Combined 3D Visualization of Volume Data and Polygonal Surfaces Using a Shear-Warp Algorithm

ANA ELISA F. SCHMIDT
TeCGraf - Departamento de Informática/PUC-RIO
anaelisa@tecgraf.puc-rio.br

MARCELO GATTASS
TeCGraf - Departamento de Informática/PUC-RIO
gattass@tecgraf.puc-rio.br

PAULO CEZAR P. CARVALHO
IMPA - Instituto de Matemática Pura e Aplicada
pcezar@impa.br

PUC-RioInf.MCC05/99 January, 1999

ABSTRACT: The present paper presents a study on integration between Shear-Warp and Z-Buffer algorithms, and extends the Shear-Warp algorithm to handle scenes composed of both volume and polygonal data. As volume data usually has a different resolution from that of the final image, in which Z-Buffer renders the polygonal data, several variants for this integration are analyzed. Results are shown to support some conclusions on the trade-off quality-versus-time that can be expected.

KEYWORDS: Shear-Warp, Z-Buffer, Polygonal Surface, Hybrid Volume Rendering.

RESUMO: Este artigo apresenta um estudo sobre a integração dos algoritmos de *Shear-Warp* e *Z-Buffer*; incluindo uma extensão do algoritmo de *Shear-Warp* para tratamento de cenas compostas de dados volumétricos e dados poligonais. Como o dado volumétrico usualmente possue uma resolução diferente da resolução da imagem final a ser criada, na qual o *Z-Buffer* renderiza o dado poligonal, são analizadas algumas variações para esta integração. Resultados são apresentados, dando suporte às conclusões que avaliam a relação entre qualidade da imagem final produzida e tempo de processamento envolvido.

PALAVRAS-CHAVE: *Shear-Warp*, *Z-Buffer*, Superfície Poligonal, Renderização Volumétrica Híbrida.

1. Introduction

In some medical applications it is necessary to display scenes composed of both volume data and polygonal surfaces. A combined view of radiation treatment beams or bone prostheses over the patient's anatomy are two examples of these applications [6]. Furthermore, the rendering process must be fast enough to achieve interactive display rates. Such rates are necessary for an immediate feedback when the user changes the viewing parameters. This is essential for data exploration and for aiding new surgery procedures augmented with 3D imaging.

Several strategies have been proposed to combine volume and surface data using well known volume-rendering techniques such as Ray-Casting and Splatting combined with polygon techniques like Ray-Tracing, Z-Buffer, and Scan-Conversions [2,6,8,11,12]. These classical volume-rendering techniques, however, are not fast enough to provide interactive response times [3].

To improve the efficiency of volume-rendering algorithms, Lacroute and Levoy [3] proposed an interesting method, called Shear-Warp, which geometrically transforms the data volume, thus simplifying the projection and composition stages. A parallel version of this algorithm was capable of producing, from a data set of 256^3, a frame rate of 15Hz in a Silicon Graphics Challenge with 32 processors [5].

More recently, the quest for interactive response time in volume rendering has brought up new approaches, using mainly specialized hardware and 3D-texture mapping. Osborne and others [9] presented the EM-Cube, a scheme for a specialized PC card that implements the Cube-4 proposal [10].

Wilson, Gelder and Wilhelms [13] presented a method that uses 3D texture and can be implemented on top of an OpenGL 1.2 graphical system. Interaction time can be achieved on high-performance graphics systems that implement 3D textures in hardware such as the SGI InfiniteReality Engine. When the hardware supports only 2D textures, a variant of the method can be used. This variant, however, produces noticeable sampling errors when the viewing direction is not aligned with one of the data-volume axes [7].

Currently, almost every graphics board can efficiently render polygon-based scenes with the Z-Buffer algorithm. The major variation among these boards is the number of triangles per frame they can render and the size and quality of the texture maps. Most of the currently available hardware supports 2D texture maps, and only a few support 3D textures. In any case, the size of the texture memory is always a problem. As computer

technology improves, so does the size of the data sets we must handle, and the best methods are those that can efficiently manage the computer resources.

Shear-Warp algorithms have the ability to deal with volume data as a sequence of 2D textures without the unwanted sampling errors that occur when these textures are directly projected on the screen. If well ordered, this access simplifies memory management between texture, main, and secondary memory.

Compared to the new 3D texture-mapping techniques, however, Shear-Warp lacks generality. One of the most important capabilities missing in the method is the handling of combined volume data and polygonal surfaces. This paper presents an extension of the Shear-Warp algorithm to deal with this case efficiently.

Shear-Warp algorithms that combine volume and polygonal surfaces must deal with two problems: aliasing and visibility. Aliasing is introduced during the surface-sampling process if the volume resolution is low, and surface polygons are rendered at that same resolution. The visibility at pixels in which voxels and surface patches mix may also present some problems related to low-resolution volumes. Levoy [6] discusses these problems in the context of the Ray-Casting algorithm.

Three strategies are discussed here to minimize aliasing and to correctly compute visibility during the volume composition process.

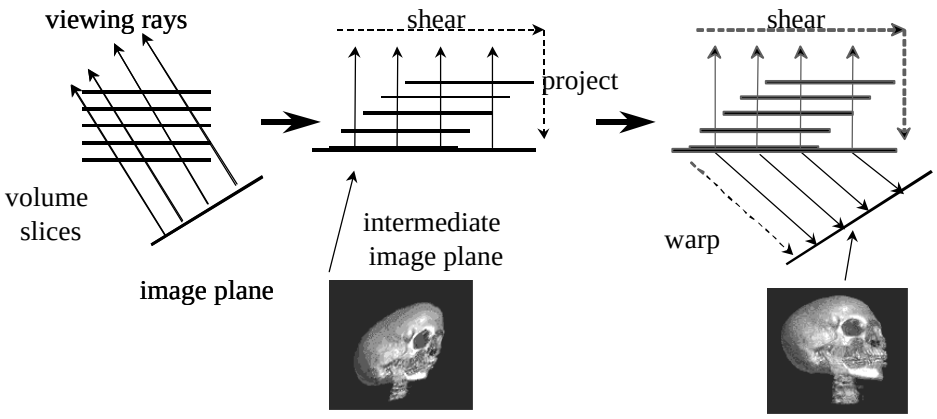
2. Shear-Warp Factorization

Shear-Warp factorization decomposes the visualization matrix into two matrices: a shear matrix and a warp matrix:

$$M_{view} = M_{view} \overbrace{\begin{bmatrix} 1 & 0 & -s_i & -t_i \\ 0 & 1 & -s_j & -t_j \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{M_{warp}} \overbrace{\begin{bmatrix} 1 & 0 & s_i & t_i \\ 0 & 1 & s_j & t_j \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{M_{shear}} . \quad (1)$$

The visualization matrix, M_{view}, transforms the volume from the standard coordinate system (i, j, k) to the final image space according to viewing parameters. The standard coordinate system (i, j, k) is obtained from the original object coordinate system (x, y, z) by means of a permutation matrix P. The k axis is chosen to be either x, y or z according to which one is the most parallel to the viewing direction. The two other axes are labeled i and j [4].

Figure 1 shows a generic scheme of the Shear-Warp algorithm for parallel projections. Horizontal lines represent the volume slices.



$$M_{view} = M_{Warp2D} * M_{shear3D}$$

Figure 1 - Overview of Shear-Warp factorization

The shear matrix transforms each slice from the standard object space into the sheared object space. After shear transformation, slices are projected to generate an intermediate image. Each sampled slice is composed in front-to-back order, using the *over* operator, to form the distorted intermediate image. Finally, the warp matrix transforms the distorted intermediate image into the final image.

In the sheared object space, the viewing direction is perpendicular to the volume slices. Moreover, since shear only translates the slices, without any rotation about the *k* axis, each scanline of the volume remains aligned with the scanlines of the intermediate image. This property is fundamental for simplifying the rendering algorithm, thus allowing run-length encoding and coherent image-composition algorithms.

The intermediate image dimensions, in pixels, are calculated as follows:

$$\begin{aligned} w_{Low} &= w_{Slice} + d_{Slice} * s_i \\ h_{Low} &= h_{Slice} + d_{Slice} * s_j \end{aligned} \quad , \tag{2}$$

where w_{Slice} and h_{Slice} are, respectively, the width and height of each slice, and d_{Slice} is the number of slices [4].

It is important to note that, in the standard Shear-Warp algorithm, the intermediate image resolution basically depends on the volume resolution and on the projection direction; it does not depend on the final image resolution. This yields an important property of the Shear-Warp composition process: its complexity is proportional to the

volume resolution, but not to the final image resolution. Since volume resolutions are usually smaller than the final image resolution, this property results in a fast algorithm. The warp transformation not only corrects the distorted intermediate image but also re-samples it to the final image resolution.

Another important speed-up factor is obtained by the scanline alignment in this algorithm. Figure 2 illustrates the use of this alignment to avoid unnecessary computations during the process of blending a slice into the intermediate image. Generally, a voxel contributes with four pixels in the intermediate image and, conversely, a pixel usually receives contribution from four voxels. Moreover, neighboring pixels in an image scanline receive neighboring voxels in a slice scanline. Furthermore, if the slices are blended in the front-to-back order, as the intermediate image pixels become opaque (α=1) they cease to receive voxel contributions. Conversely, transparent voxels (α=0) can also be ignored, since they do not contribute. In Figure 2, only the two pixels labeled *modified pixels* need to receive contributions from the slice.

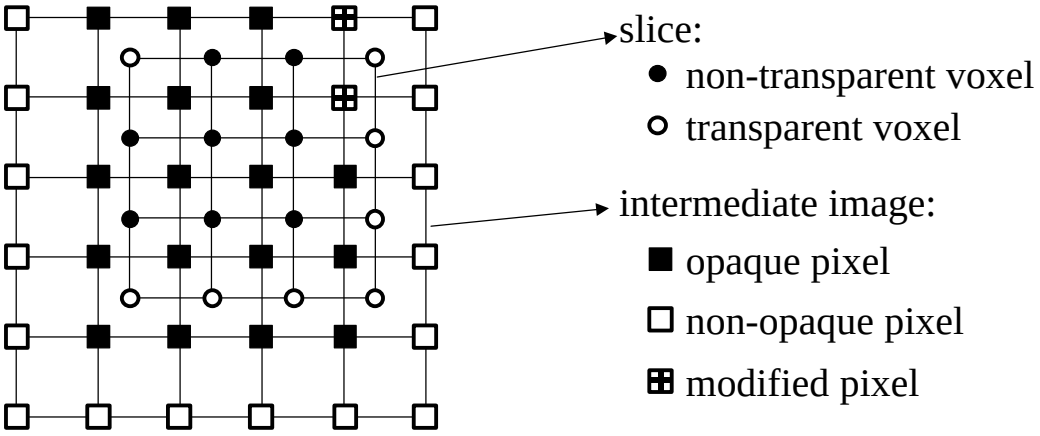


FIGURE 2 - SLICE COMPOSITION IN LACROUTE'S SHEAR-WARP ALGORITHM

Lacroute's implementation of Shear-Warp [4] uses clever encoding techniques for the intermediate image and volume slices. For the intermediate image he uses an auxiliary field to indicate, for every pixel, how many pixels can be skipped until the method can find a non-opaque pixel. Volume slices are run-length encoded as transparent and contributing voxels, also allowing the algorithm to skip the former.

3. Combining Volume and Surfaces

The Z-Buffer algorithm [1] can be used to render polygonal surfaces yielding color and depth (Z) matrices. The depth matrix is used as a mask to determine which voxels are mixed with the surface. The color matrix is blended with the transparent volume color lying in front of it. Figure 3 shows a generic scheme used to combine volume and surface information using the Shear-Warp and Z-Buffer algorithms.

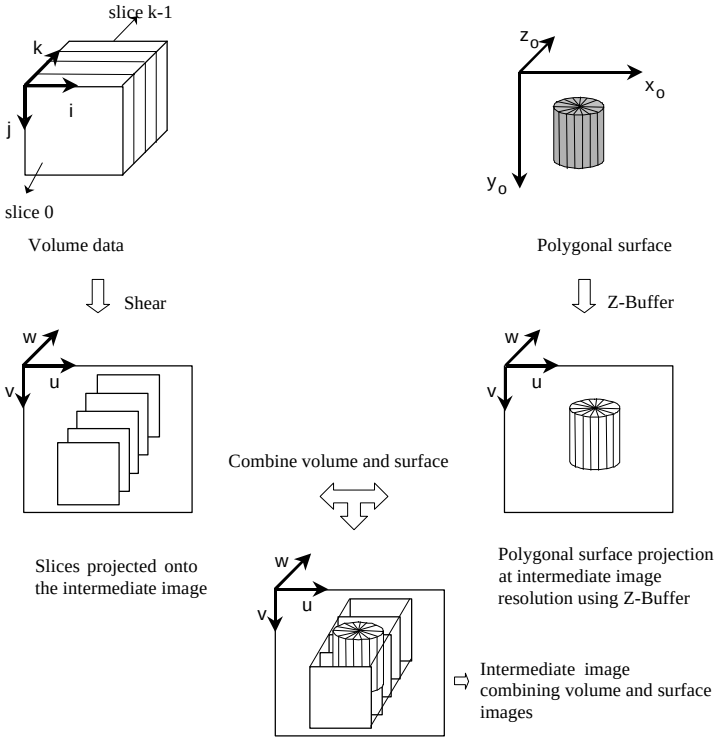


Figure 3 - Basic combination scheme

The blending process that generates the intermediate image must take surface contribution into account. For this reason, the surface must be sampled at the same resolution as the intermediate image. If this resolution is low compared to the final image resolution, then artifacts and aliasing effects are introduced at this stage.

Resolution of the Intermediate Image

To avoid low-resolution problems, the polygons must be rendered into an intermediate image whose resolution is comparable to the final image resolution. To find this resolution, let us consider the conventional warp matrix. As discussed above, if the volume resolution is low, the warp matrix not only corrects the distortions, but also re-samples the intermediate image to the final image resolution.

In order to avoid aliasing, the intermediate image resolution must be re-defined so that the new warp matrix maps intermediate image vectors into vectors of about the same length in the final image. That is, ideally the dimensions of a pixel in the intermediate image should not change significantly as the pixel is warped to the final image. Nevertheless, that is not the general case in normal applications of the Shear-Warp algorithm. Usually, the volume resolution is smaller than in the final image, and a pixel of the intermediate image is warped into *M* x *N* pixels of the final image.

Factors *M* and *N* can be computed from the warp matrix $M_{warp} = [w_{ij}]$, as Figure 4 shows. A unit step in the *u* and *v* directions in the intermediate image leads to increments of (w_{00}, w_{10}) and (w_{01}, w_{11}), respectively, in the final image coordinate system.

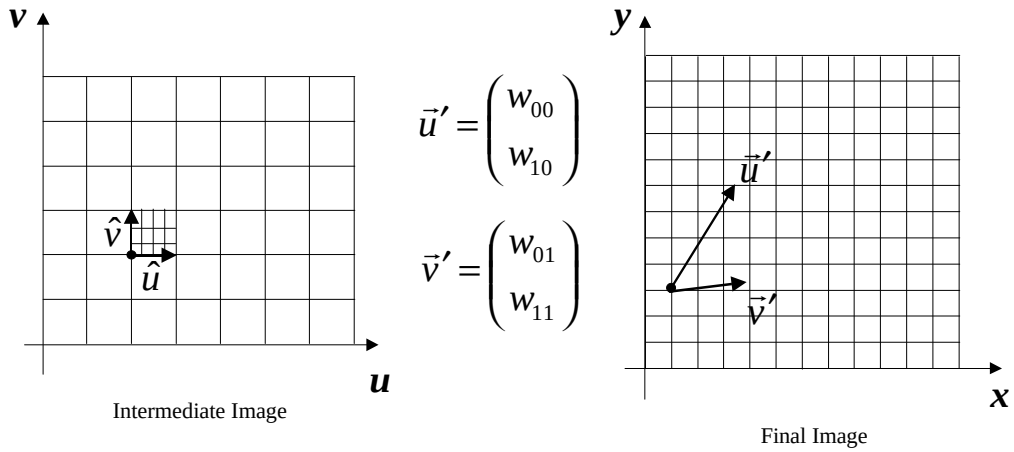


Figure 4 - Mapping intermediate space into final image space

Thus, *M* and *N* can be determined as:

$$\begin{aligned} M &= ceil(\max(|\, w_{00} \,|, |\, w_{10} \,|, 1)) \\ N &= ceil(\max(|\, w_{01} \,|, |\, w_{11} \,|, 1)) \end{aligned} \quad . \tag{3}$$

If the low-resolution intermediate image, given by equation (2), is increased by factors *M* and *N*, then a unit increment in the final image coordinates corresponds to approximately a unit increment in the new high-resolution intermediate image, whose size (w_{High}, h_{High}) is given by:

$$\begin{aligned} w_{High} &= M(w_{Low} - 1) + 1 \\ h_{High} &= N(h_{Low} - 1) + 1 \end{aligned} \quad . \tag{4}$$

This corresponds to dividing each pixel interval in the low-resolution intermediate image into *M* x *N* sub-pixels, as illustrated by Figure 5. This sub-division increases the

resolution, preserving the pixel positions of the low-resolution image. This is important for the multi-resolution images used in the multi-resolution composition algorithms to be presented below.

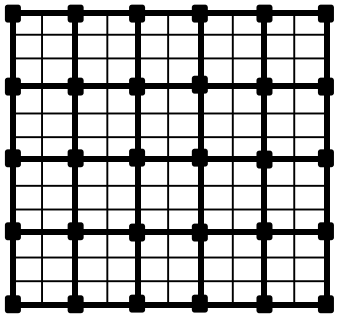


Figure 5 - Example of image scaling

Shear-Warp with Z-Buffer

The basic idea for combining the Shear-Warp and the Z-Buffer algorithms is to replace the surface polygons by their fragments represented in the color and depth buffers. With these buffers, the composition step of the Shear-Warp algorithm can be modified to include the surface contribution, thus yielding a hybrid algorithm. This algorithm blends the surface fragments that lie between two slices into the intermediate image as the slice composition progresses. The warp step remains unchanged.

If the color and depth buffers have a resolution that is different from the one corresponding to the volume slices, then the composition step becomes more complex. This happens when the surface is sampled to an intermediate image in high resolution. A simple way to avoid this complexity is to re-sample the slices after the classification step. That is, if the intermediate image resolution is increased by factors *M* and *N* given in equation (3), the image corresponding to the slice's colors and opacity must also be scaled by these factors. An important detail in this image scaling process is that RGBA channels should not be independently linearly interpolated. RGB channels must be weighted by the opacity prior to the interpolation process [14].

However, as previously noted, one of the key factors for the speed of Shear-Warp algorithm is the use of low resolution in the costly volume-composition operation. Scaling the slices by factors *M* and *N* reduces this advantage significantly. An alternative strategy consists in working with two intermediate images: one in low and another in high resolution. In areas in which the alias problem appears, the high-

resolution image is used in the composition process; other areas use the low-resolution image.

The algorithms proposed here are then classified in three groups:

- **Low-resolution composition**: intermediate image and surface in low resolution.
- **High-resolution composition**: colors and opacities of volume slices interpolated during the composition process; intermediate image and surface in high resolution.
- **Multi-resolution composition**: intermediate image in both low and high resolution; surface rendered in high resolution.

Algorithms in all three groups use the Z-Buffer algorithm to render the surface into a color and depth buffer whose resolution is defined by the intermediate image resolution.

Z-Buffer Matrix

The projection matrix used by the Z-Buffer algorithm maps the polygonal surface from the object space into the intermediate image space. Note, however, that the projection discussed here is not the one that projects objects into the final image. The projection matrix used here is the one that projects objects into the intermediate image, prior to the warping stage which builds the final image. Thus, it is based on the intermediate image resolution and on the shear parameters, as shown below.

The matrix for performing this transformation is given by:

$$M_{projSurf} = S_{res} M_{shear} P,$$

where P is the permutation matrix, M_{shear} is given in equation (1), and S_{res} is

$$S_{res} = \begin{bmatrix} \frac{w_{High}}{w_{Low}} & 0 & 0 & \frac{w_{High}}{w_{Low}} - 1 \\ 0 & \frac{h_{High}}{h_{Low}} & 0 & \frac{h_{High}}{h_{Low}} - 1 \\ 0 & 0 & k_{incr} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

when the intermediate image has high resolution, and

$$S_{res} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k_{incr} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

when the intermediate image has low resolution. Note that, since the *k* axis may need to be re-oriented, k_{incr} may be +1 or -1 according to the viewing vector orientation.

Low-Resolution Composition Algorithm

Low-resolution composition is the simplest method proposed here. The intermediate image resolution is kept with the values computed with the standard Shear-Warp algorithm. The basic steps of this algorithm are:

```
Compute the Z-Buffer projection matrix (equation (5));
Render surface into color and depth buffers;
For each volume slice k
   Compute the offsets (slice_u, slice_v) of the slice into the intermediate
   image;
   Compute weight factors for each set of 4 voxels  that contribute to one pixel
   (Figure 6);
   For each slice scanline j
      Compute the corresponding scanline v in the low-resolution intermediate
      image;
      While there are still unprocessed voxels in this scanline j
            Search for the first non-opaque pixel, (u, v)
            Find the 4 voxels (TL, TR, BL and BR) that contribute to the
            (u, v) pixel;
            If one of the current 4 voxels is non-transparent:
               Interpolate the contributions of the 4 voxels;
               Add their contribution to (u, v) pixel;
            else
               Skip transparent voxels in both scanlines j and j+1 updating pixel
               position u;
   Blend into intermediate image the contribution of the portion of the surface
   that lies between slices k and k+1;
Warp the intermediate image into the final image.
```

To speed up the voxel-blending step, RLE structures used by Lacroute [4] are employed to store non-transparent voxel values and gradients. That is, only those voxels that map to an opacity greater than zero are considered. The stored values allow significant space saving, without restricting the illumination model. Lights can be attached to the observer, and this effect can be perceived as he or she navigates through the volume.

The encoding of the intermediate image stores the number of fully opaque pixels that can be skipped in a scanline during the composition process. Since the blending operator works in front-to-back order, the slices that are closer to the observer are blended first. As pixels become opaque they do not accept further contribution; thus, the encoding seeks to efficiently skip these pixels.

An important aspect of this blending process is illustrated in Figure 6. Since both the intermediate image and the slice have the same resolution, only four voxels can contribute to a given pixel **p**.

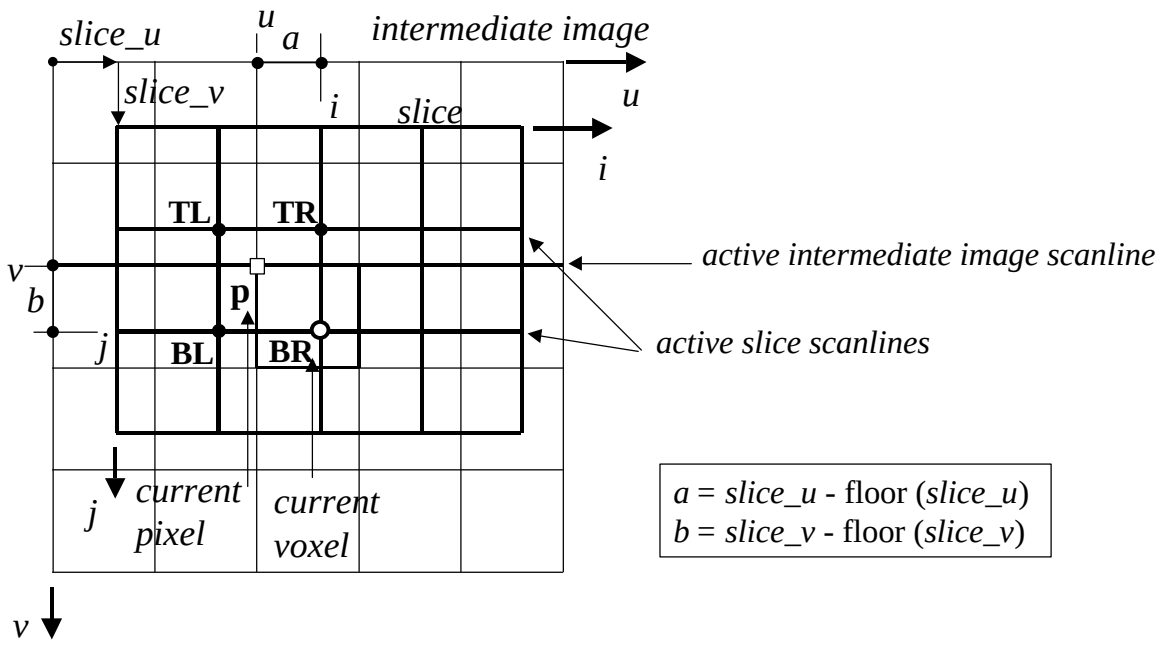


Figure 6 - Standard Shear-Warp slice composition

The contribution of voxels **TL**, **TR**, **BL**, and **BR** to pixel **p** is given by the bi-linear interpolation

$$\begin{aligned} C'_v &= (1-a)(1-b)C'_{BR} + (a)(1-b)C'_{BL} + (1-a)(b)C'_{TR} + (a)(b)C'_{TL} \\ a_v &= (1-a)(1-b)a_{BR} + (a)(1-b)a_{BL} + (1-a)(b)a_{TR} + (a)(b)a_{TL} \end{aligned} \quad , \qquad (6)$$

where a is the opacity, and $C' = a\,C$ is the opacity-weighted color channel (aR, aG, aB) of each one of the four voxels. These voxel colors and opacities are computed from user-provided transfer functions modulated by the illumination model.

This interpolated color is then blended into the pixel **p** by:

$$\begin{aligned} C'_{p,new} &= C'_{p,old} + C'_v(1-a_{p,old}) \\ a_{p,new} &= a_{p,old} + a_v(1-a_{p,old}) \end{aligned} \quad . \qquad (7)$$

As the opacity gets closer to 1, the contribution of the voxels becomes less important.

To accelerate the surface-contribution step, an auxiliary list is produced with the pixels ordered by their depth. So, in order to find the surface contribution between each slice *k* and *k*+1, the algorithm does not have to search all elements of the depth matrix. Equation (7) is also used to blend the color and opacity of the surface into the corresponding pixels.

High-Resolution Composition Algorithm

In this algorithm, the intermediate image resolution and the Z-Buffer color and depth buffers are set to high resolution (see equation (4)). The volume slices are composed into the high-resolution intermediate image with the colors and opacity interpolated as illustrated by Figure 7. In this figure, each pixel in high resolution is considered as a sub-division of a low-resolution pixel.

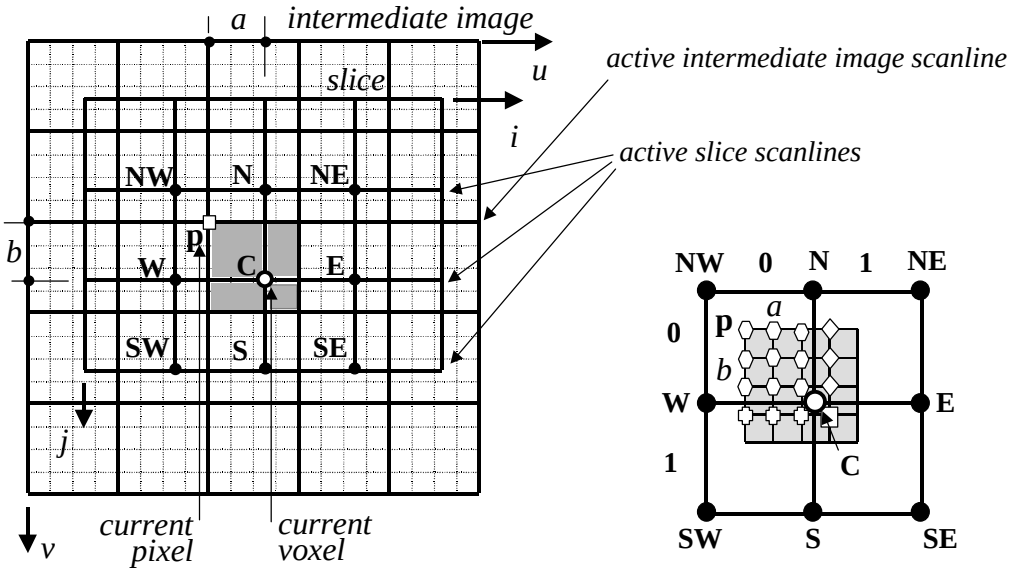
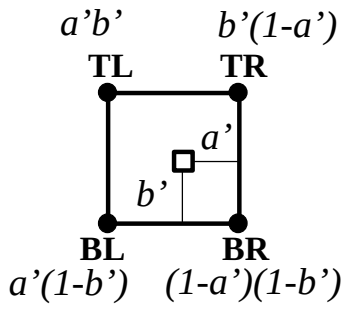


Figure 7 - Blending low-resolution slices into a high-resolution intermediate image

The strategy used here seeks to use the RLE structure of the low-resolution intermediate image and volume slices. The correspondence between current pixel (in low resolution) and current voxel that exists in Figure 6 continues to exist in Figure 7. The main difference is that, in the former, only four voxels contribute to one pixel, and, in the latter, nine voxels are required to define all sub-pixels of a low-resolution pixel. The contributions of these nine voxels, however, do not occur at the same time. For each sub-pixel, only four voxels contribute, as illustrated by Figure 8. Note in this figure

that sectors marked by the symbols (hexagon, diamond, etc.) are the ones shown in Figure 7, and that voxels **TL**, **TR**, **BL**, and **BR** have the same function as the ones shown in Figure 6. In Figure 8, for each sub-pixel, the distances *a'* and *b'* play the same role in defining the voxel weighting factor as do *a* and *b* in equation (6).



Sector		TL	TR	BL	BR
00	⬡	NW	N	W	C
01	◇	N	NE	C	E
10	⯃	W	C	SW	S
11	□	C	E	S	SE

Figure 8 - Voxels that contribute to a pixel and their weighting factors

Considering the sub-pixels indexed by *m* ranging from 0 to *M*-1 in the *u* direction, and by *n* from 0 to *N*-1 in the *v* direction, each of the four sectors is defined as shown in Table 1:

Sector	m_{min}	m_{max}	n_{min}	n_{max}
00	*0*	*floor(a∗M)*	*0*	*floor(b∗N)*
01	*floor(a∗M) +1*	*M-1*	*0*	*floor(b∗N)*
10	*0*	*floor(a∗M)*	*floor(b∗N)+1*	*N-1*
11	*floor(a∗M) +1*	*M-1*	*floor(b∗N)+1*	*N-1*

Table 1 – Sectors limits

As noted before in this section, the composition step uses the same run-length information used in the low-composition algorithm. The main differences are the number of active scanlines, and the opacity criteria for intermediate image pixels. Here, the algorithm uses three active scanlines in the run-length encoding of the current slice to contribute to one scanline in the intermediate image. Regarding the opacity criteria,

only when all sub-pixels are opaque the correspondent pixel in low resolution can be considered opaque. In this case, the composition algorithm can skip this pixel.

Finally, the warping step must be adjusted to transform this high-resolution intermediate image into the final image. The adjustment consists in multiplying the first two rows of the warping matrix by factors *M* and *N*, respectively.

Multi-Resolution Composition Algorithms

The basic idea in this class of algorithm is to increase the resolution of the intermediate image only in the regions that are influenced by the surface. To identify these regions, which require high resolution, two criteria are proposed here: (a) surface footprint, and (b) high-frequency regions. These criteria yield two multi-resolution algorithms. In the first one, all pixels of the low-resolution intermediate image influenced by the surface are marked to require high resolution. In the second criteria, the algorithm first applies the 3x3 Laplacian filter, given below, to the surface color map:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} .$$

This convolution produces a new image in which the high-frequency regions correspond to high values in each RGB channel. These high values correspond to borders and to irregularly lightened regions of the surface. Determining the high-frequency values depends on a threshold provided by the user. The lower the threshold, the larger the number of pixels to be super-sampled.

Footprint multi-resolution algorithms require two auxiliary intermediate images: one in high resolution, and one in low resolution. The high-resolution auxiliary image is used to blend the voxels and the surface into pixels identified by the footprint criteria. The low-resolution auxiliary image is used to compose only volume data.

The high-frequency algorithm uses an additional low-resolution auxiliary intermediate image to compose volume and surface data. Figure 9 illustrates the need for two low-resolution images. In this figure only the pixels marked with a diamond are computed in high resolution; the others (squares and hexagons) are computed by re-sampling the low-resolution intermediate images. To correctly compute the hexagons in

the high-resolution intermediate image, one must take into account the low-resolution image that contains both volume and surface data. If this same low-resolution image is used to compute the square pixels, however, the surface border color would be blurred, as illustrated by the figure. To correctly interpolate pixels that do not receive surface contribution (square pixels), the algorithm uses the low-resolution auxiliary intermediate image in which only voxel contributions are computed. To identify which pixels lie outside the surface projection, the Z-Buffer depth matrix is used.

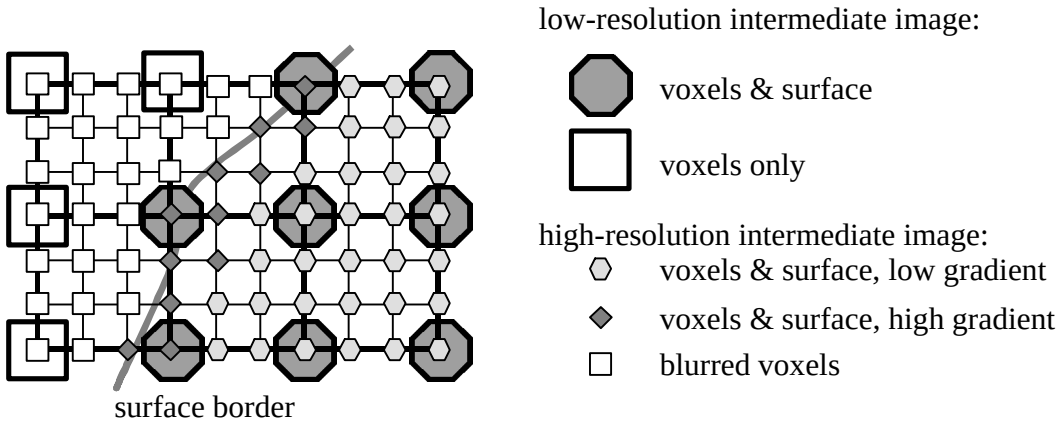


Figure 9 - Blur produced by interpolation in the low-resolution surface borders

In both multi-resolution algorithms, the versions of the intermediate image, in high and low resolution, are assembled into a single high-resolution image before the warping transformation. The warp process is similar to the one applied to low-resolution intermediate images. The difference is that factors *M* and *N* must be integrated into the warp matrix, correctly transforming the high-resolution intermediate image into the final image.

4. Test Results and Discussion

The ideas proposed here were implemented in an in-house volume-rendering program. Five versions of the Shear-Warp algorithm were considered: (1) standard (without surface polygons); (2) low resolution; (3) high resolution; (4) footprint in high resolution; and (5) high frequency in high resolution. The algorithms use orthographic projection and implement volume and intermediate image RLE encodings to accelerate the composition process. To minimize the amount of memory used by RLE volume structures, the program maps the gradient vectors into a pre-computed table with 256 uniformly-distributed normal vectors, reducing the gradient representation to one byte

per voxel. Both the RLE volume representation and the gradient quantization are a pre-processing step of the algorithm. The corresponding processing times are not included in the results presented in this section.

The system used in the tests was an Intergraph Realizm II TDZ 2000, with dual Pentium II 300 MHz and 512 Mb RAM running Windows NT. To better control the numerical experiments presented here, we made no use of the graphics hardware. That is, our software implements all steps of all algorithms, including the Z-Buffer. If one step were performed by the graphics hardware, the measured time could not be compared with the others. To make the tests simpler, the parallel capability offered by dual Pentium were not used either.

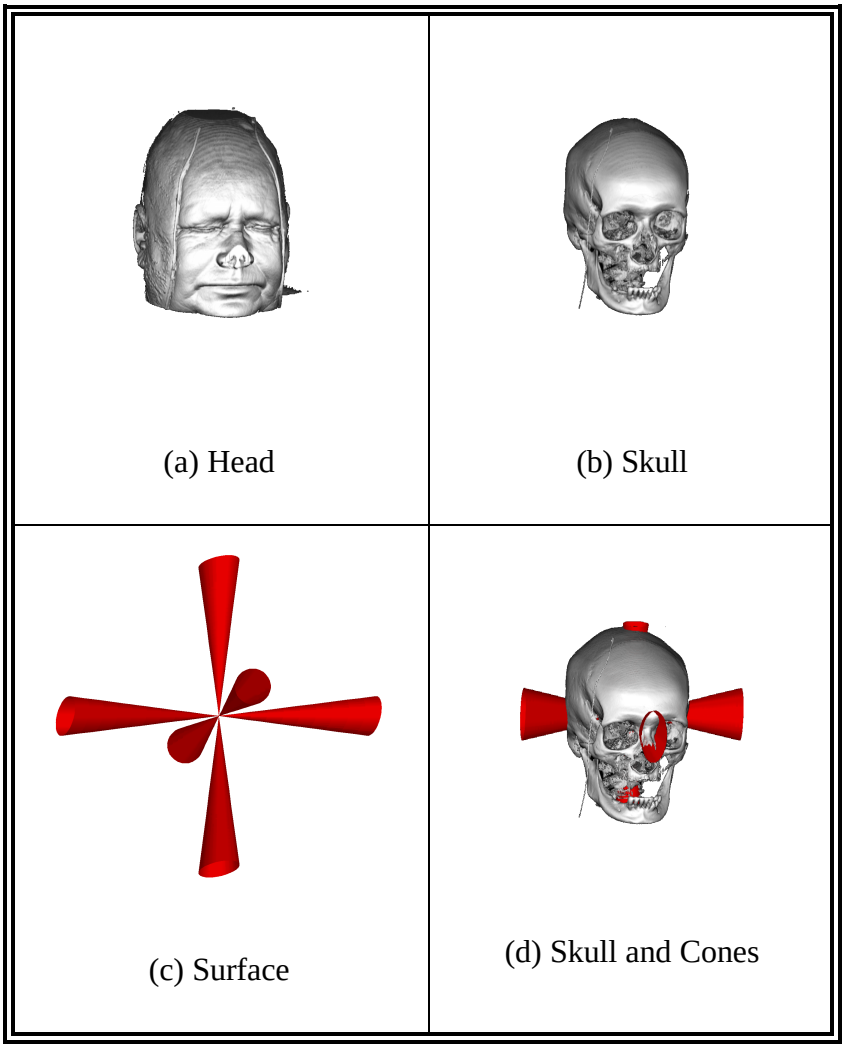
The data set used here is the CT exam of Visible Woman's head [15], shown in Figure 10(a). The head data set was assembled from slices 1001 to 1209 of the whole woman's body. Each slice has a resolution of 512x512, and the basic data set has 512x512x209 voxels. To study the effect of the volume size on the algorithms, we have filtered this data set to produce two smaller ones: 255x255x103 and 127x127x51. We also uniformly converted the original 12-bit density values into a one-byte representation, that is, with density values ranging from 0 to 255.

The speed of a direct volume-rendering algorithm is also very dependent on the transfer functions. To visualize the Visible Woman's skull, shown in Figure 10(b), the opacity transfer function α and the color transfer function C used in the tests were:

$$a = \begin{cases} 0.0, & if \ \ v \in [0,100]; \\ 1.0, & if \ \ v \in (100,255]. \end{cases}$$

$$C = \begin{cases} (0 \ \ 0 \ \ 0), & if \ \ v \in [0,30]; \\ (255 \ \ 255 \ \ 255), & if \ \ v \in (100,255]. \end{cases}$$

The polygonal surface model is composed of six cones that are parallel to the X, Y, and Z object space axes, as illustrated by Figure 10(c). About 300 quadrilateral polygons are used to model each cone. All cones are completely opaque (α = 1.0). Figure 10(d) shows a final image in which volume data and polygonal surfaces are combined.



(a) Head (b) Skull

(c) Surface (d) Skull and Cones

Figure 10 – Visible Woman's head data set and cones polygonal model

Other important rendering parameters used in the tests are shown in Table 2.

Minimum opacity value	0.05	Noise threshold	10
Maximum opacity value	0.95	High-frequency threshold	100
Ambient light	10%	Final image resolution	500^2

Table 2 – Parameters used in the tests

Table 3 shows the CPU time, in seconds, for the main steps of the five implemented algorithms, using the three versions of the volume data: 127x127x51, labeled *small*; 255x255x103, labeled *medium*, and 512x512x209, labeled *large*. Since the speed of the

algorithms is also dependent on the projection direction, the times presented in this table were computed as an average of 50 randomly distributed projection directions.

From Table 3, a series of observations can be made. Without hardware assistance, the time spent on the Z-Buffer surface-rendering step is considerable if compared to that of other steps, and is proportional to the intermediate image resolution. This is a strong indication that a hardware implementation of the Shear-Warp steps would yield interactive time.

The time for composing volume slices and surface polygons into the low-resolution intermediate image is smaller than the time for composing only slices. This happens because polygonal surfaces cause the intermediate image pixels to become opaque at an earlier time. Thus, as we are using the RLE volume and image optimizations, the volume and slice composition requires fewer computations than composing the slices alone.

In all algorithms and in most cases, the time taken to create the intermediate image is larger than the time spent in each of the other steps. Only in the low composition algorithm with small volume data, this did not happen. As the number of pixels in the intermediate image increases, the time spent in the composition step also increases. Note, however, that the number of pixels to be processed depends on the algorithm, on the geometry of the polygonal surface, and on the projection direction.

Visible Woman's Skull 500x500 Final Image	Volume size	No Surface	Low	High	Multi-Resolution	
					Footprint	High Frequency
Surface Z-Buffer Rendering	*small*	0,00	0,18	0,52	0,51	0,51
	medium	0,00	0,33	0,64	0,64	0,64
	large	0,00	0,85	0,93	0,91	0,93
High-frequency Determination	*small*	0,00	0,00	0,00	0,00	0,02
	medium	0,00	0,00	0,00	0,00	0,03
	large	0,00	0,00	0,00	0,00	0,05
Marking High-resolution Portions into Intermediate Image	*small*	0,00	0,00	0,03	0,03	0,02
	medium	0,00	0,00	0,04	0,03	0,02
	large	0,00	0,00	0,06	0,04	0,03
Creating the Intermediate Image	*small*	0,09	0,08	0,94	0,49	0,51
	medium	0,34	0,31	1,62	1,09	0,97
	large	1,32	1,25	3,82	3,11	2,41
Composing Surface Contributions between Slices	*small*	0,00	0,00	0,03	0,03	0,03
	medium	0,00	0,01	0,04	0,04	0,04
	large	0,00	0,02	0,07	0,07	0,07
Composing Slice Contributions	*small*	0,09	0,08	0,91	0,34	0,29
	medium	0,34	0,31	1,58	0,86	0,62
	large	1,32	1,24	3,75	2,79	1,95
Assembling Intermediate Images	*small*	0,00	0,00	0,00	0,12	0,18
	medium	0,00	0,00	0,00	0,19	0,31
	large	0,00	0,00	0,00	0,25	0,39
Warping	*small*	0,17	0,17	0,18	0,18	0,18
	medium	0,18	0,17	0,19	0,19	0,19
	large	0,20	0,20	0,20	0,20	0,22
Exhibition	*small*	0,04	0,08	0,09	0,09	0,09
	medium	0,08	0,09	0,08	0,08	0,08
	large	0,09	0,08	0,08	0,08	0,07
Total Time excluding exhibition time	*small*	**0,30**	**0,51**	**1,76**	**1,30**	**1,33**
	medium	**0,59**	**0,90**	**2,58**	**2,03**	**1,94**
	large	**1,61**	**2,37**	**5,09**	**4,34**	**3,71**

Table 3 –Visible Woman's skull data set times in seconds

Considering the creation of the intermediate image, the most time-consuming sub-step is slice composition, which includes voxel shading, color and opacity interpolation, and accumulation into the intermediate image. The results shown in Table 3 indicate that the algorithms can be ranked as follows, from the most to the least efficient in this processing step: low, high frequency, footprint, and high.

The step for assembling the intermediate image applies only to the multi-resolution algorithms. The footprint algorithm presents better times for this step than the high-frequency algorithm. The former creates only one low-resolution intermediate image that accumulates only volume contributions, while the latter creates two low-resolution images that accumulate volume and volume plus surface contributions, respectively. Thus, the footprint method needs to scale only one low-resolution image. Moreover, it has to assemble less pixels than the high-frequency algorithm. The data in Table 3 supports this analysis.

Finally, concerning the time spent in warping step, one can note that it depends on the intermediate and final image resolutions. In the low-composition algorithm, warping time is equivalent to that of the standard Shear-Warp algorithm. In the algorithms which create a high-resolution intermediate image, warping time increases as the resolution on the intermediate image increases.

When the volume dimensions which are parallel to the projection plane are similar to the final image resolution, the *M* and *N* factors are equal to one. In this case, the low-composition algorithm creates an intermediate image with the same resolution as the one created by high- and multi-resolution algorithms; thus, the final images created by all these methods have the same quality. In this case, it is better to use the faster low-composition algorithm.

Figure 11 shows a set of twenty-four 500^2 images as a 4x6 matrix. The two main sub-diagonals of this matrix, marked with black-background, show the final images created using the proposed algorithms. The upper-right part presents the final images and their differences for the small data set (127x127x51). Similarly, the lower-left part presents images from the large data set (512x512x209). The white-background images were obtained by final image differences in all RGB channels. The central diagonal shows the differences between the final images from small and large data sets with the same algorithm: low, high, footprint and high frequency. The upper-right and the lower-left corners of the matrix show the differences between the methods for the same data set. For example, an image in a matrix position index by "High,Footprint" presents the difference between the images obtained by these two methods.

In order to enhance visualization, the image differences were corrected by a 2.5 gamma factor. After this correction, all color channels were inverted and the final colors

were mapped to grayscale. This yielded the white-background images shown in Figure 11.

It is important to note that, in the viewing direction of the images shown in Figure 11, the resolution in the high intermediate image is always greater than that of the low-resolution intermediate image, that is, M or N is greater than one. This is true even for the large data set, due to the fact that one of the dimensions has size 209. If this were not the case, all image differences in the lower-left part of the matrix would be entirely white, indicating no difference.

From Figure 11, a series of observations can be made. The first is that the differences are proportional to the M and N factors. In the small data set these factors are larger, and the differences are more pronounced than in the large data set.

The second observation is that, differently from the others, the low-resolution algorithm interpolates colors and opacity after the composition step, yielding the differences shown in the end of the first row and in the first column of the image matrix shown in Figure 11.

Thirdly, the high-resolution algorithm differs from the footprint and the high-frequency algorithms in the surface projection and in its border, respectively. This is also clearly shown in Figure 11.

The final observation is that the difference between the footprint and the high-frequency algorithm occurs within the surface projection and in the voxels whose projection is adjacent to it.

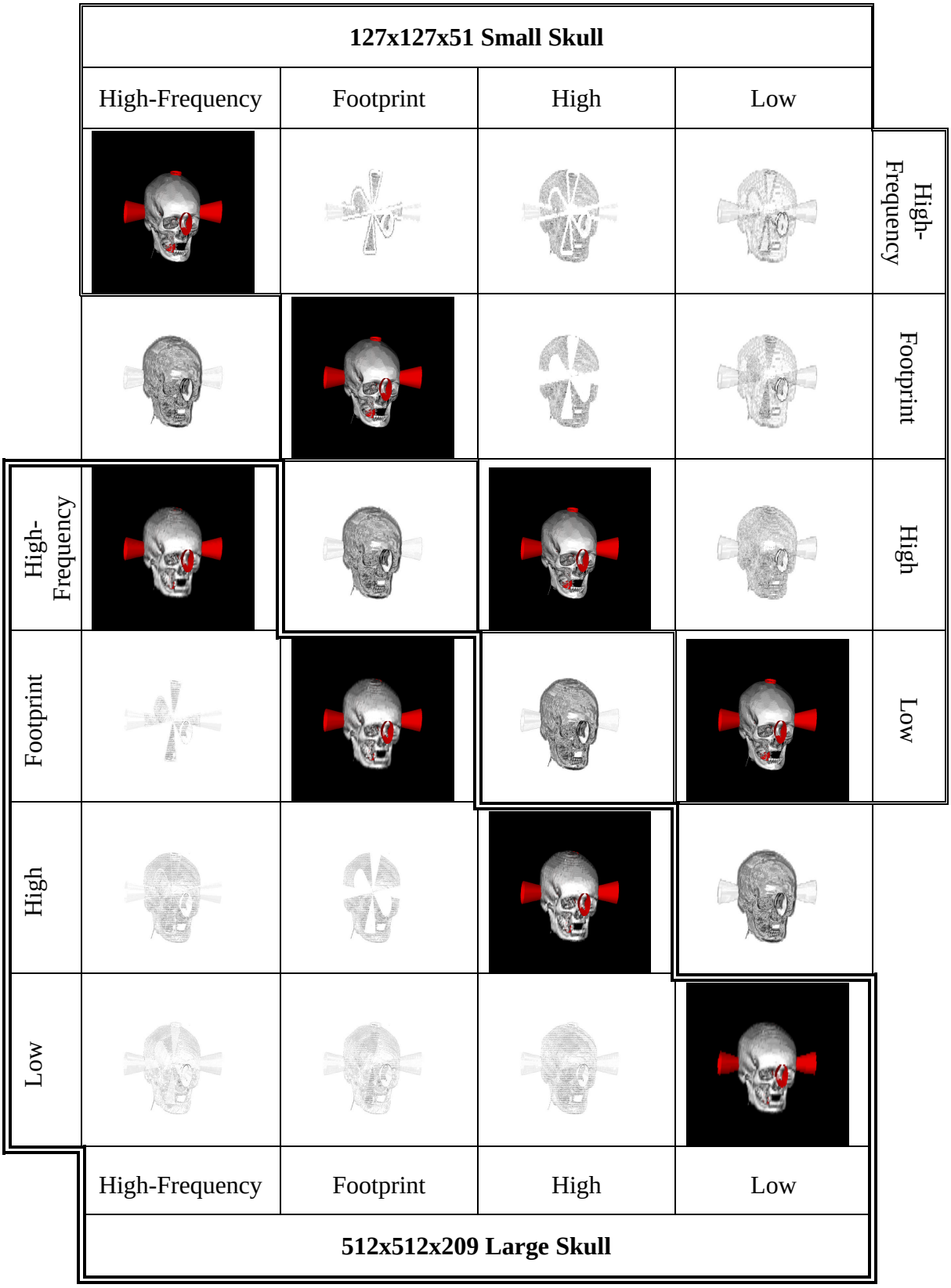


Figure 11 – Images and image differences among the proposed methods

5. Conclusions

This work presented four Shear-Warp algorithms for combining volume data and polygonal surfaces using Z-Buffer: low resolution, high resolution, footprint, and high frequency. These algorithms were implemented, and test results have shown that they correctly mix the volume data and polygonal surfaces into the final image. The difference between them concerns efficiency and final image quality.

The low-resolution algorithm presents the best time for mixing volume and surface data; however, its final image can present some aliasing artifacts on the borders of the surface representation.

The high-resolution algorithm presents the best final image quality, but its time is the highest. This happens despite the use of the RLE encoding of the slices and of the low-resolution intermediate image. Without this optimization, its running time would be totally unacceptable.

The proposed multi-resolution algorithms kept the RLE encoding, generating, in acceptable times, final images of quality comparable to that of the high-resolution algorithm. The examples studied in this research slightly favor the use of the high-frequency algorithm.

Finally, there is much to be gained if these algorithms could be implemented in hardware. In the absence of a specialized hardware, implementing the algorithms presented here can be significantly improved if two general steps were supported by the graphics boards: flexible Z-Buffer and image composition with RLE encoding. By flexible Z-Buffer we mean the ability to define the number representation used in the buffers and to efficiently access these buffers from a program in the main memory. OpenGL provides this access but in most implementations it is a very slow operation. DirectX addresses this problem and presents a partial solution.

The image composition with run-length encoding should take advantage of the opacity of the destination image in an *over* operation such as the ones shown here. Furthermore, it would be very useful if this blending operation could also be controlled by a depth buffer.

6. Acknowledgment

This work was developed in TeCGraf/PUC-Rio, and was partially funded by CNPq, by means of fellowships. TeCGraf is a Laboratory mainly funded by PETROBRAS.

7. References

1. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, New York, 2nd Edn, 92-100 (1990).
2. A. Kaufman, R. Yagel, and D. Cohen, Intermixing Surface and Volume Rendering. In *3D Imaging in Medicine: Algorithms, Systems and Applications*. K. H. Hoehne, H. Fuchs, and S. M. Pizer (Eds.), Springer-Verlag, 217-228 (1990).
3. P. Lacroute, and M. Levoy, Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH'94)*, Orlando, 451-458 (1994).
4. P. Lacroute, Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation, Technical Report: CSL-TR-95-678, Computer Systems Laboratory Departments of Electrical Engineering and Computer Science, Stanford University (1995).
5. P. Lacroute, Analysis of a Parallel Volume Rendering System Based on the Shear-Warp Factorization. *IEEE Transactions on Visualization and Computer Graphics* **2**, 3, 218-231 (1996).
6. M. Levoy, A Hybrid Ray-Tracer for Rendering Polygon and Volume Data. *IEEE Computer Graphics and Applications* **10**, 3, 33-40 (1990).
7. T. McReynolds, D. Blythe, C. Fowle, B. Grantham, S. Hui, and P. Womack, Programming with OpenGL: Advanced Rendering. *SIGGRAPH ´97 Lecture Notes, Course 11*, 144-153 (1997).
8. T. Miyazawa, and K. Koyamada, A High-Speed Integrated Renderer for Interpreting Multiple 3D Volume Data. *The Journal of Visualization and Computer Animation* **3**, 65-83 (1992).
9. R. Osborne, H. Pfister, H. Lauer, N. McKenzie, S. Gibson, W. Hiatt, and T. Ohkami, EM-Cube: An Architecture for Low-Cost Real-Time Volume

Rendering. *Proceedings of the 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 131-138 (1997).

10. H. Pfister, and A. Kaufman, Cube-4 – A Scalable Architecture for Real-Time Volume Rendering. *ACM/IEEE Symposium on Volume Visualization*, 47-54 (1996).
11. D. Tost, A. Puig, and I. Navazo, Visualization of Mixed Scenes Based on Volumes and Surfaces. *Proceedings of Fourth Eurographics Workshop on Rendering*, Paris, 281-292 (1993).
12. T. van Walsum, A. Hin, J. Versloot, and F. H. Post, Efficient Hybrid Rendering of Volume Data and Polygons. In *Advances in Scientific Visualization*, Springer-Verlag, 83-96 (1993).
13. O. Wilson, A.V. Gelder, and J. Wilhelms, Direct Volume Rendering via 3D Textures, Technical Report UCSC-CRL-94-19, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz (1994).
14. C. M. Wittenbrink, T. Malzbender, and M. E. Goss, Opacity-Weighted Color Interpolation for Volume Sampling, HP Labs Technical Report: HPL-97-31R2, Hewlett-Packard Laboratories (1998).
15. Visible Human Project: www.nlm.nih.gov/research/visible/visible_human.htm