective tool for illustrative rendering of volume data. 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 renderer and visually extracts various features of the data by selective enhancement of certain regions. 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 illustrate these volumes using stippling techniques with a particular set of features in mind. To effectively generate renderings of volume datasets 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 volume based on volume characteristics, including gradient properties and basic resolution requirements. 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, the initial estimation of stipple density from volume resolution, and the generation of an initial point distribution. Furthermore, the voxel values and gradients are all normalized.

4.1 Gradient Processing

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. Initially, we tried a traditional central difference gradient method. However, Neumann et al. [16] have presented an 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 dataset and estimated gradient at the same time. We have implemented their method for better gradient estimation.

4.2 Initial Resolution Adjustment

When viewing an entire volume dataset, as the volumes' size increases, each voxel's screen projection is reduced. Even if we assign at most one point per voxel, areas with high gradient magnitude still appear too dark. We use a simple box-filter to initially adjust the volume resolution, so that the average projection of a voxel on the screen is at least 5x5 pixels. This adjustment improves the stippling pattern in the resulting images.

We define N_{max} as the maximum number of stipples that each voxel can contain during the rendering process. After reading the dataset, we approximately calculate the maximum projection of a voxel on the screen and set the maximum number of points in the voxel to be equal to the number of pixels in the voxel projection. This reduces redundant point generation and increases the performance of the system. The following formula is used:

$$N_{max} = k_{max} * A_{vol}/(X_{res} * Y_{res} * Z_{res})^{2/3} \qquad (1)$$

where A_{vol} is the rendered area, k_{max} is a scaling factor, and the volume has resolution $X_{res} \times Y_{res} \times Z_{res}$. This is a heuristic formula because the scale of the X, Y and Z axes are not ordinarily the same. Figure 4 shows several resolutions of a dataset. In each case, most of the details of the dataset are preserved.

4.3 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 problems with frame-to-frame coherence. To alleviate this problem, we approximate a Poisson disc distribution to initially position a maximum number of stipples. According to the statistics of the gradient magnitude distribution, we generate stipples near the

gradient plane for the voxels whose gradient magnitude is above a user specified threshold. 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. 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. 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 or their respective size. We always select the stipples that will be drawn from a pre-generated list of stipples for each voxel, therefore, maintaining frame-to-frame coherence for the points.

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 voxel: whether it is part of a boundary or silhouette, its spatial position in relation to both the entire volume 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 voxel has the following information stored in a data structure:

- number of points
- gradient
- voxel scalar data value
- point size
- 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 datasets, we select a different combination of feature enhancements to achieve the best effect.

The basic formula for the point count per voxel, N_i, is the following:

$$N_i = N_{max} * T \tag{2}$$

where N_{max} is the maximum number of points a voxel can contain, calculated according to Equation 1 from the volume's projected screen resolution, and

$$T = T_b * T_s * T_r * T_d * T_t * T_l \tag{3}$$

T_b, T_s, T_r, T_d, T_t, and T_l are the boundary, silhouette, resolution, distance, interior transparency, and lighting factors, respectively, described in the following sections. Each factor is normalized in the range of zero to one. If some feature enhancements are not selected, the corresponding factors will not be included in Equation (3).

Besides the point count of a voxel, the point size is also an important factor to increase visualization quality. The basic point size of a voxel is calculated by the following equation:

$$S_i = \|\nabla \vec{V_i}\| * S_{max} \tag{4}$$

where S_{max} is a user specified maximum point size. Voxels with larger gradient magnitude contain larger points, achieving the effect of smooth point size changes within the volume. The point size for each voxel is calculated in a manner similar to Equation 2.

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 are determined using volume illustration techniques [7] . The boundary enhancement factor T_b for a voxel at location P_i is determined from the original voxel scalar value, v_i and the voxel value gradient magnitude $\|\nabla \vec{V_i}\|$ using the following formula:

$$T_b = v_i * (k_{gc} + k_{gs} * (\|\nabla \vec{V_i}\|^{k_{ge}})) \tag{5}$$

where k_{gc} controls the direct influence of the voxel value, k_{gs} indicates the maximum boundary enhancement, and k_{ge} controls the sharpness of the boundary enhancement. By making the stipple placement denser in voxels of high gradient, boundary features are selectively enhanced. This feature extraction can be further improved with silhouette enhancement techniques.

In manual stipple drawings, the highest concentration of stipples is usually in areas oriented orthogonally to the view plane, forming the silhouette edge. The silhouette enhancement factor T_s is constructed in a manner similar to the boundary enhancement factor. The parameters k_{sc}, k_{ss}, and k_{se} are controlled by the user to adjust each part's contribution, as shown in the following formula:

$$T_s = v_i * (k_{sc} + k_{ss} * (1 - (|\nabla \vec{V_i} \cdot \vec{E}|))^{k_{se}}) \tag{6}$$

where $\vec{E}$ is 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. We render more points on and inside the volume boundaries and can, consequently, incorporate light and transparency information to more effectively enhance the rendering. Figure 1(b) shows the engine block volume 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 and produces the effect that the features become larger and clearer when the volume moves closer to the viewpoint. It also helps increase rendering performance by eliminating unnecessary rendering of distant points. In order to implement resolution enhancement, we use the following formula:

$$T_r = [\frac{(D_{near} + d_i)}{(D_{near} + d_0)}]^{k_{re}} \tag{7}$$

where D_{near} is the location of the near plane, d_i is the distance from the current location of the volume to the near plane, d_0 is the distance from the initial location of the volume to the near plane (we use this position as the reference point), and k_{re} controls the rate of change of this resolution enhancement. When k_{re} equals 0, there is no enhancement. The bigger the value, the more obvious the effect. The point size also varies with the change in resolution

so that point sizes are small when the resolution is low and large when resolution is high. In Figure 4, the same model is viewed at three different distances, but the resulting stipple density is the same for each.

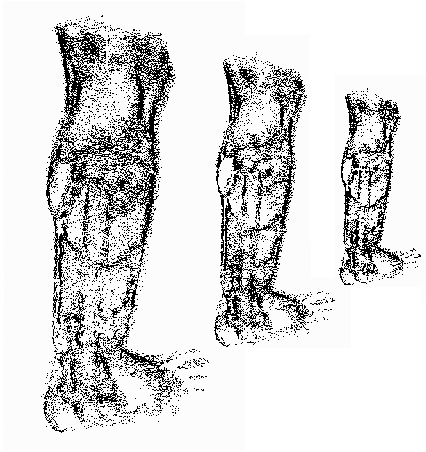


Figure 4: Resolution enhancement of the leg volume dataset. (k_{re}=1.1)

5.3 Distance

In resolution enhancement, we use the location of the whole volume in the scene. The location of different volume elements within the overall volume presents a different challenge. Distance is an important factor that helps us understand the relationship between elements within the volume. As in traditional illustration, we can enhance depth perception by using the position of a voxel within the volume box to generate a factor that modifies both the point count and the size of the points. We use a linear equation with different powers k_{de} to express the function of the distance attenuation to generate this factor via the following equation:

$$T_d = 1 + (\frac{z}{a})^{k_{de}} \tag{8}$$

Where $(-a, a)$ is the original distance range in the volume, z is the depth of the current voxel, and k_{de} controls the contribution of the distance attenuation for each voxel. k_{de} may change from negative to positive to enhance different parts in the volume. Figure 5 shows an example of distance attenuation. Comparing this image to that in Figure 1(b), 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 details 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. In our system, the number of points varies based on the gradient magnitude of a voxel to the center of the volume, thus achieving a better transparency effect:

$$T_t = \|\nabla \vec{V_i}\|^{k_{te}} \tag{9}$$

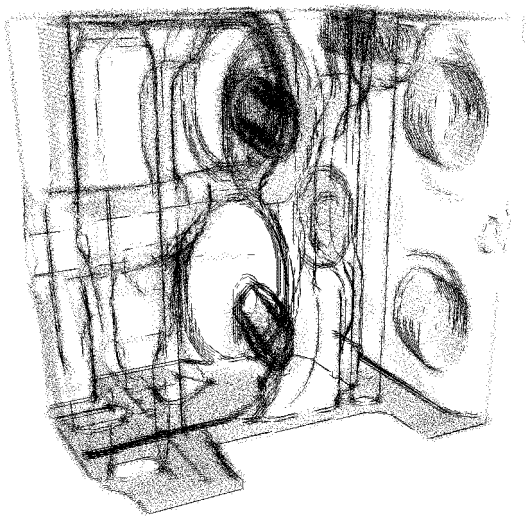


Figure 5: Distance attenuation of the engine block volume.($k_{de} = 1.5$)

k_{te} controls the falloff of the transparency enhancement. With this equation, the voxels with lower gradient magnitude become more transparent. In addition, point sizes are adjusted by the transparency factor. In Figure 6, 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.

5.5 Lighting

Achieving compelling lighting effects within the stipple renderer presents several challenges. First, when using noisy volumes, the gradients are not adequate for expressing the shadings of some structures correctly. Second, because structures often overlap in the volume, it can still be difficult to identify to which structure a point belongs in complex scenes. Finally, 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. Therefore, when lighting the volume, only the front oriented voxels (where the gradient direction is within ninety degrees of the eye direction) are rendered. The following equation is used to generate a factor to modify the point count of the voxels:

$$T_l = 1 - (\vec{L} \cdot \nabla \vec{V_i})^{k_{le}} \tag{10}$$

where $\vec{L}$ is the light direction and k_{le} controls the contribution of the light.

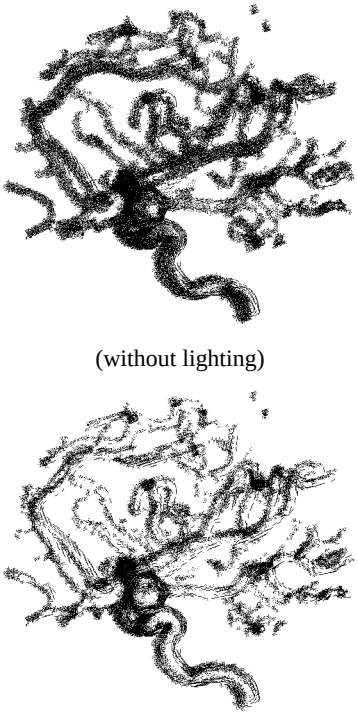
In Figure 7, light is projected from the upper right corner in the image and smaller vessels have been removed with parameter settings to better highlight the shading of the main features. The main structures and their orientation are much clearer with the application of lighting effects.

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. Searching for potential silhouette curves in the vicinity of each voxel could easily create a performance bottleneck by requiring a search in, at least,

(without interior enhancement) (with interior enhancement)

Figure 6: Stipple rendering of bonsai tree volume.($k_{te} = 0.5$)



(without lighting)

(with lighting)

Figure 7: Stipple rendering of the aneurysm volume.($k_{le} = 2.7$)

the 3x3x3 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 the following criteria:

1. $V_i * LOG(V_i) < Th_{LOG}$
2. $(\vec{E} \cdot \nabla \vec{V_i}) < Th_{eye}$
3. $\|\nabla \vec{V_i}\| > Th_{grad}$

where $\vec{E}$ is the eye (or view) vector, and Th_{LOG}, Th_{eye}, and Th_{grad} are user controllable threshold values.

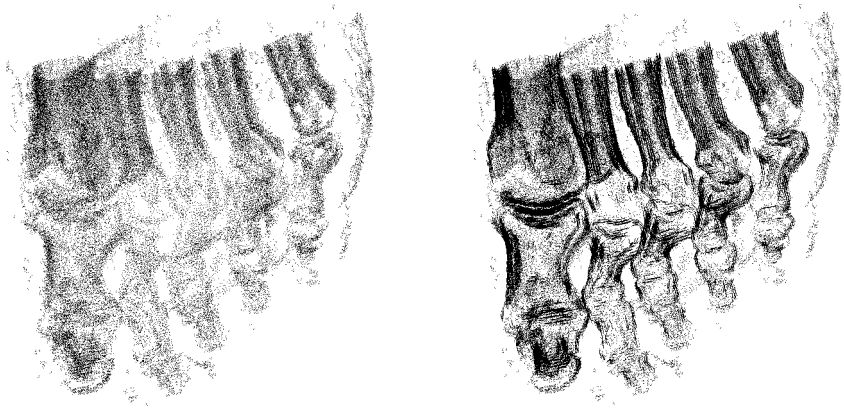
To "sketch" silhouette curves, the voxels that satisfy the above conditions have a line segment drawn through the center of the voxel in the direction of $\nabla \vec{V_i} \times \vec{E}$. Silhouette curves can be rendered at 20 to 30 frames per second and significantly improve image quality. Figure 8 shows the effectiveness of silhouette curves in highlighting structures, especially the bones of the foot dataset.

6 PERFORMANCE

We are able to interactively render reasonably-sized volume datasets using illustrative enhancement with our system on modern PCs. The total number of point primitives in a typical data set ranges from 5,000 to 2,000,000, and the silhouette curves range from 1,000 to 300,000. Performance results of our stipple system are presented in Table 1. These running times were gathered from a dual processor Intel Xeon 2.00 GHz computer with a Geforce 3 Ti 500 display card. The preprocessing time varies from seconds to a minute. 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 datasets. 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 volume 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 which alleviates this problem.

Dataset	*resolution*	*stipples*	*silhouettes*	*both*
iron	64x64x64	30.0	60.4	29.9
head	256x256x113	4.0	26.1	3.5
engine	256x256x128	4.0	20.3	3.6
leg	341x341x93	5.0	30.5	4.6
lobster	301x324x56	8.7	30.5	7.5
foot	256x256x256	5.9	30.5	5.0
aneurysm	256x256x256	15.1	30.5	12.1
bonsai	256x256x256	4.3	20.6	3.9

Table 1: Running times (frames per second) for separate rendering techniques.



(without silhouette curves) (with silhouette curves)

Figure 8: Stipple rendering of the foot volume.($Th_{eye} = 0.0, Th_{grad} = 0.2, Th_{LOG} = 0.12$)

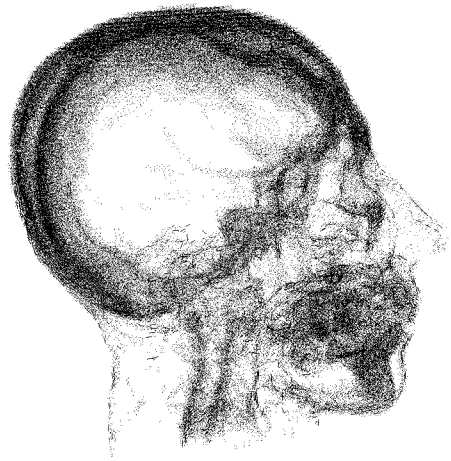


Figure 9: Head volume with silhouette, boundary, and distance enhancement and silhouette curves.($k_{gc} = 0.4, k_{gs} = 0.0, k_{ge} = 10.0, k_{sc} = 0.5, k_{ss} = 5.0, k_{se} = 1.0; Th_{eye} = 0.9, Th_{grad} = 0.2, Th_{LOG} = 0.22$)

7 Conclusions and Future Work

We have developed an interactive volumetric stippling system that combines the advantages of point based rendering with the expressiveness of the stippling illustration style into an effective interactive volume illustration system, as can be seen above in Figure 9.

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 volume data. Our system demonstrates that volumetric stippling effectively illustrates complex volume data in a simple, informative manner that is valuable, especially for initial volume investigation and data previewing. For these situations, the volume stipple renderer can be used to determine and illustrate regions of interest. These regions can then be highlighted as important areas when traditional rendering methods are later used for more detailed exploration and analysis. Initial feedback from medical researchers is very positive. 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 shades can 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 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. Titleholding 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. Furthermore, we will continue to explore additional feature enhancement techniques. Though it was too soon to include the results in this paper, initial investigation into using color within our stipple rendering system has already begun. 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.

8 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants: NSF ACI-0081581, NSF ACI-0121288, NSF IIS-0098443, NSF ACI-9978032, 9977218, 9978099, and DOE VIEWS program.

References

[1] E. Akelman. Implicit surface painting. In *Proceedings of Implicit Surfaces Conference 1998*, pages 63–68, 1998.

[2] B. Csébfalvi and M. Gröller. Interactive volume rendering based on a "bubble model". In *GI 2001*, pages 209–216, June 2001.

[3] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460, September 2001.

[4] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. Salesin. Computer-generated watercolor. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 421–430, August 1997.

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

[6] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 13–18, July 2000.

[7] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *IEEE Visualization 2000*, pages 195–202, October 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. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 453–460, July 1998.

[10] E. Hodges(editor). *The Guild Handbook of Scientific Illustration*. John Wiley and Sons, 1989.

[11] K. Höhne and R. Bernstein. Shading 3D-images from CT using gray level gradients. *IEEE Transactions on Medical Imaging*, 5(1):45–47, October 1986.

[12] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 109–116, August 1997.

[13] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report 85-022, University of North Carolina-Chapel Hill Computer Science Department, January 1985.

[14] K. Ma and V. Interrante. Extracting feature lines from 3D unstructured grids. In *IEEE Visualization '97*, pages 285–292, November 1997.

[15] L. Mroz and H. Hauser. RTVR - a flexible java library for interactive volume rendering. In *IEEE Visualization 2001*, pages 279–286, San Diego, CA, October 2001.

[16] L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4D linear regression. *Computer Graphics Forum*, 19(3):351–358, August 2000.

[17] V. Ostromoukhov. Digital facial engraving. In *Proceedings of SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 417–424, August 1999.

[18] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 335–342, July 2000.

[19] M. Pop, C. Duncan, G. Barequet, M. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation and applications. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, pages 60–68, 2001.

[20] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 579–584, August 2001.

[21] R. Raskar and M. Cohen. Image precision silhouette edges. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 135–140, April 1999.

[22] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 343–352, 2000.

[23] M. Salisbury, C. Anderson, D. Lischinski, and D. H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In *Proceedings of SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 461–468, August 1996.

[24] M. Salisbury, M. Wong, J. Hughes, and D. Salesin. Orientable textures for image-based pen-and-ink illustration. In *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 401–406, August 1997.

[25] P. Sander, X. Gu, S. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 327–334, July 2000.

[26] S. Strassmann. Hairy brushes. In *Proceedings of SIGGRAPH 1986*, Computer Graphics Proceedings, Annual Conference Series, pages 225–232, August 1986.

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

[28] S. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *IEEE Visualization 2000*, pages 203–210, October 2000.

[29] S. Treavett, M. Chen, R. Satherley, and M. Jones. Volumes of expression: Artistic modelling and rendering of volume datasets. In *Computer Graphics International 2001*, pages 99–106, July 2001.

[30] M. Visvalingam. Sketch-based evaluation of line filtering algorithms. In *Proceedings of GI Science*, Savannah, USA, October 2000.

[31] G. Winkenbach and D. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 1994*, Computer Graphics Proceedings, Annual Conference Series, pages 91–100, July 1994.

[32] G. Winkenbach and D. Salesin. Rendering parametric surfaces in pen and ink. In *Proceedings of SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 469–476, August 1996.

[33] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA volume splatting. In *IEEE Visualization 2001*, pages 29–36, San Diego, CA, October 2001.

[34] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 371–378, August 2001.