To appear in the Late Breaking Hot Topics session at the *Visualization'99* conference

IBR-Assisted Volume Rendering

Klaus Mueller, Naeem Shareef, Jian Huang, and Roger Crawfis

Department of Computer and Information Science, The Ohio State University, Columbus, OH 43212

Abstract

Volume rendering at interactive frame rates remains a challenge, especially with today's increasingly large datasets. We propose a framework, using concepts from Image-Based Rendering (IBR), that decreases the required framerate for the volume renderer significantly. All the volume renderer needs to supply is a set of renderings at 'key' view points, and the IBR renderer will interpolate the intermittent frames at good accuracy. The IBR provides methods to handle both opaque and transparent datasets, and is an inexpensive process that can be run on the user's desktop machine.

1 Introduction

Direct volume renderers can be grouped into raycasting approaches [6], cell-projection methods [17], Fourier space methods [9], Shear-warp [4], Splatting [16], and implementations using 3D texture-mapping hardware [1]. Indirect volume renderers extract the isosurfaces first and use these to build a polygonal mesh [8]. They then display this mesh employing z-buffer algorithms. Parallel renderer have also been implemented (see for example [12]), but may require relatively expensive hardware. Any of these systems are able to produce volume renderings at interactive rates - as long as the magnitude of the visualization task is kept within the system's feasible limits: For example, Shear-warp [4] and implementations using 3D texture mapping hardware [1] can produce interactive renderings even for fairly large datasets of 512^3 voxels, while the Utah parallel raycaster [12] can interactively render the CT version of the Visible Man [13] dataset.

But the researcher's and scientist's desire for bigger, better, and more detailed datasets is bound to drive even the most sophisticated system to its limits. Let's assume we have a volume rendering system that achieves near-interactive rendering when presented with datasets of 100 MB, but now we would like to render a dataset in the GB range. For example, we would like to visualize a high-resolution MRI dataset of size $1k^3$ or larger, or the visible woman's RGB dataset of size 40GB, or a terascale dataset. In this case, the system would become overloaded and new images would only be generated every, say, three seconds. Thus, no longer is the viewing system responsive to the user's input, and this makes precise navigation within the dataset difficult. While in immersive displays a delayed system response causes motion sickness, in screen-based systems, non-interactive screen updates disconnect the user from the data and breaks his or her attention to the viewing task. Interactive viewing can be restored by reducing the complexity of the rendering task, which in turn may lower the quality of the delivered images considerably. Progressive refinement can be used to return to a high-quality rendering once the user stops. Adaptive refinement techniques focus on important areas first and render others if there is time. Other systems simply display a bounding box of the dataset while the user is transforming the object. Common to all these approaches is, that since the volume renderer is always busy rendering lower-quality images at interactive rates, the user will never see a high quality image until he or she stops the transformation and the system has time to catch up. This is clearly undesirable. Rather, we would like to have a more graceful mechanism that at least presents a high-quality image every second or so, and otherwise displays a reasonable approximation at the proper orientation. We are currently pursuing research in this direction, using concepts from image-based rendering (IBR) to generate the near-accurate intermediate frames. Our novel framework hides any kind of latencies, be it those due to system overload, other rendering latencies, and network latencies that may occur when the renderer, possibly a parallel system, is located in a central location and delivers the rendering result over the network to a user's workstation.

email: {mueller, shareef, huangj, crawfis} @cis.ohio-state.edu

Our IBR renderer is an inexpensive process that operates on the user's desktop PC and only requires a low-cost 2D texture mapping board. It can use output from most volume renderers and will generate the intermediate frames from these 'key' renderings.

The next section, Section 2, discusses relevant previous work done in the field of IBR. Then, in Section 3, we will describe our system. Section 4 will show some results, and Section 5 will draw some conclusions and give an outlook onto future work.

2 Previous work in IBR

Image-based rendering (IBR) enables interactive visualization of a scene without having to pay the time-consuming costs of directly rendering the scene for every frame. Instead, renderings of the scene at arbitrary viewpoints are computed from nearby pre-rendered images, or even photographs, that are positioned at defined viewpoints. These images are then used as representations of the scene instead of the model data itself. Using IBR, high-quality walk-throughs and fly-throughs of complex environments can be performed in real-time on PC's and low-end workstations.

IBR capitalizes on frame-to-frame coherence: As the viewer moves through the scene, it is assumed that only few changes occur from one viewpoint to the next [2]. Typical IBR systems consist of three steps: acquisition of pre-computed or sample images, resampling of these images, and image reconstruction. A well-known example for IBR is the Apple QuickTime VR system [2], which reconstructs new views from pre-computed panoramic images for fly-throughs of an environment.

In IBR, we are interested in reconstructing the plenoptic function [10] of an object from its samples. The plenoptic function is a 5D function and describes the flow of light at all locations in space in all directions. If we assume that the viewed object is opaque, and if we limit our interest to light leaving the object's convex hull, then we can reduce the plenoptic function to 4D [3][7]. For example, the Lumigraph [3] stores samples of an object's 4D plenoptic function, and images from any arbitrary viewpoint can be computed via interpolation of this sample space. The 4D samples are obtained from a set of precomputed or acquired images taken around the object. Lightfield Rendering [7] uses similar ideas. When the Lumigraph is created, the acquired images are not only used to establish the discretized 4D plenoptic function, but also to reconstruct an approximation of the underlying object on a 3D cubic grid. Then, each time an image is calculated from the Lumigraph, this voxelized object is projected into a z-buffer, and the projection is used to pick and weight the most appropriate samples from the Lumigraph for interpolation of the new image. While the Lumigraph uses static, pre-computed imagery, Shade [14] computes images on the fly and expires them when they no longer satisfy an error criterion. He utilizes projective image representations of scene objects in the form of sprites to quickly display successive views. In a later paper, Shade [15] employs sprites that have depth values, called Layered Depth Images (LDIs). An image for a new view is then reconstructed from multiple LDI's in a back-to-front visibility order, which solves the occlusion problem.

While IBR has been successfully applied in a variety of ways for surface graphics, there have been no efforts so far to exploit IBR concepts for volume graphics. Part of the reason may be that using IBR for scientific volume visualization is more difficult, since here we often have highly transparent data, for which we cannot reduce the plenoptic function to 4D. We have obtained preliminary results for an approach that reconstructs the full 5D plenoptic function, using a reasonable approximation. On the other hand, if the volumetric data are opaque, then our approach yields a good reconstruction of the 4D function.

3 The new approach: IBR-assisted volume rendering

We shall now explore how we can exploit the concepts of IBR in the special scenario of volume visualization. Consider Fig. 1a: The dotted arrows show the path of a set of rays, integrating the volume for a certain viewpoint V_a. The solid arrows show the ray paths for a slightly different viewpoint V_b. A very simple IBR implementation would use the integrals computed for V_a, and re-use them for V_b by mapping the image of V_a onto a billboard and viewing this billboard from V_b (as shown in Fig. 1b). However, we observe in Fig. 4a that the 3D illusion vanishes rapidly, even for small differences of V_a and V_b: The viewer recognizes quickly that he or she is looking at a 2D billboard of a 3D object, and not at a true 3D object. This comes at no surprise (see again Fig. 1a): The linear paths of the integrals of V_b (the ones we should be using) differ significantly from the linear paths of the integrals for V_a (the ones we are using).

A better approximation can be obtained by breaking the volume into several image-aligned slabs, rendering these slabs separately, and compositing the slab images for other viewpoints. A complete ray integral is then constructed by combining the appropriate partial integrals stored in consecutive slab images (see Fig. 2a). Interpolation is used to obtain partial integrals at non-grid positions in the slab images. The integration error is now reduced

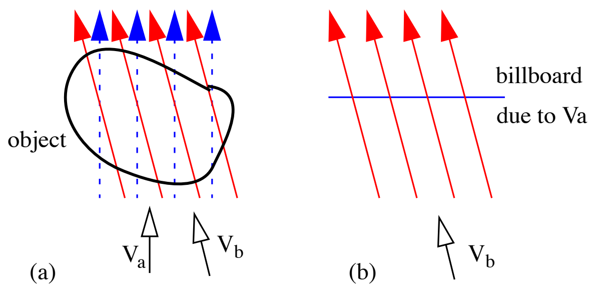

Fig. 1. (a) Paths of ray integrals for two different integration directions of a volumetric object, V_a (dotted) and V_b (solid). (b) A billboard is constructed by mapping the image obtained from rendering the volume from direction V_a onto a polygon. This billboard is then viewed from different directions. The observer quickly realizes that he/she is watching a billboard painted with a 2D image of a 3D object, and not a true 3D object.

to the error due to the angular deviation within a slab. This method of retaining and utilizing partial volume integrals for the reconstruction of images at arbitrary views represents a novel way to reconstruct the 5D plenoptic function from its samples in the IBR framework. Although this reconstruction is only approximate as it compounds samples along a ray into a slab ray, we can control the error by varying the thickness of the slabs.

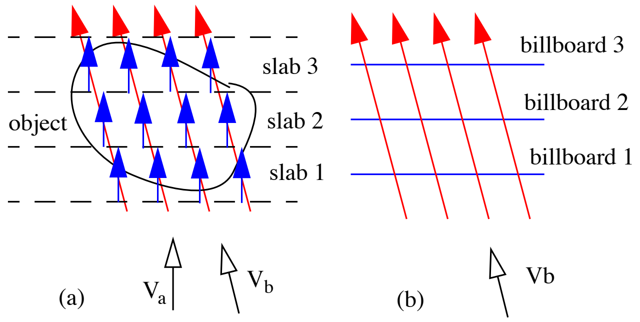

Fig. 2. (a) The volume is decomposed into slabs, and each slab is rendered into an image from view direction V_a. The ray integrals for view direction V_b can now be approximated with higher accuracy by combining the appropriate partial ray integrals from view V_a (stored in the slab image). Interpolation is used to obtain partial integrals at non-grid positions. (b) The three billboard images can be composited for any view, such as V_b shown here.

The IBR compositor places the slab images in the center plane of their respective slabs and uses them to render other viewpoints (see Fig. 2b). This is conveniently achieved by placing a polygon into each slab centerplane and mapping the respective slab image as a texture onto it. These texture-mapped polygons are then rotated, translated, zoomed, and projected, according to the current viewing parameters. This is accomplished using a scene graph and commodity graphics boards. Fig. 4b confirms that this works well for a transparent and X-rayed volumes, in which all rays traverse the volume entirely. However, for opaque objects, we expect the IBR approximation of the volume rendering process to be more quickly revealed. Consider Fig. 4c: Although the process holds up very well for angles up to 6°, the pre-integrated object slices seem to be pulled apart as the object is rotated further. Holes appear when viewing rays hit regions in the slab images that were only covered by invisible voxels in the interior of the object (see Fig. 3).

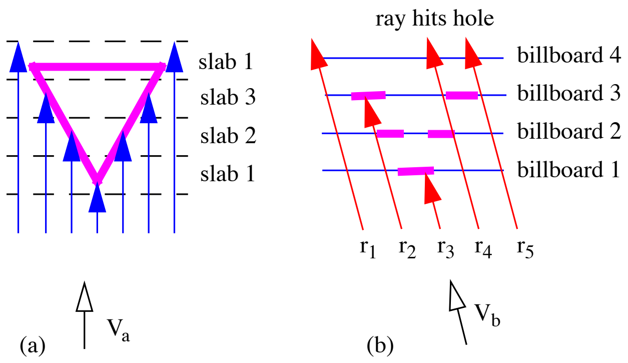

Fig. 3. (a) Rendering of an opaque object into slabs from view direction V_a. (b) The slab images only contain the iso-surface of the object. The ray r_4 penetrates billboard image 2 and hits the black background. A hole appears in the composited image.

To address this issue, we devise a scheme that enhances our (slab image) billboards with depth information. We achieve this by subdividing a slab image into a grid of small tiles, say of size 16×16 pixels. Then, when rendering a slab image, we record, for every tile, the z-value of the frontmost (z-near) and backmost (z-far) splat that was projected onto it. This is implemented using a simple counter-set and results in a coarse-scale z-buffer. From this we can generate a C_0-continuous quadmesh for each slab image and map the tile images onto it. The (x, y)-vertex coordinate of a mesh quadrilateral is given by the (x, y) coordinate of the respective tile vertex, while the z-coordinate is given by the average of the z-values of the four tiles that have that vertex in common. This yields a polygonal object in which the quadmeshes represent a close spatial approximation of the underlying object surface, while the tile textures provide the object detail. Fig. 4d shows the mesh-augmented IBR rendering of the head dataset at a rotation angle of 7°. We observe that the gap has disappeared. Fig. 5 shows the corresponding poly mesh, rotated out for illustrative purposes. The texture mapped quadmeshes can be transformed to any new viewpoint and then composited. Note that if the quadmesh is viewed from the direction at which the slab images were obtained, then the rendered image is correct, at least for orthographic projection, This is because the quadmesh retains the (x, y) coordinates of the tiles, and a projection of a tile that was warped in z only will yield the same image as the projection of a flat one.

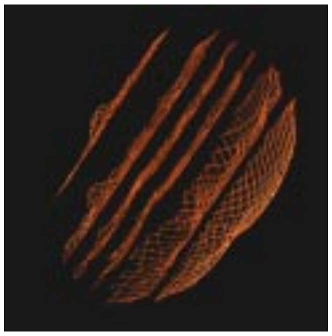
Fig. 5. The polymesh for the head dataset for the view in Fig. 4d, rotated for illustrative purposes.

4 Results

We applied our algorithm to three volume datasets: The *head* dataset is an MRI volume obtained from UNC, Chapel Hill and has 256×256×163 voxels. The *tomato* dataset is an MRI volume obtained from Lawrence Berkeley Lab [5] and has 256×256×64 voxels. The tomato has been segmented into five structures: core, endocarp, locule, placenta, and seeds. The *nerve* dataset is volumetric sample of a ganglion nerve, acquired with a confocal microscope. It has 512×512×76 voxels. We have used our image-aligned sheet-buffered splatting algorithm [11] to render the slab images, but any other renderer could have been used as well. Occlusion culling was used inside the individual slabs, but was not used in-between slabs. All slab images were rendered at a resolution of 512×512 pixels.

Fig. 6 shows a collection of images rendered with the IBR algorithm at progressively increasing angular discrepancies from the orientation at which the slab images were calculated from the slab renderers. We show discrepancies of 8°, 16°, and 24° for each dataset. We observe that artifacts start to appear around 16°. Generally, transparent datasets, such as the tomato, and dispersed datasets, such as the nerve, allow larger angular deviations than opaque, more regular objects, such as the head. Thus the IBR rendering allows the viewing angles to be varied within a cone with angle 16°. Thus, the user can tilt the object back and forth within a range of 32°. The IBR framerate was close to 30/s for all cases.

Quicktime movie files, acquired at real-time from the system, are available at http://www.cis.ohio-state.edu/~mueller/IBR.

5 Conclusions and future work

We have presented a back-end IBR module that can be used with almost any volume renderer, including polygonal. Without IBR assistance the volume renderer would have to produce a new image every 1/30s for interactive viewing, which is quite ambitious, even for smaller datasets. The IBR takes this load off the renderer, and allows interactive viewing with little artifacts until the volume renderer has produced a set of high-quality slab images for the next key viewpoint.

The volume renderer that feeds the IBR can be located at a remote site, sending its images via ATM, or it can reside as a separate process directly on the user's local workstation. The only condition on the volume renderer is that it produces a set of partial images due to image-aligned slabs. Future research may relax this condition. The IBR is an inexpensive process and uses a low-cost commodity texture mapping hardware to perform the compositing

We are currently investigating schemes that deal with the update of the IBR slab image sets as the user roams out of the

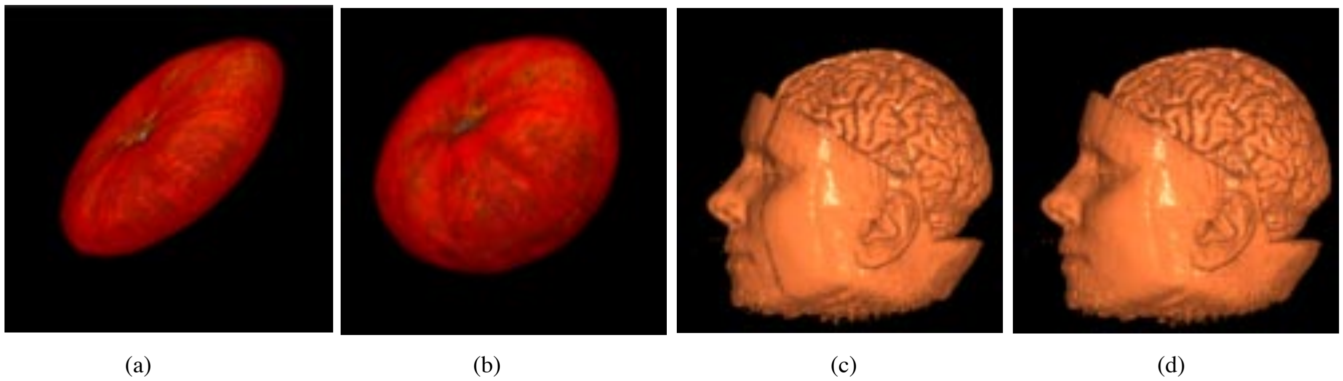
(a) (b) (c) (d)

Fig. 4. A single billboard, mapped with an image of a semitransparently volume rendered tomato, and rotated by 20°. (b) The LBL tomato data set was partitioned into 6 slabs, and each slab was rendered separately. The resulting images were mapped onto billboards, rotated by 20°, and composited. While the rotated object in (a) reveals that it is merely a 2D polygon with a tomato image painted onto it, the composited object in (b) retains its 3D appearance. (c) The head dataset, composited at 7° with 6 planar slab images. We observe the appearing gap between the two front-most images. (d) The same dataset and rotation angle, but now the images were mapped onto a polymesh. The gaps have disappeared.

applicable range of one slab image set into another. Here we would like to know: Do we simply switch to the more appropriate set, or do we cross-dissolve between two or more slab image sets that were computed at viewpoints close by. The issue of cross-dissolving brings up another issue: Do we cross-dissolve between two or more completed IBRs or do we cross-dissolve two or more slab images and compute the IBR from those. Also, we need to develop methods that predict where the user is headed to. Ideally, the renderer should supply a slab image set before the user actually reaches the position at which the set was rendered. That way, the full rotational range of the set can be utilized for the IBR.

Finally, what makes our system different from the Lumigraph is that we perform a lazy sampling of the plenoptic function. This lazy evaluation enables us to change the volume's transfer function or the isorange on the fly, and the IBR can also interpolate between slab image sets obtained with different transfer functions.

References

[1] B. Cabral, N. Cam, J. Foran, "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," *1994 Symp. on Vol. Vis.*, pp. 91-98, 1994.

[2] S.E. Chen, "QuickTime VR - An Image-Based Approach to Virtual Environment Navigation," *Proc. SIGGRAPH '95*, pp. 29-38, 1995.

[3] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The Lumigraph," *Proc. SIGGRAPH '96*, pp. 43-54, 1996.

[4] P. Lacroute, M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," *Proc. SIGGRAPH '94*, pp. 451-458, 1994.

[5] Lawrence Berkely Lab, "Whole Frog Project", from http://www-itg.lbl.gov/, 1994.

[6] M. Levoy, "Efficient ray tracing of volume data," *ACM Trans. Graph.*, vol. 9, no. 3, pp. 245 - 261, 1990.

[7] M. Levoy, P. Hanrahan. "Light Field Rendering," *Proc. SIGGRAPH 96*, pp. 31-42, 1996.

[8] W. Lorensen, H. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. SIGGRAPH '87*, pp. 163-169, 1987.

[9] T. Malzbender, "Fourier volume rendering," *ACM Trans. Graph.* vol. 12, no. 3, pp. 233-250, 1993.

[10] L. McMillan and G. Bishop, "Plenoptic Modeling: An Image- Based Rendering System," *Proc. SIGGRAPH '95*, pp. 39-46, 1995.

[11] K. Mueller, N. Shareef. J. Huang, R. Crawfis, "High-quality splatting on rectilinear grids with efficient culling of occluded voxels," *IEEE Trans. Vis. and Comp. Graph.*, (to appear), June, 1999.

[12] S. Parker, P. Shirley, Y. Livnat, C. Hansen, P. Sloan, "Interactive ray tracing for isosurface rendering," *Proc. Vis. '98*, pp. 233-238, 1998.

[13] http://www.nlm.nih.gov/research/visible/visible_human.html

[14] J. Shade, D. Lischinski, D. Salesin, T. DeRose, J. Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments," *Proc. SIGGRAPH '96,* pp. 75-82, 1996.

[15] J. Shade, S. Gortler, Li-Wei He, R. Szeliski, "Layered depth images," *Proc. SIGGRAPH '98*, pp. 231-242, 1998.

[16] L. Westover, "Footprint evaluation for volume rendering," Proc. SIGGRAPH'90, pp. 367-376, 1990.

[17] J. Wilhelms, A. Van Gelder, "A coherent projection approach for direct volume rendering," *Proc. SIGGRAPH '91,* pp. 275-284, 1991.

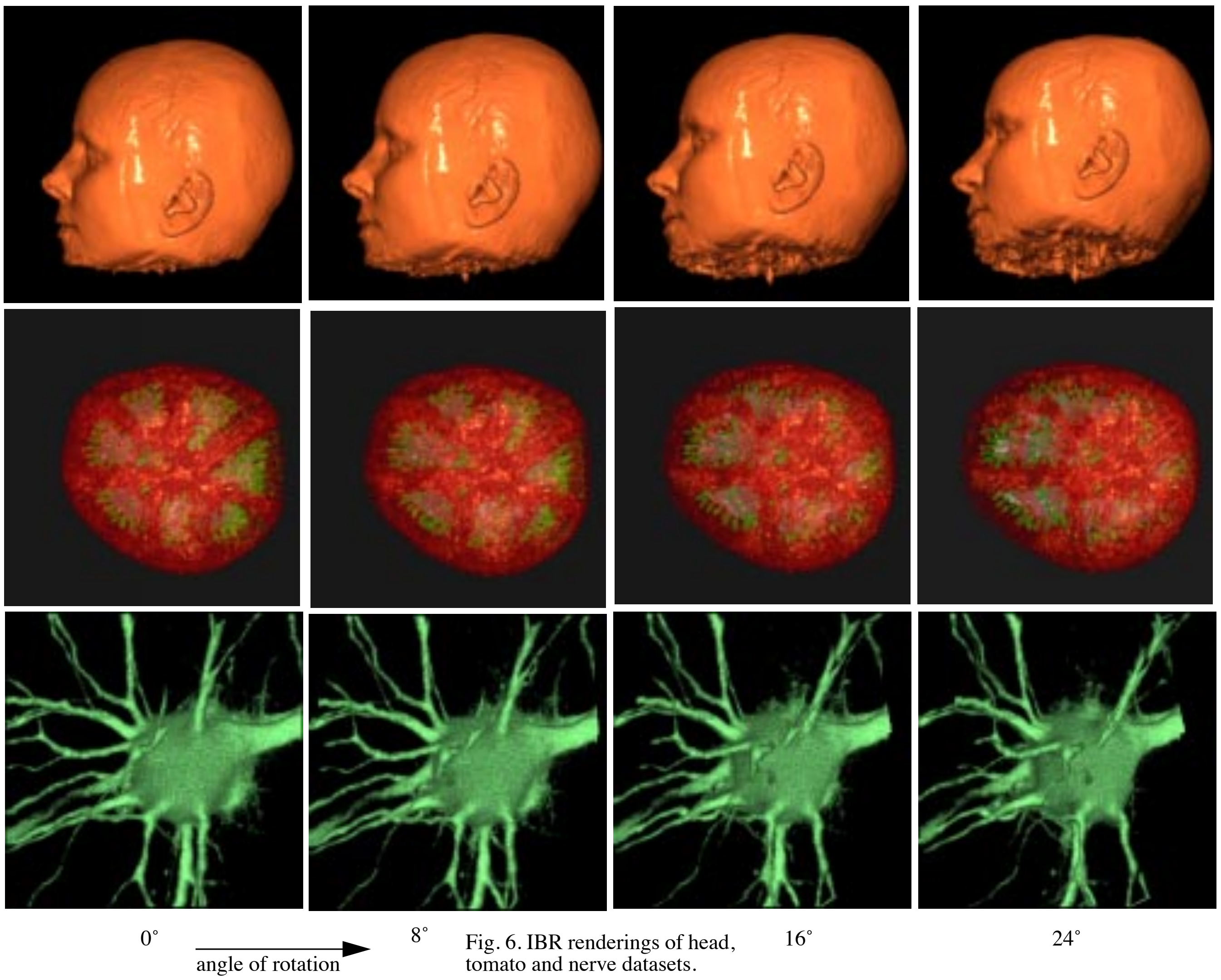

Fig. 6. IBR renderings of head, tomato and nerve datasets.