A view-dependent approach to MIP for very large data

Naeem Shareef and Roger Crawfis
Dept. of CIS, The Ohio State University, Columbus, OH

ABSTRACT

A simple and yet useful approach to visualize a variety of structures from sampled data is the Maximum Intensity Projection (MIP). Higher valued structures of interest project over occluding structures. This can make MIP images difficult to interpret due to the loss of depth information. Animating about the data is one key way to try to decipher such ambiguities. The challenge is that MIP is inherently expensive and thus high frame rates are difficult to achieve. Variations to the original MIP algorithm and classification can help to further alleviate ambiguities and provide improved image quality. Unfortunately, these improved techniques are even more expensive. In addition, they require substantial parameter searching and tweaking. As today's data sizes are increasingly getting larger, current methods only allow very limited interaction. We explore a view-dependent approach using concepts from image-based rendering. A novel multi-layered image representation storing scalar information is computed at a view sample and then warped to the user's view. We present algorithms using OpenGL to quickly compute MIP and its variations using commodity off-the-shelf graphics hardware to achieve near interactive rates.

Keywords: MIP, view-dependent, image-based, transfer function, graphics hardware.

1. INTRODUCTION

Due to its simplicity, MIP is a visualization technique that has been effectively used in practice to visualize such structures as vasculature, bone, and soft tissue from MR and CT data. It is a view-dependent operation that filters the data to pass higher scalar values. For example, structures highlighted with radioactive tracers are projected over low-level occluding structures and noise. Visibility ordering and depth information is lost which makes the static MIP image difficult to interpret. A good illustration of this is to try to determine the visibility ordering of a large network of highly intertwined vasculature from a single MIP image (see Figure 3, top left). Interactive rendering helps to mentally reconstruct the lost information by providing 3D cues via motion parallax.

MIP is an inherently expensive operation because all scalar values along a sampling ray need to be evaluated. Whereas various approaches[4, 10, 13, 2, 3, 5, 6, 7] have tried to reduce bottlenecks in rendering or utilize acceleration hardware to reduce latency, they do not scale well with increasing dataset size. Due to new advances in imaging technology and various research initiatives, volume datasets will have unprecedented quality with higher spatial and scalar resolutions. The Visible Human project[1] has made available MR and CT volume datasets sampled at.33 mm resolution of the entire human body that requires storage on the order of gigabytes. The Ohio State University houses an MRI facility[14] with the strongest magnetic field in the world at 8 tesla. This device is capable of producing extremely high resolution MRI data. Data management, network transmission, and visualization are all active areas of research for data of this magnitude. Thus any proposed "end-to-end" solution relies upon expensive approaches. Remote visualization, which is key towards the effective use of this data, say from a doctor's office or remote satellite facility, is very difficult.

Variations[4, 11] and classification[6,13] to the MIP approach allow for less ambiguity and improved image quality. One approach[4] applies a depth weighting function to the scalar values before the MIP operation, to simulate depth shading. Another approach[11] projects the closest local maximum over a user-defined threshold. A transfer function can pick out desired structures or contrast-enhance adjacent structures. These approaches require considerable parameter searching and tweaking to arrive at a desirable rendering and add computational complexity.

We present a view-dependent approach that allows for MIP rendering at interactive frame rates on commodity platforms, such as a PC equipped with off-the-shelf graphics acceleration hardware with texture mapping. It is an image-based approach that pulls from recent research in image-based rendering (IBR). An image-based representation is constructed at a pre-chosen viewpoint by subdividing the view frustum along the z-direction into a number of image-aligned slab regions. This partitioning subdivides all sampling rays into sub-intervals and corresponding min/max values are computed and stored for each interval. We construct a novel image-based representation that is used to compute reprojections at interactive rates using graphics hardware with texture mapping. Images have many advantages as a rendering

primitive including flexibility and efficiency in storage, retrieval, transmission, and compression/decompression. We implement an image-based browser and rendering framework where the large and cumbersome volume data is replaced with a more manageable purely image-based representation.

In the next section we describe the MIP operation, its two variations, and classification. Section 3 describes our novel image-based representation which is computed and stored for each view sample. In Section 4 we present MIP algorithms using OpenGL to render image layers fast. We describe an image-based framework for interactive browsing in Section 5. Finally, we show renderings using our approach on three volume datasets.

2. MIP, DMIP, LMIP, AND CLASSIFICATION

The original MIP approach simply projects the maximum scalar value along a sampling ray from the eye through a pixel. Equation 1 defines the operation applied to a scalar function *f(z)*, defined on a sampling ray through pixel *p* and parameterized by *z*. Typically the resulting scalar value is converted to a grey level intensity. Here, we limit the ray depth to a distance *D* extending from the image plane to a distance past where the ray exits the volume. MIP involves a search over all values on a ray and is thus an expensive operation. Also, the selection operation ignores visibility ordering as well as depth information, so that higher valued structures farther away may occlude portions of closer structures. Interpretation of such a MIP computed image can be difficult without such cues. A variation to MIP[4] applies depth shading to the original MIP approach with a weight function parameterized by distance from the eye. We call this approach DMIP and illustrate its operation in equation 2. Heidrich et. al.[4] use standard graphics hardware and a linear function for *d(z)*. A later variation to MIP, called LMIP, finds the closest local maximum greater than a user-defined threshold. Equation 3 defines the LMIP operation with a threshold *t*, where *LMAX* is true if the function value is a local maximum. If no scalar on the ray is greater than or equal to *t*, then the maximum intensity is projected, as in the original MIP approach. Implementations using a front-to-back ray traversal can exploit early ray termination for this particular technique.

$$I(p) = \max_{0 \le z \le D} (f(z)) \tag{1}$$

$$I(p) = \max_{0 \le z \le D} (d(z) \cdot f(z)), \quad 0 \le d(z) \le 1 \tag{2}$$

$$I(p, t) = f(z_k), z_k = \min_{0 \le z \le D} (z | ((f(z) \ge t) \; AND \; LMAX(z))) \tag{3}$$

The application of classification[9] using a transfer function *g()* can help to pick, remove, or contrast-enhance various structures. A transfer function is defined as any function which returns some material property. We consider transfer functions that modulate the scalar field. As shown in equation 4, the maximum operation is applied to the transformed scalar value. Examples of some commonly used transfer functions are ramp, step, and window functions, though the function may be defined arbitrarily. One example use of transfer functions in CT data in medical visualization is to window value ranges that represent particular tissue types.

$$I(p) = \max_{0 \le z \le D} (g(f(z))) \tag{4}$$

Research has focused mainly on acceleration techniques to reduce bottlenecks in the rendering pipeline. Usually a cost is incurred in terms of image quality with simplifications in one or more steps of the pipeline, e.g. resampling and reconstruction. Also, most approaches pre-classify the volume. Sakas et. al.[10] use a fast DDA algorithm to identify voxels along the sampling ray using a raycasting approach. To reduce the cost of tri-linear interpolation, they use a nearest neighbor or an approximation to the maximum of a cell. Splatting[2], shear-warp[3], and cell projection[5, 6, 7] have also been used where the volume is preprocessed to speedup rendering. Mroz et. al.[6] allow for “windowed” transfer functions. Heidrich et. al.[4] use hardware acceleration to compute an approximation to DMIP. A limited set of isosurfaces is precomputed using Marching Cubes. The isosurfaces are sorted in depth by their isovalues and the z-buffer is used to compute the MIP. The size of the representation for very large datasets can overwhelm today’s graphics hardware even with a few isovalues. Yen et. al.[13] reduce computational complexity by clipping away portions of the volume using the near and far clipping planes to render oblique slabs of the volume. The application of transfer functions is supported, even though the volume is preprocessed into “opaque” runs in order to support fast rendering using shear-warp. The volume must be

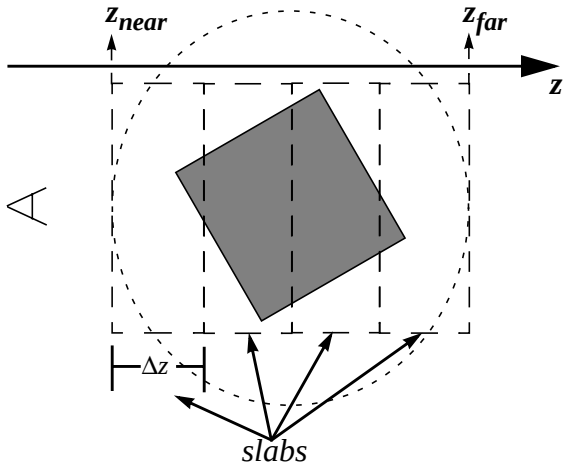


Fig. 1a: The view frustum is partitioned into *k* partitions along *z* between the near and far clipping planes. The result is an array of image-aligned slabs. Four slabs are shown here each with thickness Δz .

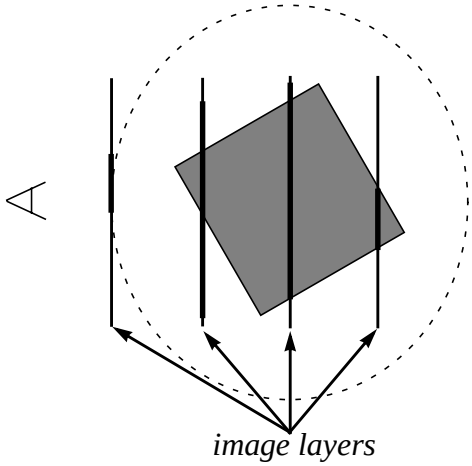


Fig. 1c: The resulting representation is an array of image layers storing a (*min*, *max*) function value pair per pixel. The bold portions on each image layer shown here depicts pixels receiving projected data information.

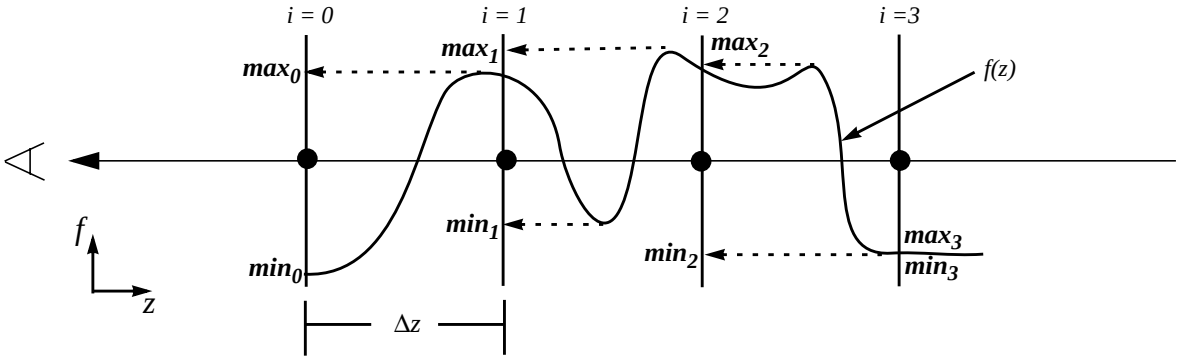


Fig. 1b: Each sampling ray defines a 1D scalar function $f(z)$ (a single ray is shown here), which passes through successive image layers (vertical lines numbered *0*, *1*, *2*, and *3*). Function values along a ray interval are projected towards an image layer pixel and both minimum and maximum operations are applied to the values to compute a resulting (min_i, max_i) value pair that is stored at the pixel (black dots).

re-processed when the transfer function changes. All of these techniques will have difficulty providing high frame rates for very large datasets.

3. A MULTI-LAYERED VIEW-DEPENDENT FUNCTIONAL REPRESENTATION

We describe, by construction, a novel image-based representation that stores scalar information at a chosen view sample. For now, we assume an orthogonal projection. The near clipping plane is placed at the image plane and the far clipping plane at a distance *D* that extends to include the entire volume. The view frustum is subdivided by image-aligned cutting planes, i.e. perpendicular to the view vector, placed along the *z*-axis at intervals of $\Delta z = D/k$, where $D = far_z - near_z$ and *k* is the number of divisions. This partitions the volume into *k* image-aligned slabs, as shown in Figure 1a. Data within a particular slab is projected towards the view sample to an image plane placed at the front face of the slab. This partitioning scheme divides each sampling ray into *k* ray intervals of length Δz . The function values within an interval are projected towards the corresponding image plane pixel as shown in (Figure 1b). The minimum and maxi-

mum over the sub-interval is stored at each pixel p for each image plane L_i (see equation 5). The resulting representation is a multi-layered array of images ordered from front-to-back, as shown in Figure 1c. We call each such image an *image layer.*

$$I_{L_i}(p) = (\min_{z_i \le z < z_{i+1}} (f(z)), \max_{z_i \le z < z_{i+1}} (f(z))), 0 \le i < k \tag{5}$$

4. MIP RENDERING USING IMAGE LAYERS

The construction just described, subdivides a sampling ray into k pieces and a function value pair (z_{min}, z_{max}) is computed for each piece. If the function is continuous, then all values between z_{min} and z_{max} are defined on the interval. When the user's camera coincides with the view sample, the k values contain complete information to compute an accurate projection. When the user's view is rotated a small angle away from the sample, a reprojection of these values computes an approximate warp. Texture mapping using standard graphics hardware acceleration can be used to quickly compute the reprojection. Image layers are texture mapped onto simple "billboard-like" geometry and then projected towards a novel view. Each "billboard" is placed at the middle of its corresponding slab.

It is useful to separate the *min* and *max* components of an image layer into two separate images, called the *min image* and the *max image*, respectively. We present algorithms using OpenGL[14] to render MIP images from image layers. First, OpenGL's **glBlendEquation** function is used with **GL_MAX** to perform a maximum operation on incoming fragments. To implement the original MIP approach each max image is loaded as a luminance texture and then the texture + geometry are projected to the current view. When the user's view coincides with the view sample, an accurate MIP image is computed because all max samples on the same ray map to the same framebuffer pixel. Each incoming pixel fragment is a resampled texture value. At nearby viewpoints, image layer projection warps pixels in the same interval together.

To compute a DMIP we additionally utilize the **glFog** function to apply the fog factor to the incoming fragment as a function of depth. Only the max images are projected as well. Three functions are available to weight a fragment using OpenGL: **GL_LINEAR**, **GL_EXP**, and **GL_EXP2**. Using **GL_LINEAR**, the slope of the weight function is set with $m = -1/(end - start)$.

The LMIP approach requires a front-to-back search along a sampling ray and the detection of a local maximum (see equation 3). This operation is not supported in graphics hardware. A ray interval may encompass many local maxima. An image layer pixel holds the largest over these maxima. We describe a two-pass algorithm using OpenGL's alpha and depth tests that computes similar visualizations to LMIP. First, a MIP is computed using **glBlendEquation** with **GL_MAX**, as just described, after clearing the framebuffer. In the second pass, blending is disabled and the alpha and depth tests are enabled. The **glAlphaFunc** is set with **GL_GEQUAL** and the user-defined threshold value. The **glDepthFunc** is set to **GL_LEQUAL**.

We only need the max image for any of these rendering techniques. To apply re-classification, the transfer function can be evaluated on each max image on a per-pixel basis whenever the transfer function changes. A classified layer, called a *classification image*, can be computed in this manner and used in the reprojection. Unfortunately, this construction is only a coarse sampling along the sampling rays. As defined in equation 4, the MIP operation should be applied to the transfer function g, which requires all values of f on a sampling ray. Rather than resampling f along the ray again, we can examine $g(x)$, $x \in (f_{min}, f_{max})$. If f is C^0-continuous, then all values in the range $[f_{min} \ldots f_{max}]$ exist on a ray interval. A classification image C can be computed from the min and max images by evaluating the transfer function for all values in the range of min/max values per-pixel, i.e.

$$C(p) = \max_{f_{min} \le f_k \le f_{max}} (g(f_k)) \tag{6}$$

This precludes the use of applying a transfer function on texture loads. A transfer function lookup table of size n, the number of samples in the scalar function domain, is commonly used to classify resampled values. When used here, interactive performance becomes difficult because multiple table lookups are required per-pixel, i.e. a lookup for each value in the min/max interval. Since the transfer function needs to be evaluated for a contiguous interval of domain values, we can reduce the table lookup cost to a single lookup per-pixel with a table that stores the right-hand side value of equation 6 for all possible intervals. We construct a table of size $n^2/2$ indexed by $(f_i, f_j - f_i)$, $f_i \le f_j$, for all i, j in the domain of f. The

size of the table is manageable for commonly used scalar field sizes, such as [0 ... 255]. The classification images can be used in conjunction with the previous MIP algorithms in the reprojection.

5. AN IMAGE-BASED BROWSER

Using our view-dependent approach, a view sample with associated image layers allows the user to roam nearby the sample at interactive rates using commodity graphics hardware. By computing a collection of view samples placed about the data, the user is able to browse the data without an appreciable degradation in the quality of the view reconstruction, if view samples are placed close enough. We have implemented an image-based browser that supports object-centered viewing, parallel projection, and zooming, using the image-based MIP algorithms described here. View samples are uniformly distributed on a bounding sphere. The user's viewpoint is restricted to the sphere and view reconstruction is computed with the image layers from the nearest view sample. Image layers are pre-computed off-line before browsing using a high quality renderer. The images are collected into a database indexed by view sample.

We envision a networked visualization framework in which a graphics client, e.g. a PC with an inexpensive graphics card, runs the browser and requests image layers from a dedicated server. Either the entire database of image layers can be loaded beforehand or, if too large, relevant image layers needed to reconstruct the current view can be requested and cached as needed by the browser. By trading off the cumbersome large volume data with images, the user is able to browse the data both spatially and parametrically using the MIP algorithms. Our implementation pre-fetches imagery that may be needed as the user moves. This paradigm may be viewed as a progressive refinement approach where an approximate view is computed between view samples and an accurate view is computed at a view sample. Thus, a view sample is akin to a key frame. This scheme also supports remote visualization and application on the web.

6. RESULTS

In this section we show renderings from our image-based browser implemented with C++ and OpenGL. All images were rendered on an SGI O2 with an R5000, 200 MHZ processor with 128MB memory. The image layers are computed with a modified high quality software splatting approach called Image-aligned Sheet Based Splatting[8] on an SGI Origin 2000 with four R10000, 225 MHZ processors and 1GB memory. In all results shown here, we use a 16 x 16 quadmesh for each image layer. The vertices of the mesh are each displaced in the *z*-direction to approximately fit the data to form a terrain-like mesh. This improves parallax at off view sample views. The image layer textures are subdivided into tiles so that a tile maps to a single quad. Tiles that contain no projected information are removed, with their corresponding quad, from the representation. We implemented a transfer function editor that allows the user to draw a 1D curve in free-hand and then apply the function to the view reconstruction on-the-fly. All of the images are rendered at interactive rates.

Figures 2 - 4 show renderings on two volume datasets obtained from the *volvis*[15] volume repository. Both have dimensions 256^3 and 1 byte per voxel. A dataset called "Aneurism" images a network of intertwined vessels. The other is called "Skull" and is a CT volume of a human skull. We compute 15 image layers at a chosen view sample and show the results of reprojection. Figures 2 (top left) and 4a show our MIP algorithm applied to the datasets when the user's view is coincident with the view sample. The transfer function is an identity function. Texture mapping and per-fragment blending utilize much of the rendering cost.

We illustrate our DMIP algorithm in Figures 2 and 3. The first row of Figure 3 shows the application of the MIP algorithm where the image to the right shows a reprojection at a view rotated ~13 degrees away from the view sample. Image quality does not degrade much at this deviation. It is easy to see how difficult it is to determine the visibility ordering of the vessels with the original MIP approach. Even animating about the view sample at multiple frames per second does not completely distinguish ordering. The second row shows renderings when the **glFog** function set with **GL_LINEAR**, **GL_FOG_START** = 0.0, and **GL_FOG_END** = 2.0 with respect to the corresponding two user's viewpoints used for the first row of images. The depth ordering of the vasculature becomes apparent in these images. The bottom image shows **GL_LINEAR** used with its default values. Only the vessels closest to the user have a large enough weight factor. Figure 4 shows renderings when **glFog** is set with **GL_EXP** and **GL_EXP2**. These functions also distinguish ambiguities found in the MIP renderings and provide a smoother transition over the linear function. The steepness of the function may be chosen to further emphasize closer structures. The **glFog** parameters are set with the GUI on-the-

fly.

A view sample is placed at the front of the skull's face of the "Skull" dataset and reprojections are shown Figure 4. The MIP rendering shown in Figure 4a shows two prominent vessels appearing at the top and bottom of the image. It is difficult to determine the depth of these structures due to their very high values. Figure 4b is a rendering using our version of the LMIP algorithm. At a threshold value of 55 from the range [0 ... 255] we notice that our algorithm can provide the same advantages as the LMIP algorithm where the material in front of the brighter vessels properly occludes the brighter vessels, e.g. the left eye-socket and in the neck. Also, the teeth in the front of the mouth occlude the teeth located in the back. Though contour artifacts due to the slab layering are apparent, they are not distracting and the threshold can be changed at interactive rates. Figures 4c and 4d show two different transfer functions applied with the original MIP algorithm. The transfer function used for Figure 5c contrast-enhances portions of the skull, while Figure 5d shows further contrast and isolation of the teeth and roots.

ACKNOWLEDGMENTS

We would like to thank the Department of Energy ASCI Program for generous support for this project. Additional support was provided through an NSF Career Award (#9876022). Equipment was provided by the ASCI project and by NSF grants (#9818319) and (9986052).

REFERENCES

1. M. J. Ackerman, "The Visible Human Project," *J. Biocomm.*, **18**, p. 14, 1991.
2. W. Cai and G. Sakas, "Maximum Intensity Projection Using Splatting in Sheared Object Space," *Proc. EUROGRAPHICS '98*, pp. C113-C124, 1998.
3. B. Csebfalvi, A. Konig, and E. Groller, "Fast Maximum Intensity Projection Using Binary Shearwarp Factorization," *TR-186-2-98-27 at the Institute of Computer Graphics, Vienna University of Technology*, 1998.
4. W. Heidrich, M. McCool, and J. Stevens, "Interactive Maximum Projection Volume Rendering," *Proc. IEEE Vis. '95*, pp. 11-18, 1995.
5. L. Mroz, A. König, and E. Gröller, "Real-Time Maximum Intensity Projection," *Proc. of the Joint EUROGRAPHICS - IEEE TCCG Symp. on Vis.*, pp. 135-144, 1999.
6. L. Mroz, A. König, and E. Gröller, "Maximum Intensity Projection at Warp Speed," *Technical Report TR-186-2-99-24, Institute of Computer Graphics and Algorithms, Vienna University of Technology, A-1040 Karlsplatz 13/186/2*, 1999.
7. L. Mroz, H. Hauser, and E. Gröller, "Interactive High-Quality Maximum Intensity Projection," *ComputerGraphics Forum*, Vol. 19, No. 3, pp. 341-350, 2000.
8. K. Mueller, N. Shareef, J. Huang, and R. Crawfis, "High-quality Splatting On Rectilinear Grids With Efficient Culling Of Occluded Voxels," *IEEE TVCG*, June, 1999.
9. H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Sobierajski Avila, K. Martin, R. Machiraju, J. Lee, "The Transfer Function Bake-Off," *IEEE CG&A*, **21**, pp. 16-22, 2001.
10. G. Sakas, M. Grimm, and A. Savopoulos, "Optimized Maximum Intensity Projection," *Proc. of 5th EUROGRAPHICS Workshop on Rendering Techniques*, pages 55-63, 1995.
11. Y. Sato, N. Shiraga, S. Nakajima, S. Tamura, and R. Kikinis, "LMIP: Local Maximum Intensity Projection - A New Rendering Method For Vascular Visualization," *J. of Comp.Assisted Tomography*, 22, 1998.
12. M. Woo, J. Neider, T. Davis, and D. Shreiner, *The OpenGL Programming Guide*, Third Edition, Addison-Wesley, 1999.
13. S. Y. Yen, S. Napel, and G. D. Rubin, "Fast Sliding Thin Slab Volume Visualization," *Proc. IEEE Volvis. '96*, pp. 79-86, 1996.
14. The Super High Field Magnetic Resonance Facility, The Ohio State University.
15. *http://www.volvis.org*

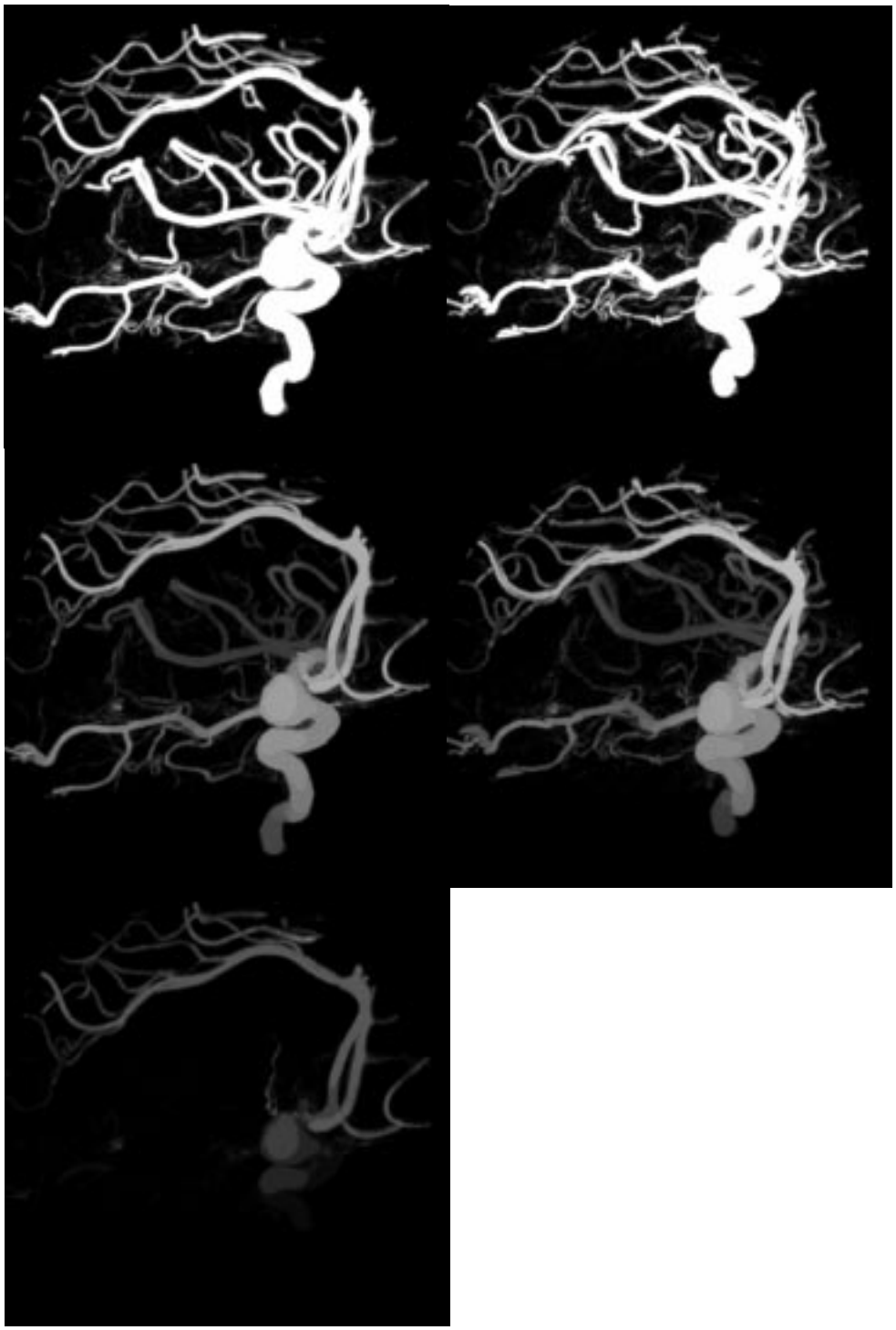


Fig. 2: The DMIP algorithm applied to the “Aneurism” dataset. The first column shows rendered images when user’s views is coincident with the view sample, while the second column shows a view rotated ~13 degrees away. The first row applies the MIP algorithm. The second row shows renderings with **GL_ FOG_END** = 2.0. The bottom image uses **GL_LINEAR** with default values.

.

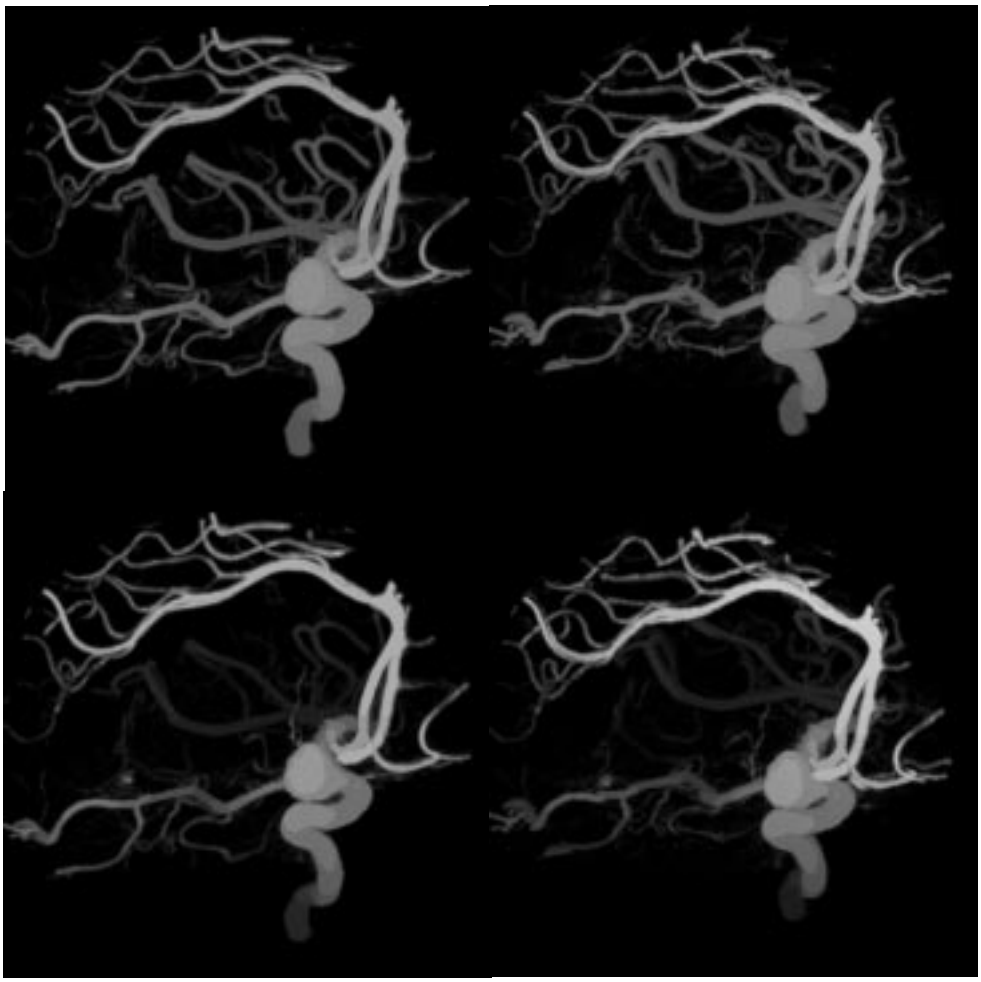
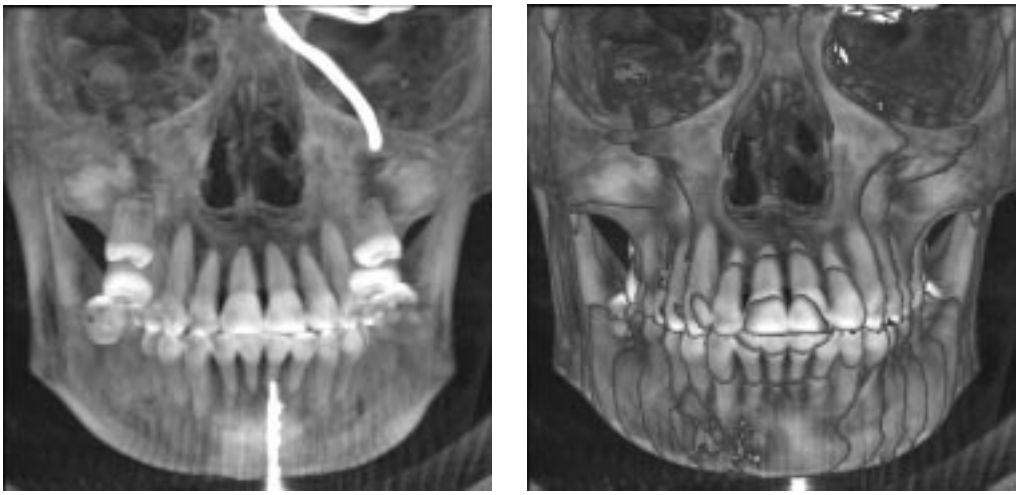


Fig. 3: The DMIP algorithm applied to the “Aneurism” dataset using **GL_EXP** and **GL_EXP2**. The viewpoints in each column here are the same as in the corresponding columns in Fig. 3. The first row shows the DMIP algorithm applied with **GL_EXP** and **GL_DENSITY** = 0.75. The second row shows **GL_ EXP2** with **GL_DENSITY** = 0.95.



Figs. 4a and b: The left image shows the MIP algorithm and the right image shows the LMIP algorithm with threshold = 55 applied to the “Skull” dataset.

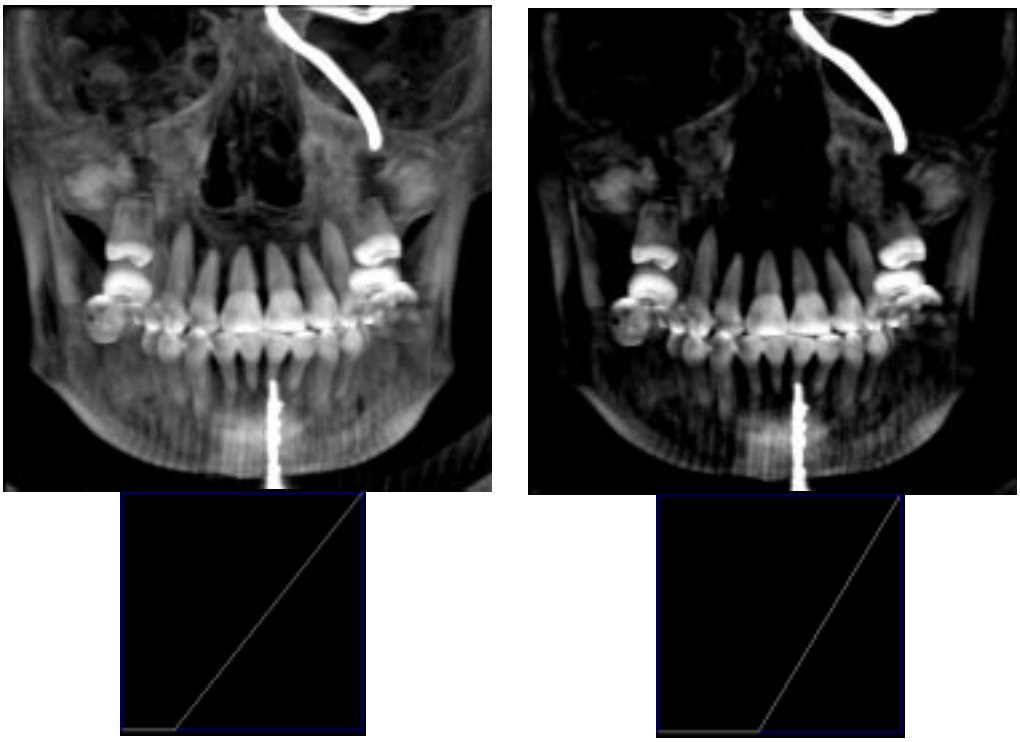


Fig. 5c: Two transfer functions constructed to reduce noise surrounding the skull and contrast-enhance portions of the bony structure.