Evaluation of a Real-Time Direct Volume Rendering System

M. de Boer, J. Hesser, A. Gröpl, T. Günther, C. Poliwoda, C. Reinhart, R. Männer
Lehrstuhl für Informatik V, Universität Mannheim, D-68131 Mannheim, Germany
guenther@mp-sun1.informatik.uni-mannheim.de

Summary

VIRIM, a real-time direct volume rendering system is evaluated for medical applications. Experiences concerning the hardware architecture are discussed. The issues are the flexibility of VIRIM, the restriction to two gradient components only, the duplication of the volume data sets on different modules, the size of the volume data set, the gray-value segmentation tool, and the support of algorithmic improvements like space-leaping, early ray-termination and others.

It turned out that flexibility is the main benefit and absolutely necessary for VIRIM. Given this flexibility the application areas of real-time rendering systems increase dramatically: Most of the user requirements focus now not on visualization but on general volume data processing. The most serious bottleneck of VIRIM is the limited volume memory that is integrated on the first prototype.

The most frequently used tool of VIRIM is gray-value segmentation. It is highly useful if original, i.e. unsegmented data have to be dealt with, and if pre-segmented data have to be investigated.

All other benefits and architectural shortcomings are not critical for the application areas of VIRIM, i.e. operation simulation and control in head surgery.

Introduction

In many time critical applications like medical operation simulation and control, real-time frame rates are a prerequisite for the acceptance of direct volume rendering by the user. The real-time condition can be met by two different approaches, by optimizing the underlying algorithms and by using special purpose hardware. Both approaches have been successfully applied. Lacroute and Levoy [1] have demonstrated that rendering of a 256^3 data set on a 16-processor SGI Challenge is possible under some restrictions: The incident light is parallel, shear-warp is applied, and a preprocessing step of approximately 1 minute is necessary where e.g. gradients are precalculated and the data set is prepared for rendering (e.g. run-length encoding in three coordinate directions). Higher performance processors and faster cache memory will speed-up these approaches in the next years by reducing the precomputation as well as the rendering time.

Knittel [2] has presented a small PCI card solution that allows to generate frame rates of approximately 5 Hz on 256^3 data sets

by using space-leaping and adaptive supersampling. He uses preprocessing that takes several minutes (30 were mentioned) to code the data set. Distance coding and a lossy fusion of 8 sub-cubic neighborhoods into a 32 bit word are realized in this phase. During rendering each 32 bit word contains the information for resampling one point in the volume. The main memory of the PC serves as volume memory for the rendering system; which keeps the size and the cost of the PCI board low.

The DIV^{2}A system of Lichtermann [3] is a relatively flexible system for direct volume rendering. It uses space-leaping and early ray-termination to speed-up rendering by up to an estimated factor of 20. Three special purpose ASICs have been produced to resample points in the data volume in several cycles per resampling point and to provide the resampled values and the estimated gradients to a digital signal processor for rendering. The volume memory is realized in SRAM and volume data is stored in an interleaved way. However memory usage is only 50% in order to reduce inter-processor communication.

All three systems require appropriate data to achieve the required real-time frame rates. However these approaches fail to give the desired performance for semi-transparent objects. For such cases different systems have to be proposed or realized. One commercial system is the RealityEngine of SGI [4]. With the volume texture hardware it performs the compositing step of the rendering algorithm in real-time (10 Hz for 256^3 data sets). However shading must be precomputed.

Cube-4 [5] that is currently simulated by an FPGA multi-chip module allows to process a full scanline in parallel by using a skewed memory architecture (cubic-memory) and locally connected pipeline processors for rendering. The system promises a maximal rendering rate of 30 Hz for 1024^3 data sets by using 1024 memory banks and the same number of processors each working at 30 MHz.

Vogue [6] is a realization that uses 4 dedicated ASICs into which the full rendering algorithm is mapped. It has been successfully simulated and it is assumed to be integratable as a "pizza-box" solution.

In contrast to other systems that have been suggested, simulated, or emulated only, VIRIM is a fully operational prototype for real-time direct volume rendering which has been in use since June 1995 [7]. It is designed for maximal flexibility at moderate cost. It will be described below in more detail. Both architecture and implemented algorithms are shortly mentioned. Operation simulation and control, where VIRIM has been evaluated first, has particularly difficult real-time demands. Experiences made during the evaluation phase are discussed below. The outlook describes the improvements planned for redesign of the hardware.

VIRIM Architecture

VIRIM uses image space parallelism for rendering (see Fig. 1). First the object data set is resampled into an image data set where one coordinate direction (y) coincides with the main viewing direction. The x direction is parallel to the scanlines. Two light sources are used where the light rays are parallel to the x-y plane of the image data set. One light source lies in the direction of the viewer, the other 45° apart from the first one [7].

After this geometry operation rendering is performed on the x-y-slices of the image data set.

The architecture of the VIRIM system corresponds to this approach. It consists of two components (see Fig. 2), a geometry unit and a ray-cast unit.

The geometry unit is used for resampling, perspective calculation, and gradient estimation. Since it requires full access to the volume memory it has been designed for maximum speed in

order to reduce the number of geometry units to a minimum that allows to mitigate object data set distribution problems on several volume memories. The geometry unit is integrated into a parallel pipeline processor that generates each clock cycle (20-40 MHz) one sample point and its two gradient components.

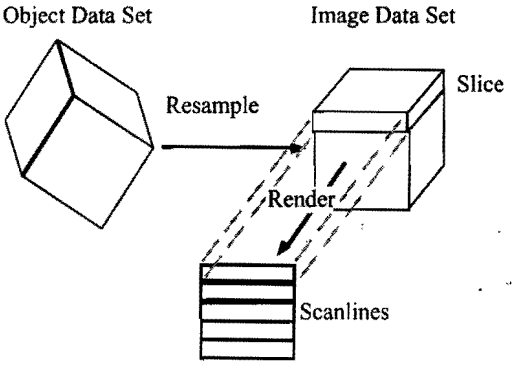

Fig. 1: Sketch of the image parallel approach of VIRIM. First the object data set is resampled into an image data set. Each slice of the image data set supplies the information for rendering one scanline of the projection.

The ray-cast unit performs the remaining rendering operations in a programmable way on a multiprocessor consisting of digital signal processors (DSPs). Since only x and y gradient components are used for shading, each scanline can be processed independently from others and no communication is necessary between the DSPs.

Geometry unit and ray-cast unit are connected by a 240 MBytes/s fast bus that transfers sample point density and its two gradient components. Each geometry unit supplies data for 8 to 128 DSPs. At a 20 MHz data rate 16 DSPs are required per geometry unit to harness its full speed.

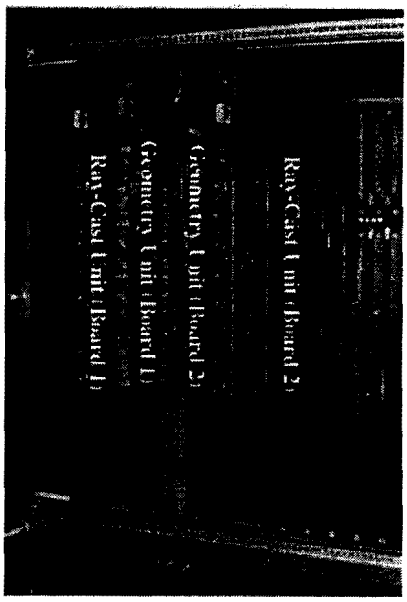

Fig. 2: Picture of VIRIM. The two outer boards are the ray-cast unit, the two inner boards the geometry unit. All four boards as well as a host workstation are connected by a VME bus. The geometry-ray-cast bus is integrated in the customized VME backplane.

Implemented Rendering Algorithms

VIRIM is designed for maximal flexibility in real-time volume rendering. Currently four different algorithms are implemented, maximum intensity projection, ray-casting, and two volume ray-tracing algorithms.

Maximum intensity projection (MIP) [10] is a commonly used projection method to display volume information. The projection is generated by determining the maximum gray-value along each ray cast from an image pixel into the data set. The main application area in medicine is angiography where the only objects that give contrast are blood vessels; which are displayed and which can be viewed from any direction.

One standard rendering algorithm is ray-casting. From each image pixel a ray enters the virtual scene. At equally spaced positions on the rays sample points are interpolated from their 8 neighboring voxels in the object data set by e.g. trilinear interpolation. Gradients are estimated and shading is performed on each sample point. Gradients are determined by local difference filters. Shading is calculated according to reflectance models like Phong shading where it is assumed that each sample point obtains a constant intensity of light from each light source. Finally, the contributions of the rays' sample points are composited into the final projection using the *over* operator [11].

Our implementation on VIRIM differs from that of Levoy [8] by two modifications. First, only two gradient components, one parallel to the scanline and one parallel to the main viewing direction, are calculated. Second, the classification step for assigning opacity to interpolated sample points is reduced to gray-values only instead of gray-values and gradient magnitude.

The third class of algorithms are volume ray-tracing algorithms. In contrast to ray-casting volume ray-tracing takes into account the absorption of incident light during its way through the data volume to the sample point. Two light sources are used in order to avoid totally black areas. Volume ray-tracing thus allows to generate shadows that are helpful in some applications [12].

We have implemented two different algorithms of this class, the Heidelberg Raytracer and the VIRIM ray-tracer. The Heidelberg Raytracer realizes Phong shading and compositing in an unusual way. Unnormalized gradient components are used for Phong shading. All reflection coefficients are multiplied by gradient magnitudes. Additionally, the ambient component is neglected in favor of a component that emits light proportionally to the local density. Also an unusual compositing operation is applied. The standard compositing multiplies the light from backward by the transparency of the voxel, the reflected light by the opacity (=1–transparency), and adds up both contributions. In the Heidelberg Raytracer the reflected component is not weighted and therefore the contrast in the image is lower than that for ray-casters.

The VIRIM ray-tracer in contrast uses standard Phong shading with normalized gradient components and standard compositing; which gives sharp images but is prone to artifacts. Most of the artifacts vanish if supersampling is used.

During our evaluation phase we experienced with all four implemented algorithms and came to the following conclusions:

Maximum intensity projection is nearly always unsuited for representing volume information. The standard ray-casting approach turns out to be very useful for stereo projection, i.e., if two images are calculated with a 4° inclination and a shift due to the eye distance —each such image is presented to one of the viewers' eyes. Since ray-casting does not produce shadows the viewer can look into caverns and perceives their depth. However it seems as if these caverns are illuminated from inside. Algorithms that produce shadows obscure such caverns and the 3D-effect is restricted to surfaces only.

If however the normal (non-stereo) display mode is used, caverns are difficult to detect with the standard ray-casting approach since the perception of depth is insufficient. Shadows that are generated by the volume ray-tracing algorithms help the viewer to detect them and to better perceive their form (see Fig. 3).

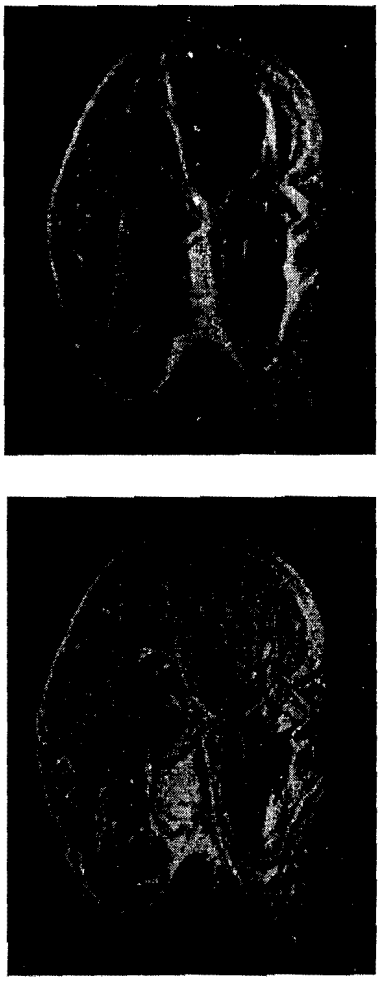

Fig. 3: Visualization of a human heart cut into halves. Top: Rendered with volume ray-tracing; Bottom: Rendered with ray-casting. As can be seen the caverns are more clearly visible due to the shadowing generated with volume ray-tracing.

Application Area

VIRIMs first application area is the support for operation simulation and control in minimal invasive head surgery. During both operation planning and control the interface to the user is realized by a tracking system. The tracking system consists of a magnetic source and several sensors in a stick-like instrument (or the real endoscope); which allow to determine its position and orientation with a spatial resolution of a few millimeters. The stick thus represents an input device with 6 degrees of freedom that is used to steer the visualization process. This instrument is either used like an endoscope whose position and orientation in the virtual head determines which view is computed. Alternatively a fixed view is chosen in which the movement of instruments like endoscopes, scalpels, forceps, etc. is displayed.

During operation simulation the problem arises how to access the operation area with minimal risk to injure blood vessels, nerves, eye or brain. A pre-operative segmentation assigns to each voxel of the data set its respective object like tissue, blood vessel etc. These data are loaded on VIRIM and the surgeon can begin the operation planning task. The surgeon uses the input device as if he/she operates with the real endoscope in the real patient. Tasks like finding the optimal access path and the lesion-volume that is to be removed can be planned.

In the operation control phase the real endoscope replaces the former input-stick to be tracked. Its actual position is compared in real-time with the pre-planned path. Two images are presented to the surgeon: one created by the real endoscope (optically or by camera), and one computed from the patient's data by the visualization system. Both images show always the same view although the computed image can be manipulated by an appropriate choice of the visualization parameters. A semi-transparent view, e.g., allows to see normally invisible structures like blood vessels or nerves hidden below the visible surface. Moreover the actual instrument-position can be checked continuously against the preplanned position and any deviation can create warnings. Should it be necessary to alter the access path during operation the modified operation can be simulated on-line.

In May 1996 VIRIM has been successfully installed in the Clinic for Head Surgery at the University of Heidelberg.

Experiences and Discussion

This section describes the experiences that have been made during the first year of use of the prototype. Several critical points have been identified.

- **Volume memory:** One of the most serious obstacles for using VIRIM is the limitation of the volume memory size. Modern imaging devices like CTs generate slices with a 512×512 pixel resolution; industrial CTs generate even larger slices. Another source of large data sets is 4D visualization where 3D data sets are imaged at different times. VIRIM allows to visualize 4D data by switching between the 3D cubes and thus generates images of moving 3D data sets.

  As consequence the 256^3 volume memory size is too small and will be changed for the next version.

- **Data set duplication:** Another critical point seems to be data set duplication which would be required if multiple modules were used. However currently the rendering speed of one module is fast enough for most users. A full-scale system with four or more modules is thus not expected to be built in the next future.

- **Gray-value segmentation:** The possibility to manipulate gray-values during visualization turned out to be an extraordinary valuable tool. It is permanently used and its functionality is steadily increased taking into account practical experiences with VIRIM. It works as follows: In pre-segmented data, e.g., different objects in the volume are marked by a different code that forms, together with the gray-value of the voxel, a 16 bit word (see Fig. 4). A 64k×16 look-up-table in the geometry unit allows to transform the original gray-values of the object data set into opacity values. In this look-up-table each object is assigned a segment of size 2^n that defines the opacity mapping for all of its voxels in the object data set.

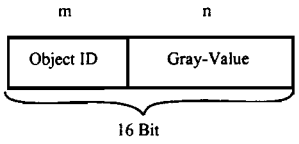

Fig. 4: A 16 bit voxel word consisting of object identification bits and gray-value.

By individually manipulating these segments the opacity of all objects can be manipulated without interference with other objects. An example is the MRI data set of a human head shown in fig. 5. After segmentation of eye, brain, skull, and skull ceiling, each of these objects is assigned a 2 bit code (in this case m = 2). The user interface allows to call these objects by name which is coded in the data set format. During visualization the skull ceiling, e.g., can be set to semi-transparent by pressing the corresponding button and by changing its transparency (see Fig. 5) interactively.

A newer application is operation simulation. Here an instrument is immersed into the data volume. The instrument is generated by the host computer by writing its shape directly into the volume data. The movements of the instrument in the volume data can be used for removing the corresponding virtual tissue. This is achieved by marking all voxels that are touched by the instrument with a bit code and by using the gray-value segmentation tool to set all such marked voxels to transparent (see Fig. 6).

This tool is used to simulate an access and can be used to quantitatively measure which amount of tissue to remove during operation.

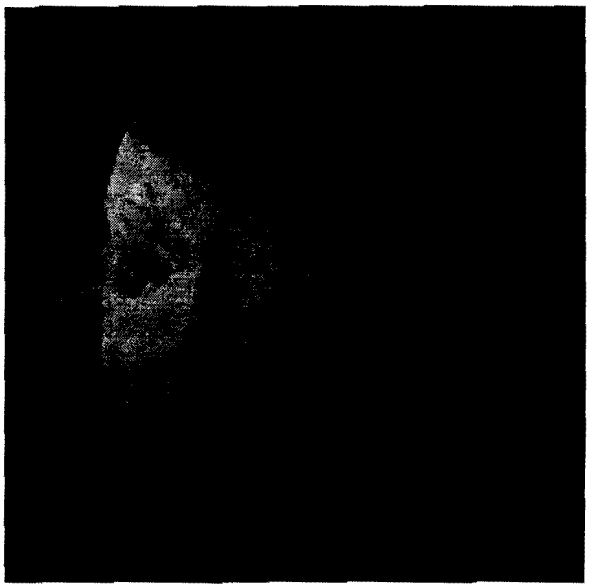

Fig. 5: An MRI head is rendered. The skull ceiling is segmented from the remaining head. It is rendered semi-transparently. Below the semi-transparent ceiling the brain is visible.

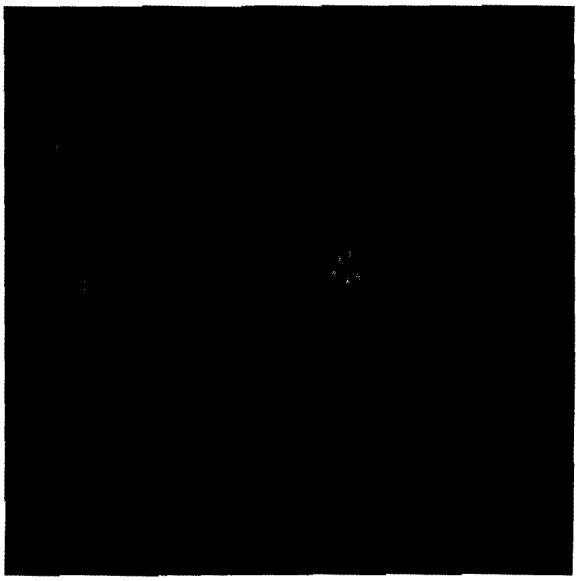

Fig. 6: A typical image generated during operation simulation. A three-dimensional cross represents the surgical instrument that removes the tissue locally. Below the opening in the head the brain is visible.

- **Look-up tables:** One disadvantage of the VIRIM hardware is the lack of a look-up table (LUT) after resampling or interpolation of data set voxels. We have observed that setting the opacity of objects with the gray-value segmentation LUT alone leads to some blurring of the objects during visualization (see also [13]). However the decision to use a look-up table before interpolation allows additionally to handle pre-segmented data which would not be possible after interpolation. We concluded that a better system should support both approaches.

- **Gradients:** More detailed simulations that revealed the sources of artifacts in the rendering algorithms showed that the Sobel operator for gradient estimation can be replaced by a simpler difference filter. The lack of the z component (perpendicular to the scanline and the main viewing direction; the final projection is given in x-z coordinates) is not a major problem. A typical example where the difference is seen most obviously is given in the two images of Fig. 7.

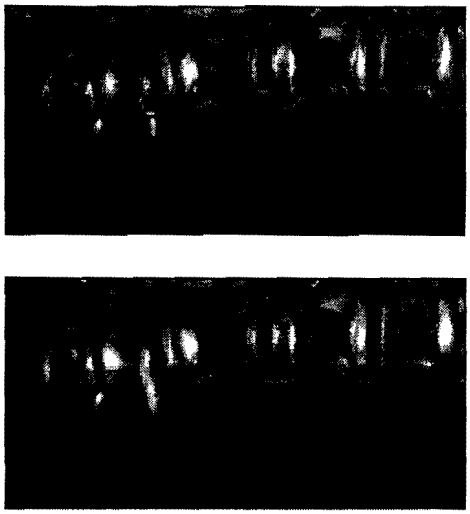

Fig. 7: Top: Rendered image without z gradient. Bottom: Rendered image with z gradient.

- **Flexibility of VIRIM:** One of the major design goals for the VIRIM system was to keep it maximally flexible. A

priori it was not clear which would be the most appropriate rendering algorithm for the anticipated application areas. The flexibility is based on a flexible resampling scheme, on programmable interpolation filters, and on digital signal processors (DSPs) that perform shading and compositing.

The resampling scheme allows in a programmable way to resample many individual slices of arbitrary size and orientation from the volume memory, and to transfer the resampled slices to multiple DSPs. In other words, the VIRIM architecture supports slice-based operations on volume data very efficiently. These operations can be used for different purposes, e.g. for resampling individual image slices that are directly presented on the computer screen or for image processing algorithms implemented on DSPs.

The interpolation filters allow to resample the data set voxels in a non-linear way instead of trilinar interpolation. It turned out that this is not as important as initially supposed. Throughout the data sets investigated their contribution to image quality is negligible. The overhead of using non-linear interpolation is not justifiable to our experience.

Most important however is the free programmability of the DSPs. This flexibility allowed us to implement the four different rendering algorithms that have been described above. Since the system offers a high data rate between processors and volume memory as well as a high processor performance, VIRIM allows to execute arbitrary image processing algorithms in addition to visualization. Currently a region-growing algorithm is being implemented for segmenting medical data sets.

Our experiences show that flexibility is most important for real-time rendering systems which are used for interactive work with the data set.

- **Algorithmic improvement techniques:** In contrast to DIV^{2}A, space-leaping, early ray-termination, and adaptive supersampling have not been implemented for VIRIM so far.

  Adaptive supersampling has a great potential to reduce the amount of computations at the sacrifice of loosing small details that can be critical in medicine. Instead we used a more efficient method, dynamic resolution, that is possible when operating with volume data at interactive rates. Dynamic resolution reduces the resolution of the image data set by a factor of 2 while viewing parameters are changed thus saving a factor of 8 in rendering time. Whenever the user keeps the parameters for one image generation period the image is generated at full resolution.

  This feature is used nearly always since it allows to achieve full interaction speed with one module only.

  Space-leaping and early ray-termination are promising in about 50-70% of all cases (reduction of computational time by a factor 5-20); in all other cases the user displays the objects semi-transparently where the performance gain is estimated to 50%. It seems therefore necessary to incorporate these methods in the next generation of renderers we are currently investigating.

  Space-leaping and early ray-termination change the resampling sequence. Space-leaping omits those voxels that are empty. The information about empty spaces in the volume

have to be known before accessing the memory by using a preprocesing step. During rendering this information has to be read to generate the required resampling positions — this is not supported by VIRIM.

Early ray-termination omits all subsequent sample point for the considered ray when the light intensity falls below a threshold. This decision can be used only in the ray-cast unit in order to stop the geometry unit and change the resampling sequence. Stopping and restarting the geometry unit however would consume too much time to use this improvement technique efficiently.

In order to mitigate the problem for its implementation on VIRIM the slice wise processing of the final projection has to be replaced by a sub-cube based approach. In this approach each signal processor of the rendering unit renders non-intersecting sub-cubes of the object data set. The partial images of each sub-cubes is then composited to the final projection. A typical size of such non-intersecting sub-cubes is 8^3.

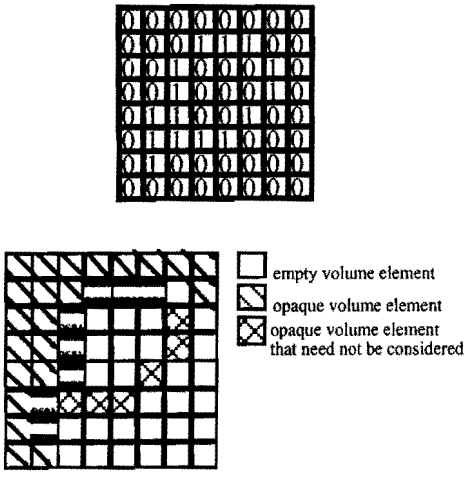

Fig. 8: Left: The data structure to describe empty and non-empty sub-cubes is a three-dimensional binary array. 0 indicates empty, 1 indicates non-empty sub-cubes. Only a slice of that array is shown. Right: Light rays illuminate only those sub-cubes that are not empty. The dark squares are the sub-cubes that are illuminated first and assigned to signal processors accordingly. The hatched squares are the sub-cubes that are not illuminated since the light intensity cast onto these sub-cubes is below a user-defined threshold.

In a first phase empty sub-cubes must be distinguished from non-empty sub-cubes. For this purpose the maximum intensity algorithm is used; which returns the largest opacity in the respective sub-cube. The required time for 256^3 data sets is 0.8 s for one module. These results are stored on the host system as a list-data structure of non-empty sub-cubes (see Fig. 8).

In the second phase, the host assigns each signal processor sub-cubes to render. The assignment of sub-cubes is in front-to-back order, i.e., the non-empty sub-cubes nearest to the viewer are processed first.

The partial image, that is obtained by rendering one sub-cube, consists of the pixel brightness and the corresponding ray intensity. It is stored in the volume memory.

Before rendering the next layer of sub-cubes by the signal processors the processors have to check whether the sub-cube can contribute to the final image, i.e., whether the calculated ray intensities hitting the sub-cube exceed a user-defined threshold. Therefore each such processor first reads the required intensity and the pixel brightness from the volume memory. The processor integrates the intensity and compares it with the user-given threshold. If the threshold is exceeded the processor renders the sub-cube and composites its partial result with the pixel brightness and updates the intensity. Both, updated intensity and pixel brightness, are again stored in the volume memory.

In the other case it directly requests the host for the next sub-cube to be rendered.

A simulation shows that for typical data sets with hard surfaces and many empty spaces (e.g. heart data set or visualization of bone of a skull) this approach of space-leaping

and early ray-termination reduces the number of rendered sub-cubes by approximately a factor of 5.

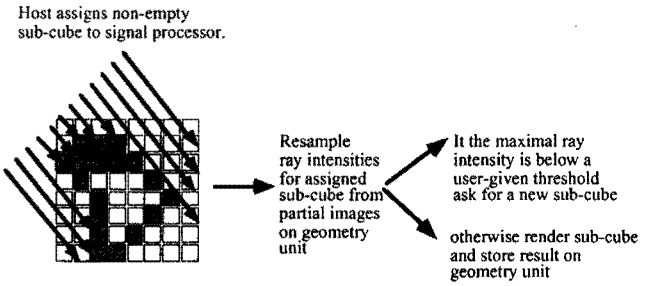

Fig. 9: Implementation of early ray-termination.

Nevertheless the efficiency of this approach is not as good as pure software solutions (see [1]) although the preprocessing time is in the range of a second for one module; which is tolerable compared to minutes for other approaches.

These experiences, made during the first year of operation of VIRIM, may be biased due to the specific application where the users work with the data instead of only visualizing them. Due to our approach the user is tempted to use different techniques for changing the transparency of the data set, and to move freely within it.

Outlook

VIRIM is currently being redesigned for commercialization. The new system will have a larger volume memory of 128 MB and a look-up table after the interpolation step of the geometry unit.

Acknowledgments

This work is supported by the Ministry of Education and Research, Germany under grant 01 IR 406 A8 and by the Landesforschungsschwerpunktprogramm of Baden-Württemberg under grant 7532.24-2-16.

References

[1] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform. Computer Graphics, Proc. of SIGGRAPH '94, Orlando, FL, 1994, pp. 451-457.

[2] G. Knittel. A PCI-based Volume Rendering Accelerator. W. Straßer, 10th Eurographics Workshop on Graphics Hardware, Maastricht, The Netherlands, 1995, pp. 73-82.

[3] J. Lichtermann. Design of a Fast Voxel Processor for Parallel Volume Visualization. W. Straßer, 10th Eurographics Workshop on Graphics Hardware, Maastricht, The Netherlands, 1995, pp. 83-92.

[4] R. Fraser. Interactive Volume Rendering using Advanced Graphics Architectures. SGI Developer News, Dec., 1994, pp. 5-9.

[5] H.-P. Pfister, A. Kaufman, F. Wessels. Towards a Scalable Architecture for Real-Time Volume Rendering. 10th Eurographics Workshop on Graphics Hardware, Maastricht, The Netherlands, 1995, pp. 123-130.

[6] J. Hesser, R. Männer, G. Knittel, W. Straßer, H. Pfister, A. Kaufman. Three Special-purpose Architectures for Real-Time Volume Rendering. Eurographics '95, Maastricht, The Netherlands, 1995, pp. C-111—C-122.

[7] T. Günther, C. Poliwoda, C. Reinhart, J. Hesser, R. Männer, H.-P. Meinzer, H.-J. Baur. VIRIM: A Massively Parallel Processor for Real-Time Volume Visualization in Medicine. W. Straßer, 9th Eurographics Workshop on Graphics Hardware, Oslo, Norway, 1994, pp. 103-108.

[8] M. Levoy. Display of Surfaces from Volume Data. IEEE CG&A, Vol. 8, No. 5, 1988, pp. 29-37.

[9] H.-P. Meinzer, K- Meetz, D. Scheppelmann, U. Engelmann. The Heidelberg Ray Tracing Model. IEEE CG&A, Nov. 1991.

[10] K.-H. Höhne, M. Bomans, A. Pommert, M. Riemer, C. Schiers, U. Tiede, G. Wiebecke. 3D Visualization of Tomographic Volume Data using the Generalized Voxel Model. *The Visual Computer*, **6**, pp. 28-36.

[11] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, Reading, MA, 2d. ed., 1990.

[12] H.J. Wieringa. MEG, EEG and the Integration with Magnetic Resonance Images. Ph.D. thesis, Univ. Twente, The Netherlands, 1993.

[13] M. Bosma, J. Smit, J. Terwisscha van Scheltinga. Super ResolutionVolume Rendering Hardware. 10th Eurographics Workshop on Graphics Hardware, Maastricht, The Netherlands, 1995, pp. 117-122.