Interactive NPR

Eric B. Lum and Kwan-Liu Ma
University of California at Davis

1 Introduction

This tutorial surveys the state-of-the-art techniques for interactive non-photorealistic rendering with an emphasis on applications to volume visualization. This first section begins by giving a general motivation behind the need for interactivity in NPR and discusses some of the general approaches toward achieving high-speed rendering. The second section surveys methods that have been developed for interactive non-photorealistic rendering of polygonal surfaces, the area where the vast majority of the research in the field has been devoted. The third section describes an approach for how non-photorealistic rendering can be applied to interactive volume visualization, while the fourth section concludes this tutorial.

1.1 Motivation

Interactivity is often associated with spatial exploration, where parameters such as position, zoom and light direction can be varied over time. The resulting animations controlled by the user allow further insight to be gained in the subject being viewed. Much of the interest in non-photorealistic rendering relates to its use in enhancing spatial structures and clarifying shape. Thus, combining interactive methods with NPR is a natural progression since the combination of the two can be even more effective for illustrating shape.

Non-photorealistic rendering techniques typically have some set of rendering parameters associated with them that vary the style of the resulting images. The required tuning of these parameters does not make these algorithms less desirable, but rather are essential for giving the user the tools necessary for creating the types of images they desire. In the case of scientific visualization, there is rarely any single set of ”correct” rendering parameters as seen in the two images shown in Figure 1. Rendering parameters should be selected that add emphasis and clarity to the aspects of the visualization the user is interested in.

Often, the user of non-photorealistic rendering software is not an artist. They might, for example, be a scientist who would like to generate an image that illustrates a particular structure they are studying. The user might not know which non-photorealistic rendering techniques are appropriate, or might not even have a clear vision of how they would like the resulting visualization to look. If, however, the

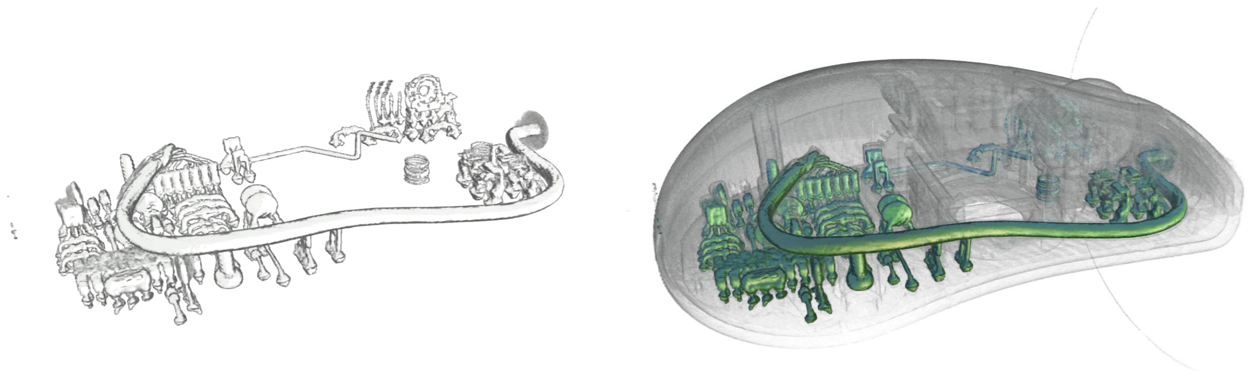


Figure 1: Two non-photorealistic volume renderings of a mouse data set with different sets of rendering parameters. Interactive rendering allows users to select rendering parameters appropriate for their application

tuning of rendering parameters is interactive, the user can quickly explore the parameter space to find an appropriate set of values.

This selection of parameters is an iterative process, where the user changes a parameter, views the result, and then changes parameters further. If the users can see the results of changes in sub-second times they will be much more effective selecting appropriate parameters for their task. Thus, changes in view and object position are only one aspect of interactive rendering. It is equally important to give the user the interactivity necessary to explore the rendering parameter space.

Unfortunately, many factors make interactive non-photorealistic rendering difficult. First, the addition of non-photorealistic rendering techniques typically adds to the amount of calculation required in the rendering process. As a simple example, silhouette edge rendering requires the additional calculations associated with silhouette extraction. Furthermore, the standard technique of rendering data at lower resolution or rendering to a lower resolution window to achieve interactivity is often not suited to the needs of the user when selecting non-photorealistic rendering parameters. Non-photorealistic rendering can be used effectively to clarify fine structures. In order to specify rendering parameters optimized for viewing these structures, objects must be rendered at a high enough resolution for them to be visible.

1.2 Achieving Interactivity

A number of techniques have been developed to facilitate interactive non-photorealistic rendering. Several of these approaches are software based and use data structures that allow for the efficient extraction of features to be illustrated non-photorealistically. To achieve interactivity there has been a trend toward the use of consumer PC graphics cards, many of which have capabilities exceeding those found in high end workstations just a few of years ago. With each product iteration these cards have improved performance, increased memory capacity, and new features supported in hardware. More recent interactive non-photorealistic rendering algorithms load data (geometry and textures) into video memory and then perform rendering calculations on the graphics card to exploit the high performance memory and processor found on the card, and avoid the I/O costs of sending large amounts of data from main memory across the graphics bus to the graphics card for every frame. Current consumer graphics cards have up to 128 megabytes of memory and support programmable vertex operations as well as the customizable combining of textures.

Another method to achieve interactivity is the use of clusters of commodity PCs for relatively low cost high performance rendering of large amounts of data [LM02] [LMC02]. In particular, by using the aggregated CPU performance, memory capacity, and I/O of a PC cluster with consumer graphics cards, scalable algorithms exist to render large problems at higher frame rates. The use of a PC cluster can still present a number challenges related to how to deal with the relatively slow communication between nodes and how to efficiently distributing the rendering task such that the processing load is well balanced among nodes.

2 Non-Photorealistic Surface Rendering

Most of the research in interactive non-photorealistic rendering has been focussed on the display of polygonal surfaces. These include methods for silhouette edge extraction and rendering, non-photorealistic illumination, and the illustration of surfaces with textured hatching or brush strokes.

2.1 Silhouette Edge Rendering

Silhouettes edges have been utilized for the illustration of surfaces [SABS94, IFP95] since dark lines drawn around an object can be effective in showing an objects structure. These lines can also be used to help indicate an object's spatial relationship with other objects that consist of similar material since dark edges drawn around overlapping objects can provide depth cues with respect to their surfaces.

A number of interactive software techniques have been developed for silhouette edge extraction and rendering. Markosian et al. [MKT$^+$97] describe an interactive probabilistic approach for silhouette extraction using a variation of Appel's hidden-line algorithm. Gooch et al. [GSG$^+$99] present a method which uses an edge structure based on the direction of the neighboring face normals. In a preprocessing step, edges are stored as a hierarchy of arcs on a Gauss map that can be used to extract silhouettes in logarithmic time in the number of edges for smooth models. Since this approach is software based, it also gives them flexibility in changing line styles. An example of an image rendered using their technique is shown in Figure 2. Elber [Elb99] describes a method for line art rendering that uses line art strokes that are generated in a preprocessing step and placed in a data structure for quick extraction during rendering based on view and light source direction. Strokes are used for both hatching and silhouette rendering and are grouped based on surface normal direction. Buchanan and Sousa [BS00] as well as Lake et al. [LMHB00] present software methods for interactive silhouette extraction by setting flags for each edge based on the front/back face properties of the neighboring faces. Hertzmann and Zorin [HZ00] present an approach that uses the concept of dual surfaces for interactive silhouette detection. Their deterministic method works for both orthographic and perspective projection and avoids complete traversal of all edges of the mesh. Tests they conducted suggest performance to be roughly linear to the number of silhouette triangles.

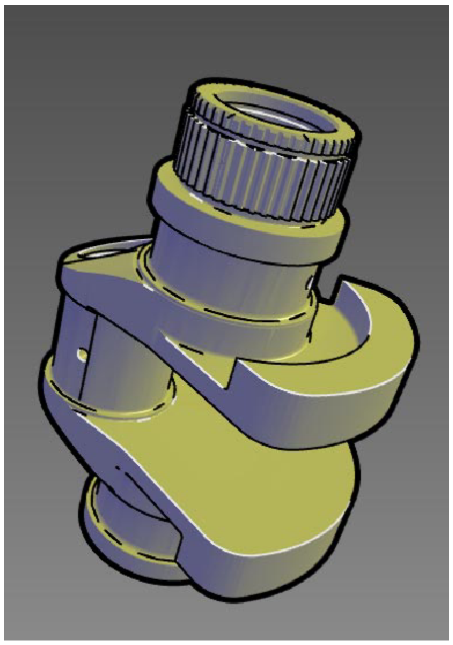


Figure 2: Result from Gooch et al. [GSG+99] showing interactively rendered technical illustration with non-photorealistic silhouette edges and shading. Their approach uses a Gauss map to efficiently extract silhouette edges that are rendered in varying line styles.

Several methods have also been developed that utilize graphics hardware for silhouette edge rendering. Gooch et al. [GSG+99] describe how environment mapping can be used for single pass hardware accelerated silhouette edge rendering. This is accomplished by using an environment map that is dark in regions perpendicular to the view direction. They note that the resulting silhouettes are artistically interesting but can be inappropriate for technical illustration since line thickness is nonuniform. Raskar and Cohen [RC99] render silhouette edges in hardware using multi-pass rendering. They draw all polygons once with z-buffer on, and then draw all back-facing in a second pass as either wireframe with z-buffer equality or as filled polygons that have been moved toward the camera in the second pass. In following work, Raskar [Ras01] presents a method for the rendering of ridges, valleys and silhouettes edges by rendering extra polygons for each edge that are generated on the graphics card using programmable vertex operations.

Northrup and Markosian [NM00] describe a screen/object space hybrid approach for silhouette rendering where silhouette edges are detected in object space, but viability and adjacency is determined in screen space. This allows for the linking of silhouette edge segments into smooth curves that are rendered in various styles interactively controlled by the user. An example that demonstrates the artistic silhouette edges produced by their technique is shown in Figure 3.

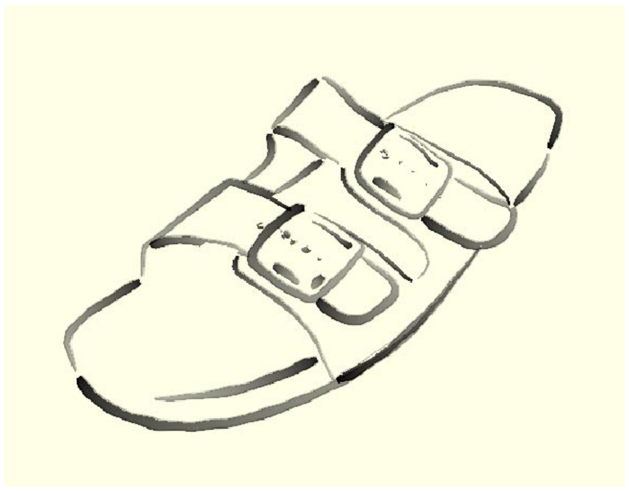


Figure 3: Result from Northrup and Markosian [NM00] that demonstrates their artistic silhouette edge rendering. Their hybrid object/screen space method links silhouette edges into curves that are rendered in various styles.

2.2 Non-Photorealistic Illumination

Lighting can be extremely effective in conveying the shape and structure of an object. Artists often illustrate lighting through not only the varying of pigment value (intensity) but also through the variation of color temperature [Mac99]. Directly illuminated objects are represented with warmer colors which include yellow, orange and red. Gooch et al. [GGSC98] use cool to warm shading for the illumination of surfaces. They present a shading model which allows extreme changes in color value to be reserved for outlines and highlights and describe how an approximation of their model can be implemented using graphics hardware that supports Phong shading. Sloan et al. [SMGG01] present methods that allow users to interactively specify their own non-photorealistic shading models. Their system provides real-time feedback which gives users the ability to explore a wide range of artistic styles.

Lum and Ma [LM01] describe an inverted non-photorealistic illumination model that renders illuminated objects by building up layers of simulated pigment based on the surface darkness. By adding paint to a simulated white canvas where objects are dark, they achieve color coherence similar to that which would appear from the mixing of a limited palette of paints. However, their implementation uses ray-tracing and is therefore not fully interactive at higher screen resolutions.

2.3 Non-Photorealistic Texture

Textured surfaces in the form of either brush strokes or hatching strokes can help to clarify shape and indicate material properties. Klein et al. [KLK$^+$00] describe what they call ”art maps”, which consist of multi-resolution mipmapped textures with constant sized brush strokes. This permits rendering in hardware with brush stroke textures that do not vary in size regardless of a polygon’s projected size in screen space.

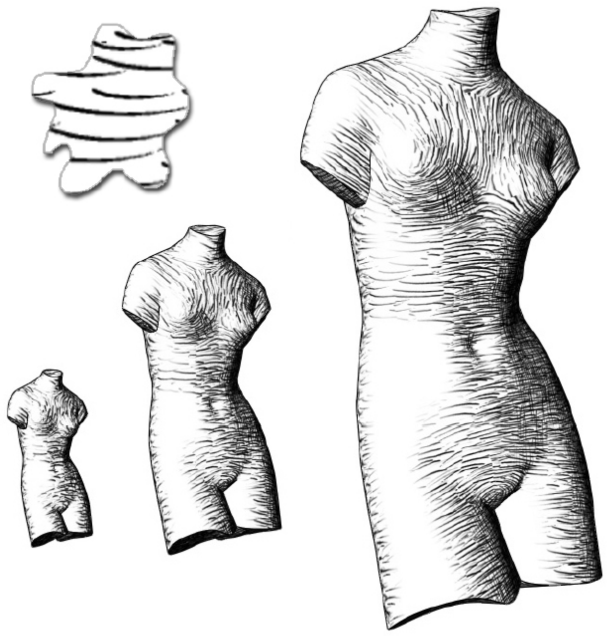


Figure 4: Result from Praun et al. [PHWF01] showing Venus model rendered using tonal art maps(TAM). Notice that hatch strokes do not change in size as the size of the model in projected screen space is reduced.

Praun et al. [PHWF01] extend this technique with mipmapped textures they call ”tonal art maps” (TAM) which contain pre-rendered hatched textures of varying density(tone) and resolution. In order to preserve tone coherence between tone levels, lighter levels contain subsets of darker leveled tones. By utilizing the R,G,B color channels of each texture, they are able to perform real-time hatching with six grayscale tone levels using a single rendering pass with two texture units. Hatching direction is oriented along the principal curvature directions of a surface such that stroke directions follow the shape of an object using principal curvature directions as illustrated in Figure 4. Since the lighter tone levels are subsets or darker levels, and the TAMs are applied in object space, the resulting animation exhibits frame-to-frame coherence and avoids the ”shower-door” effect. In further work by Webb et al. [WPFH02], two methods for finer control of tone using newer features found on consumer PC graphics hardware are presented. The first technique utilizes volumetric textures, where tone density is varied over the third dimension. The second method uses pixel shaders to provide a per-pixel lighting texture thresholding for reduced aliasing. Examples of images rendered using each of these techniques are shown in Figure 5. In some ways similar to TAMs, Lake et al. [LMHB00] describe pencil sketch shading using multiple textures of varying pencil stroke density. The textures can be applied in screen space for a hand-drawn appearance as seen in Figure 6 or the textures can have fixed object space coordinates for better frame-to-frame coherence.

Majumder and Gopi [MG02] present a technique for using graphics hardware accelerated charcoal rendering. They introduce what they call the ”contrast enhancement operator” (CEO) which modifies lighting contrast to account for the limited dynamic range found in charcoal drawings. Charcoal is

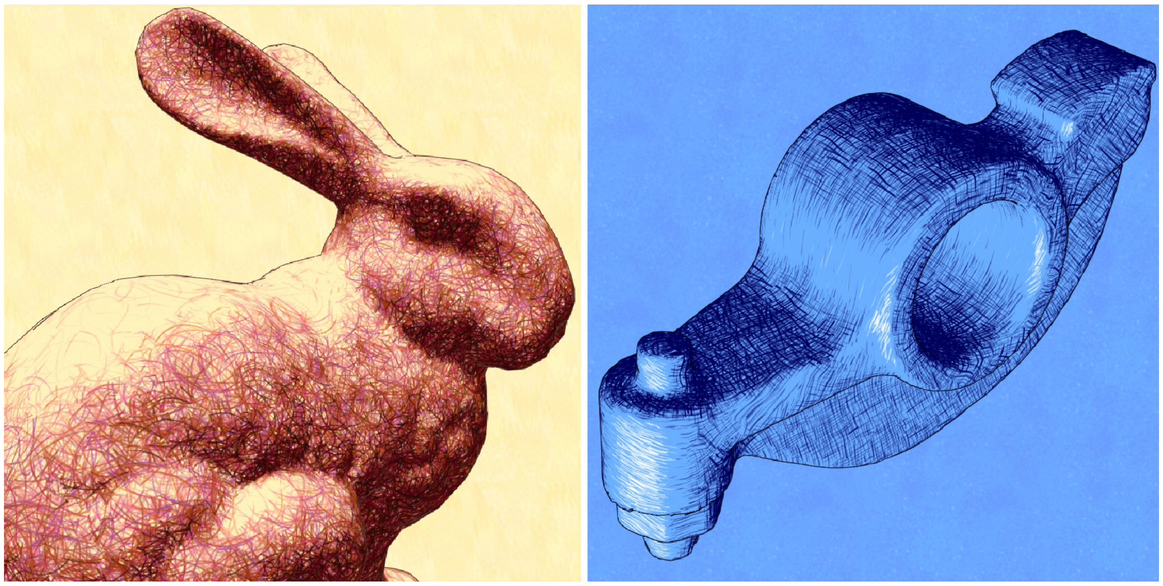


Figure 5: Results from Webb et al. [WPFH02] that show tonal art maps with fine tone control. Left image shows bunny model rendered using their volumetric texture technique, while the right image of a rocker arm uses their pixel shader method.

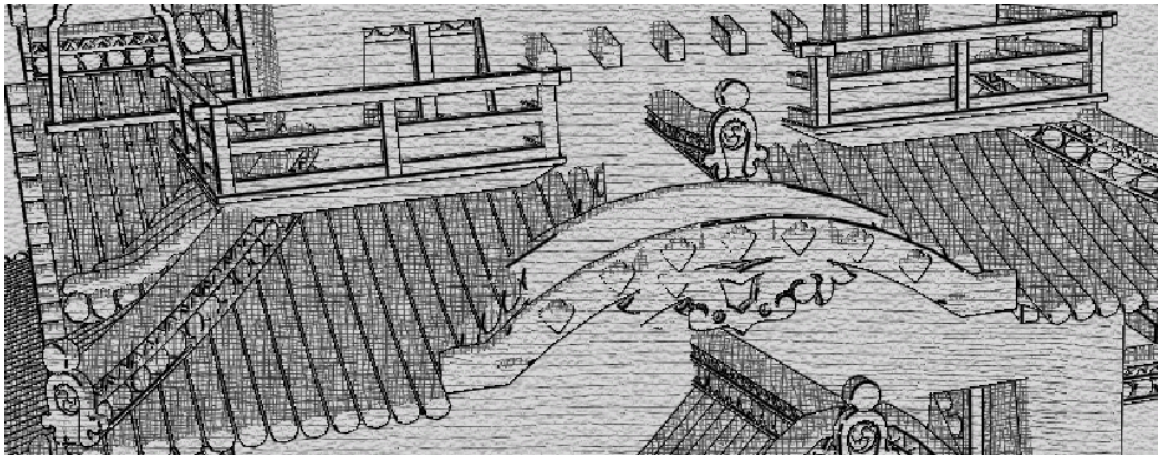


Figure 6: Result from Lake et al. [LMHB00] showing shading with pencil textures of varying density. These textures can be applied projected screen space for a more hand-drawn appearance or can be attached in object space for smooth animation.

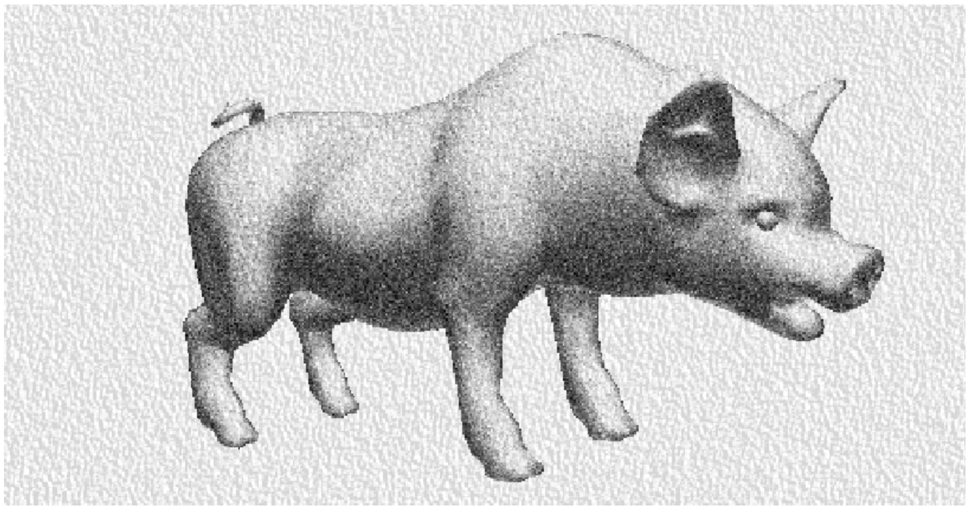


Figure 7: Result from Majumder and Gopi [MG02] shows a pig model rendered in a charcoal style using texture hardware. They use a ”contrast enhancement operator”(CEO) which modifies lighting contrast to account for the limited dynamic range found in charcoal drawings.

represented using grain textures of varying densities and can be displayed at real-time rendering rates using a consumer PC graphics card. Their method also allows the user to interactively vary parameters that influence rendering style, including grain density, paper coarseness and the amount of smudging. An example of an image generated using their technique is shown in Figure 7.

Several authors have described particle-based approaches for interactive storke-based rendering. Kowalski et al. [KMN$^+$99] describe a method for rendering fur, grass, and trees using procedural stroke-based textures. They use graphics hardware to render reference color and ID images. The color image is used to guide the placement of strokes in screen space while the ID image maps pixels to object space to keep track of stroke positions in 3-D space for frame-to-frame coherence. Kaplan et al. [KGC00] use interactive particle systems to render polygonal models in a number of styles. The user is able to control how the particles are rendered and can also change their density and position to mimic a number of hand drawn effects as shown in Figure 8. Like the oriented strokes introduced by Meier [Mei96], the particles exist in 3D object space thus preserving frame-to-frame coherence. Cornish et al. [CRL01] use view-dependent particle systems for stroke-based rendering. By using multi-resolution data structure, they are able to control particle density for efficient interactive rendering.

Mohr and Gleicher [MG01] describe the replacement of the OpenGL runtime graphics library with a NPR version that extracts scene information for non-invasive interactive rendering with alternate styles. Frudenberg et al. [FM01] present a pen-and-ink style game engine that permits interactive walk-throughs with frame-coherent hatch textures.

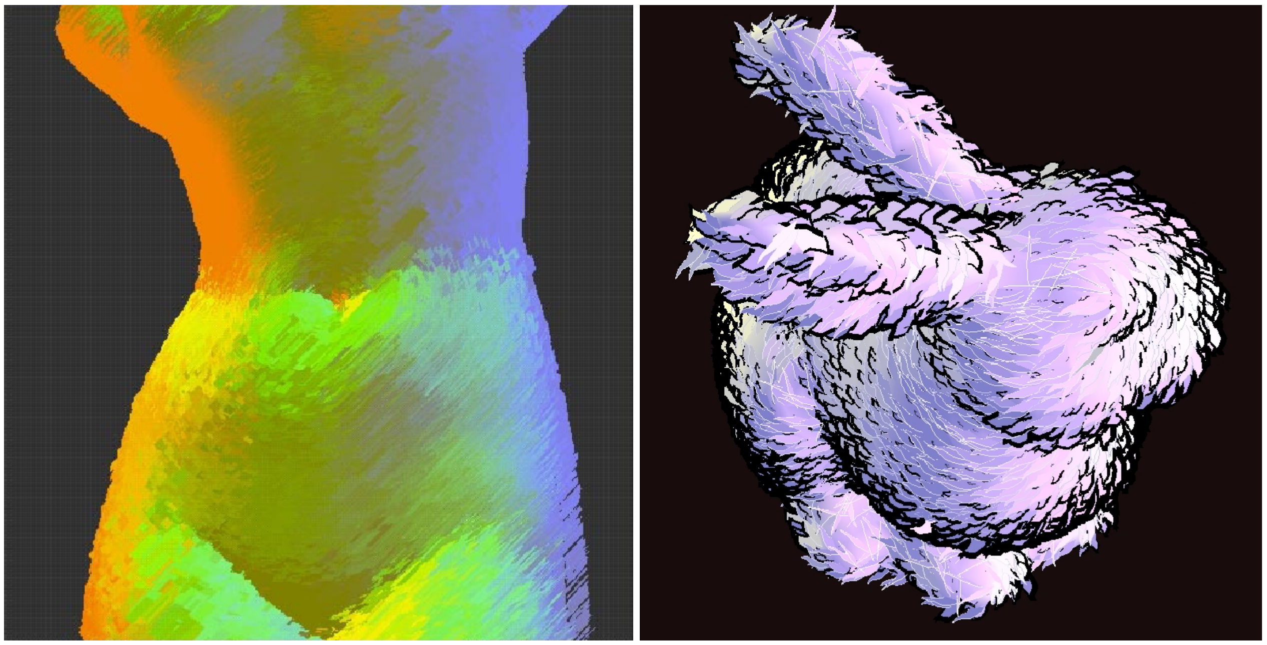


Figure 8: Results from Kaplan et al. [KGC00] shows different artistic styles rendered using their particle approach. The particles exist in 3D object space thus preserving frame-to-frame coherence.

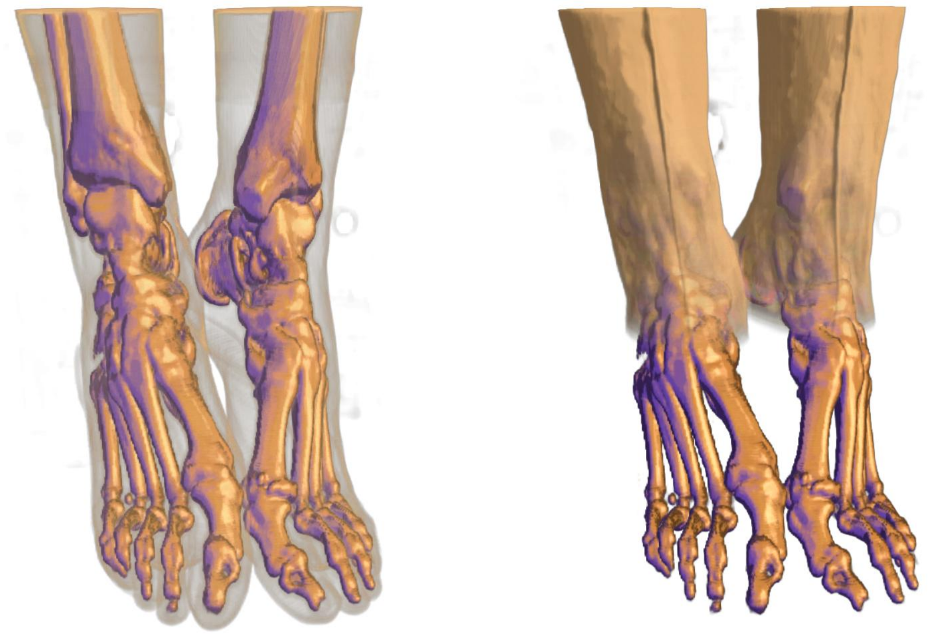


Figure 9: Left: Using gradient based feature enhancement the skin surface is made visible. Right: The skin and flesh are rendered in a more photorealistic style, while the bones are rendered with hue varied shading and silhouettes edges.

3 Non-Photorealistic Volume Rendering

Direct volume rendering is a technique for the display of data that samples a 3D space [Lev90]. It has wide application in science and engineering including the study of data sets from medical MRI and CT scans, weather simulation, structure analysis, air flow surrounding an aircraft, and many other investigations. Volume data sets often consist of regularly spaced scalar samples, called voxels (volume elements), which are mapped to color and opacity values using a lookup table referred to as transfer function. Rendering consists of the display of the accumulation of the semi-transparent material in the volume. By varying the transfer function, the user is able to assign colors to different structures in the volume based on scalar value, and can visualize structures of interest by making them more opaque than surrounding material.

Strictly speaking, traditional volume rendering is not necessarily a form of photorealistic rendering, since it often involves the visualization of values like temperature, pressure, or density that are not directly visible in reality. Never the less, by applying artistically inspired NPR methods to volume rendering, creating more effective visualizations is possible. Treavett and Chen [TC00] present a technique for pen-and-ink style volume rendering. Ebert and Rheingans [ER00] describe a software approach for non-photorealistic volume rendering using techniques that include silhouette edge rendering, cool to warm shading, and depth-cued color variation. Treavett et al. [TCSJ01] describe a software method for non-photorealistic rendering of volumes in painterly and pen-and-ink styles, as well as what they call ”artistic modelling”. This technique involves manipulation of the volume data itself, including the distortion of the volume to give a warped appearance or the addition of solid stroke textures to give the effect of rough carving. In our work [LM02] we show how texture hardware on a consumer PC graphics cards can be used for interactive volume rendering with a number of non-photorealistic techniques. We also describe how large volumes can be interactively rendered non-photorealistically using a cluster of graphics card equipped PCs.

3.1 Volume Rendering With Graphics Hardware

Direct volume rendering can be accomplished by drawing a set of view-aligned polygon slices that sample a 3-D texture containing the volumetric data [VGH96]. By using pixel textures, which allow a texel value to store coordinates into a second texture, scalar and lighting information can be encoded into a single static texel, with transfer function and shading changes occurring through the variation of a single texture [MHS99]. Kniss et al. [KKH01], describe how multi-dimensional transfer functions can be interactively specified and rendered with traditional lighting by using multi-textured/multi-pass rendering. Engel et al. [EKE01] use a technique they call pre-integated volume rendering, which also uses multi-textured/multi-pass techniques for improved rendering accuracy using fewer polygon slices.

Paletted textures store indices into a color palette that samples the RGBA color space. Through the manipulation of the palette over time, the textures can be varied based on transfer function or the viewing parameters without changing the data stored in the textures themselves. It is important to use a representation that avoids any manipulation of data that is stored for every voxel since volumetric data sets tend to be large, and traversing the entire volume in software can severely hamper interactivity. Unlike pixel textures which are filtered prior to their texel lookup, paletted texture are

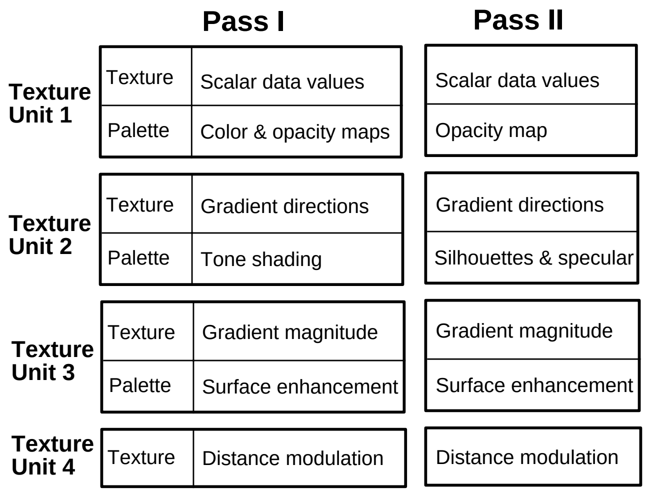


Figure 10: Rendering using two passes each with 4 texture units.

filtered after color lookup in RGBA space, permitting non-linear palette mappings to be used with linear interpolation in hardware.

Multi-texturing allows several textures to be combined on a single polygon during the rendering process [WNDS99]. By utilizing several separate volumetric textures that store scalar data value, gradient magnitude, and gradient direction, and combining them with properly adjusted color palettes, several different non-photorealistic rendering techniques can be rendered in hardware. The Nvidia Geforce3 has four texture units, which means a total of four textures can be blended onto a single polygon. For some effects, additional rendering passes can be added, which consist of blending additional textured polygons on top of the previously rendered polygons in the frame buffer.

An example of how the textures can be allocated in these passes is shown in Figure 10. In this example each of the four texture units is assigned a different texture. The first texture consists of the original scalar values stored in a paletted 3-D texture. The second stores the normalized gradient direction of each voxel. One method for encoding these directions is to use 8-bit paletted textures, with each direction quantized to one of 240 vectors obtained from the faces of a subdivided combination of a dodecahedron and icosahedron [VGH96]. The gradient direction information is useful for lighting and silhouette edge rendering. A third texture contains gradient magnitudes which can be used for enhancing surfaces. The fourth contains a 1-D texture for manipulating color and opacity based on the spatial properties of voxels. The actual number of texture units needed as well as the required number of rendering passes depend on the number of NPR techniques to be applied as will be discussed in the following sections.

3.2 Hue Varied Shading

The cool to warm shading described by Gooch et al. [GGSC98] can be implemented by modulating each voxel color with a lighting texture that manipulates color temperature. As described in the previous section, a paletted normal direction texture can be used, with each texel containing an index into a

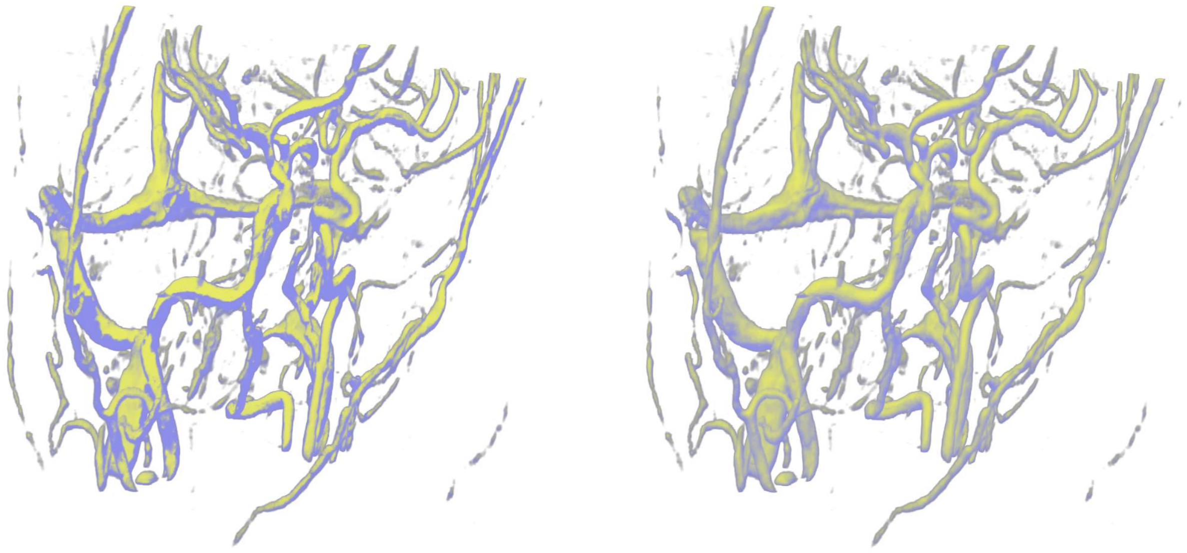


Figure 11: Left: Vessel with abrupt color temperature varied shading. Right: smooth shading.

sampling of a normalized vector space. The palette for this texture is created by first calculating the dot product between each possible normal direction and the light direction. Once the dot product is calculated, the color temperature variation for that product is looked up in a shading colormap specified by the user.

The manipulation of the saturation of the colors used in the shading colormap can control the degree temperature variations in shading are visible. By making the colors more saturated, the effects of this type of shading becomes more subtle permitting more of an objects original color (as specified in the transfer function) to be seen. Furthermore, by changing color value across the shading colormap more traditional lighting with variation in color intensity can be produced.

In addition by making the transition between cool and warm colors relatively short, it is possible to produce the abrupt lighting transitions seen in the image of blood vessels shown in the left image of Figure 11. This can be contrasted with the relatively smooth lighting found in the right image of Figure 11.

3.3 Silhouette Edge Illustration

Silhouette edge rendering can be particularly useful in volume rendering applications since transfer functions are often set such that objects are semi-transparent, sometimes making spatial relationship difficult to determine. Silhouette rendering can be accomplished by using the paletted gradient direction texture with the color palette adjusted such that voxels with gradient perpendicular to the view direction are modulated to black. Alternately this could also be done using a gradient direction texture with per pixel environment mapping similar to the technique described by Gooch et al. [GSG$^+$99].

The image on the top of Figure 12 shows fine blood vessels that are enhanced by rendering silhouette edges. Notice that the spatial relationships between the overlapping vessels near the top of the left

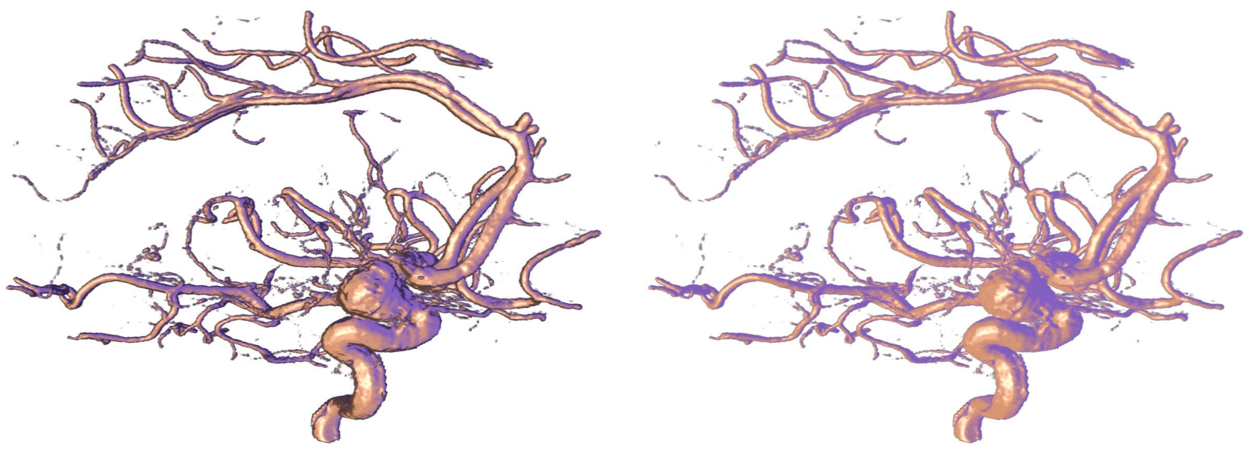


Figure 12: The silhouettes shown on the left image help clarify the spatial relationship between vessels.

image are clearer than those found in the right image without silhouette edge rendering in Figure 12.

3.4 Surface Enhancement

Much of the appeal of direct volume rendering, versus other volume visualization techniques like isosurface rendering, is its usefulness in visualizing continuous regions of semi-transparent material. However, enhancing surfaces can still be effective in clarifying some structures in a volume. Gradients have been used for surfaces enhancement in direct volume rendering applications [Lev90]. Since the transition between features in a volume tend to have the highest gradient magnitude, increasing the opacity in these regions can help to clarify surfaces. The skin surface shown on the left side of Figure 9 is made visible through gradient enhancement, with the underlying material still visible.

Surface enhancement can be implemented by assigning one texture unit a gradient magnitude texture, and allowing the user to specifying a gradient opacity map that modulates the rendered voxel based on gradient. Silhouette edges and specular highlights are typically associated with surfaces, and can be inappropriate for constant valued, semi-transparent regions in a volume. By rendering silhouette edges in a second rendering pass, a separate gradient enhancing functions for the specular and silhouette rendering, resulting in silhouette edges and specular highlights that are only rendered in regions of high gradient.

3.5 Color Based on Position

Color can be manipulated based on distance to improve depth perception [FvFH96]. Aerial perspective has been used by painters to convey depth through the variation of color hue and value based on depth. Typically warmer hues are used for the foreground and become cooler in the background. In addition, color values tend to become lighter and less intense with distance [Kun99].

This can be implemented in hardware using a 1-D texture that modulates the color of the rendered volume along some direction. The depth cues provided by the variation in color are evident in the left

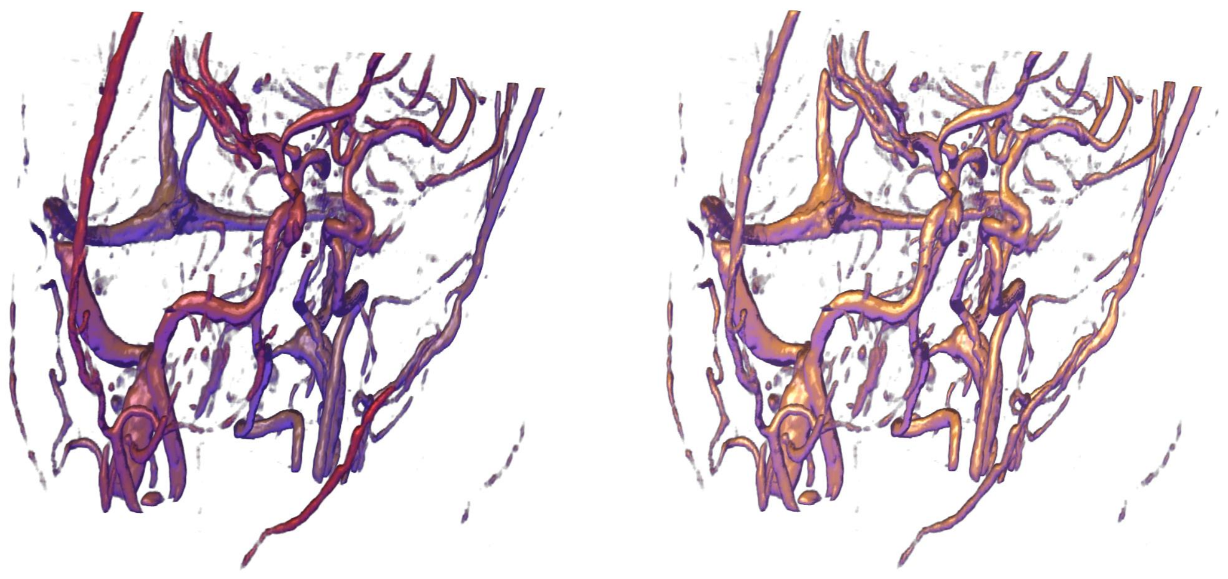


Figure 13: The depth cues provided by the variation in color temperature clarify spatial relationships between vessels. Left: Depth based color manipulation Right: Without depth based color manipulation .

image of Figure 13 where the warmer colored foreground vessels appear closer. This can be contrasted with the right image of Figure 13 where the spatial relationship between vessels is less clear.

Through non-linear fading of the alpha channel along the view direction, closer material can be made more transparent making underlying features more visible, with foreground material still slightly visible to provide context for the features of interest as shown in the right image in Figure 1. Figure 9 on the other hand shows a fading of opacity based on vertical position.

3.6 Multiple Rendering Parameters

Non-photorealistic rendering can be used to add emphasis to specific aspects of a data set. One way this can be accomplished is to use multiple rendering styles for different objects in a volume. For example non-photorealistic and more traditional rendering methods can be combined, making the different materials more distinguishable, or rendering styles can be varied to emphasize or deemphasize the different types of features in a volume. One result of using multiple sets of rendering parameters is that the parameter space is multiplied in complexity, making interactivity all the more important.

Hauser et al. describe how multiple rendering techniques can be combined when visualizing a single volume to better illustrate different types of objects in a volumes [HMBG00]. A similar capability can be accomplished by permitting the user to specify multiple transfer functions, each with its own set of non-photorealistic rendering parameters. Each transfer function can be set to render a different type of object in the volume, with rendering requiring an extra set of rendering passes for each additional transfer function.

The right image in Figure 1 shows the use of this technique, where the internals of the Microsoft

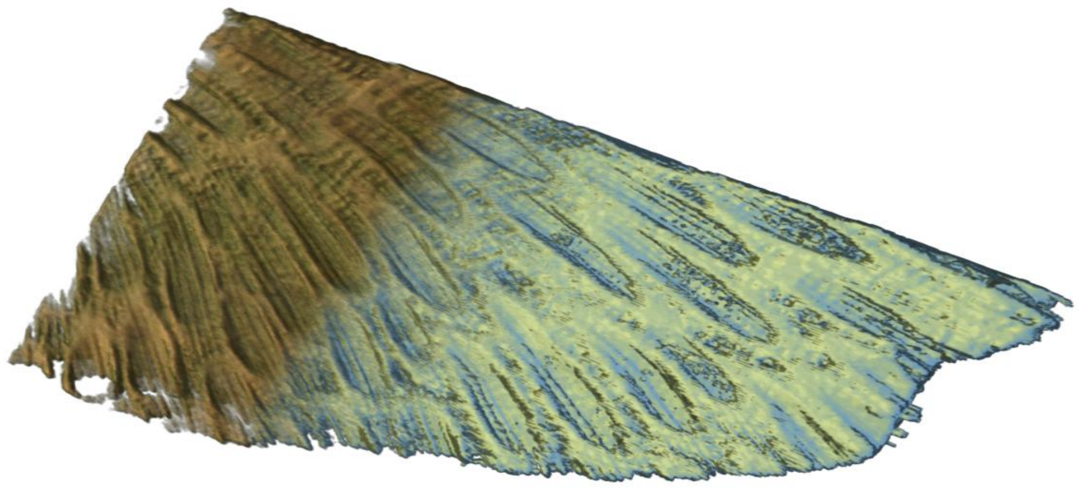


Figure 14: Non-photorealistic and photorealistic rendering styles are mixed in this image of coral.

mouse have been rendered in a non-photorealistic style, while the external plastic parts are rendered more photorealistically. The right image in Figure 9 show bones rendered non-photorealistically using cool to warm shading and silhouettes. The skin on the other hand is rendered in a more photorealistic style, with a fade that allows the several of the bones to be unobstructed by the flesh and skin. In Figure 14 a piece of coral is shown using non-photorealistic and photorealistic rendering parameters. The smooth transition between parameters sets is accomplished using position-based opacity modulation.

3.7 A Complete Example

To summarize the process, we use a CT scan of a Microsoft mouse to illustrate the effect of each non-photorealistic rendering technique. As shown in Figure 15(a), when the data set is rendered using only the transfer function without lighting or NPR enhancements it is very difficult to acquire intuition about the spatial structure of each object, particularly with respect to depth.

With addition of cool to warm shading, shown in Figure 15(b), it becomes easier to determine the surface orientations as seen in Figure 15(c). The addition of warmth and coolness to the volumes does not yield distinctly warm or cool colors, but rather results in a variation in relative temperature to indicate shape information. Notice however, that there is little variation in color intensity across each volume making it still difficult to gain depth clues as to the spatial interactions of the rendered structures. With the addition of the silhouettes edges seen in Figure 15(d) the individual structures become much clearer as seen in Figure 15(e). For example, the separation between the two capacitors near the center of the mouse is much more distinct that when only cool to warm shading is used. Next, Figure 15(f) displays depth color cues for the volume, and Figure 15(g) shows the result after the volume is modulated by these cues. This has the subtle effect of adding warmth to the nearer portion of the mouse. Finally in Figure 15(h) we see the result when a second set of rendering parameters is used to render the outer shell of the mouse with a more photorealistic rendering style. Using depth based variation in opacity the closer portions of the mouse are more transparent allowing the inside to be seen.

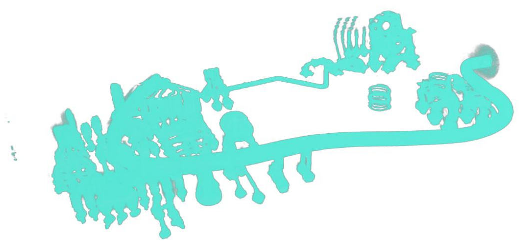
3.8 Parallel Rendering and Performance

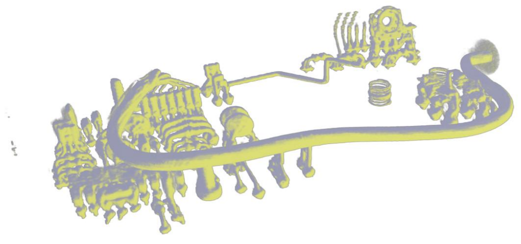
One significant limitation of volume rendering using consumer PC graphics hardware is the limited amount of video memory. For example, the Nvidia Geforce3 has between 64 and 128 megabytes of video memory that is shared between the frame buffer and texture memory. It is very desirable to fit the volume being rendered entirely in texture memory to avoid having to swap data into the graphics card from main memory over the relatively slow graphics bus. By subdividing the volume spatially and distributing it across a cluster of PCs equipped with graphics cards it is possible to fit significantly larger volumes into the aggregated video memory of the entire cluster. In addition to the larger amounts of texture memory provided by a PC cluster, performance improvements also result from the combined fill-rate of multiple graphics cards.

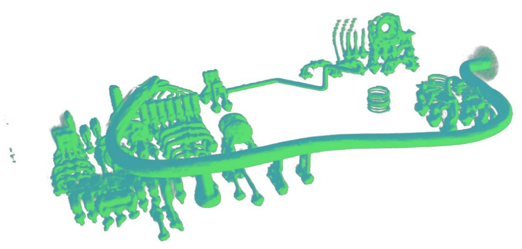
In our implementation we describe in [LM02] we subdivide and distribute the volume to different nodes of a cluster using k-d tree subdivision. During rendering, for each frame, every node on the cluster renders its subvolume and composites the resulting subimage using binary-swap [MPKH94] with the final image being sent to the host for display. We used a PC cluster with nine computers, each with an AMD Athlon 1.3 Ghz processor, one gigabyte of PC133 SDRAM and a Geforce3 with 64 megs of video memory. Eight of the computers use 100-Base-T fast Ethernet. The ninth host computer, used for final display and user interface control, has a gigabit Ethernet connection to the cluster's switch. With this low cost cluster were able to render to a 512×512 window, a $512\times512\times512$ volume at about 2.0 frame per second using two rendering passes. If four rendering passes are used to render the volume using two sets of rendering parameters, the frame rates drops to about 1.2 frame per second. Our experimentation showed this framerate is sufficiently high to make possible interactive exploration of rendering parameter space, in particular the variation of transfer function, viewing direction and non-photorealistic rendering parameters. This permits the tuning of parameters for the creation of meaningful images that illustrates specific structures in a volumetric data set.

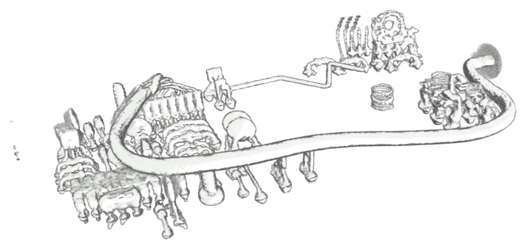
There are some limitations to this approach for interactive non-photorealistic volume rendering. First, rendering rates are still not fast enough for the real-time generation of smooth animation. Higher performance can be achieved by rendering fewer axis aligned polygons, or by using fewer NPR techniques requiring fewer texture units and rendering passes. But the time required for reading the frame buffer from each node and the communication time for compositing the resulting sub-images soon becomes a bottleneck.

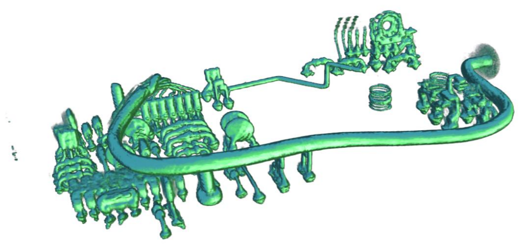
The use of graphics hardware also limits the size of the volume each node can render by the combined texture memory of the cluster. For example, rendering a $1024\times1024\times1024$ volume with eight PCs would require each node to render a $512\times512\times512$ volume which would not fit in texture memory and would make necessary the relatively slow transfer of large amounts of data across the graphics bus for every frame. Despite these limitations, the use of graphics hardware allows for rendering performance that exceeds any software approaches.

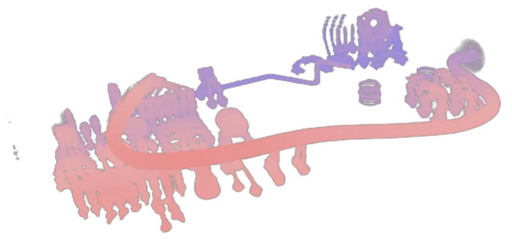

(a) Volume without lighting

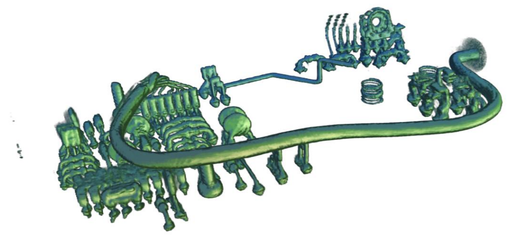

(b) Cool to warm shading contribution


(c) Volume with cool to warm shading


(d) Silhouette contribution


(e) Volume with silhouette


(f) Depth color cue contribution


(g) Volume with depth color cue

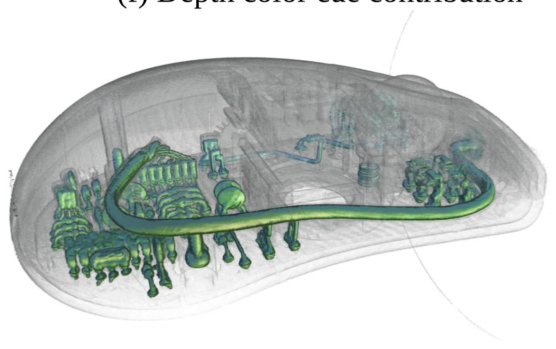

(h) Final volume visualization

Figure 15: A complete example of combined techniques using the mouse data set.

4 Conclusion

In recent years, non-photorealistic rendering has received increasing amounts of attention for both art and scientific applications. Non-photorealistic rendering combined with interactivity, can enable spatial exploration by the user so they can gain further insights into the an objects structure by viewing it multiple angles. Exploration of rendering parameter space, on the other hand, allows for the control of how non-photorealistic rendering techniques are applied and can be used to produce illustrations more appropriate for the users task. Consequently, we foresee increasing use of non-photorealistic rendering in scientific data visualization. With the continued advancement of graphics hardware capabilities, there will be ample room for the development of new interactive non-photorealistic rendering techniques that utilize even more rendering styles.

References

[BS00] J. W. Buchanan and M. C. Sousa. The edge buffer: A data structure for easy silhouette rendering. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000)*, June 2000.

[CRL01] D. Cornish, A. Rowan, and D. Luebke. View-dependent particles for interactive non-photorealistic rendering. In *Graphics Interface 2001*, pages 151–158, June 2001.

[EKE01] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Eurographics / SIGGRAPH Workshop on Graphics Hardware ’01*, pages 9–16, 2001.

[Elb99] G Elber. Interactive line art rendering of freeform surfaces. In *Computer Graphics Forum*, pages 1–12, September 1999.

[ER00] D. Ebert and P Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization 2000 Conference*, pages 195–202, October 2000.

[FM01] B. Freudenberg and T. Masuch, M.and Strothotte. Walk-through illustrations: Frame-coherent pen-and-ink style in a game engine. In *Eurographics 2001*, 2001.

[FvFH96] D. J. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison Wesley, 1996.

[GGSC98] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH ’98 Conference Proceedings*, pages 447–452, July 1998.

[GSG$^+$99] B. Gooch, P. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. In *1999 Symposium on interactive 3D graphics*, April 26-28 1999.

[HMBG00] H. Hauser, L. Mroz, G.-I. Bischi, and M.E. Groller. Two-level volume rendering- fusing mip and dvr. In *IEEE Visualization 2000 Conference Proceedings*, pages 211–218, 2000.

[HZ00] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *SIGGRAPH 2000 Conference Proceedings*, pages 517–526, August 2000.

[IFP95] Victoria Interrante, H. Fuchs, and S. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of IEEE Visualization '95 Conference*, pages 52–59, October 1995.

[KGC00] M. Kaplan, B. Gooch, and E. Cohen. Interactive artistic rendering. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000)*, June 2000.

[KKH01] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization 2001 Conference*, October 2001.

[KLK$^+$00] A. W. Klein, W. W. Li, M. M. Kazhdan, W. T. Correa, A. Finkelstein, and T. A. Funkhouser. Non-photorealistic virtual environments. In *SIGGRAPH '00 Conference Proceedings*, 2000.

[KMN$^+$99] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes. Art-based rending of fur, grass and trees. In *SIGGRAPH '99 Conference Proceedings*, pages 433–438, August 1999.

[Kun99] J. Kunz. *Watercolor Basics: Color*. North Light Books, 1999.

[Lev90] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[LM01] E. B. Lum and K.-L. Ma. Non-photorealistic rendering using watercolor inspired textures and illumination. In *Pacific Graphics 2001 Conference Proceedings*, pages 322–330, October 16-18 2001.

[LM02] E. B. Lum and K.-L. Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002)*, June 3-5 2002.

[LMC02] E. Lum, K.-L. Ma, and J. Clyne. A hardware-assisted scalable solution for interactive volume rendering of time-varying data. *(to appear)IEEE Transactions on Visualization and Computer Graphics*, 2002.

[LMHB00] A. Lake, C. Marshall, M. Harris, and M. Blackstein. Stylized rending techniques for scalable real-time 3d animation. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000)*, June 2000.

[Mac99] G. Mackenzie. *The Watercolorist's Essential Notebook.* North Light Books, 1999.

[Mei96] B. J. Meier. Painterly rendering for animation. In *SIGGRAPH '96 Conference Proceedings*, pages 477–484, August 1996.

[MG01] A. Mohr and M. Gleicher. Non-invasive, interactive, stylized rendering. In *2001 ACM Symposium on Interactive 3D Graphics*, 2001.

[MG02] A. Majumder and M. Gopi. Hardware accelerated real time charcoal rendering. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002)*, June 2002.

[MHS99] M. Meissner, U. Hoffmann, and W. Strasser. Enabling classication and shading for 3d texture mapping based volume rending using opengl and extensions. In *IEEE Visualization '99 Conference Proceedings*, 1999.

[MKT$^+$97] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes. Real-time nonphotorealistic rendering. In *SIGGRAPH '97 Conference Proceedings*, pages 415–420, August 1997.

[MPKH94] K.-L. Ma, J. Painter, M. Krogh, and C. Hansen. Parallel volume rendering using binary-swap compositing. *IEEE Computer Graphics & Applications*, 14(4):59–68, July 1994.

[NM00] J.D. Northrup and L. Markosian. Artistic silhouettes: A hybrid approach. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000)*, June 2000.

[PHWF01] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *SIGGRAPH '01 Conference Proceedings*, August 2001.

[Ras01] R. Raskar. Hardware support for non-photorealistic rendering. In *Siggraph Graphics Hardware '01*, August 2001.

[RC99] R. Raskar and M. Cohen. Image precision silhouette edges. In *1999 Symposium on interactive 3D graphics*, April 1999.

[SABS94] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94 Conference Proceedings*, pages 101–108, July 1994.

[SMGG01] Peter-Pike Sloan, William Martin, Amy Gooch, and Bruce Gooch. The lit sphere: A model for capturing npr shading from art. In *Graphics Interface 2001*, pages 143–150, June 2001.

[TC00] S. M. F. Treavett and M Chen. Pen-and-ink rendering in volume visualisation. In *Proceedings of IEEE Visualization 2000 Conference*, pages 203–209, October 2000.

[TCSJ01] S. M. F. Treavett, M. Chen, R. Satherley, and M. W. Jones. Volumes of expression: Artistic modelling and rendering of volume datasets. In *Proceedings of Computer Graphics International 2001*, pages 99–106, July 2001.

[VGH96] A. Van Gelder and U. Hoffman. Direct volume rendering with shading via three-dimension textures. In *ACM Symposium on Volume Visualizatrion '96 Conference Proceedings*, 1996.

[WNDS99] M. Woo, J. Neider, T. Davis, and D. Shreiner. *The OpenGL Programming Guide, Third Edition*. Wesley, 1999.

[WPFH02] M. Webb, E. Praun, A. Finkelstein, and H. Hoppe. Fine tone control in hardware hatching. In *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002)*, June 2002.