VIRIM: A Massively Parallel Processor for Real-Time Volume Visualization in Medicine

T. Günther†, C. Poliwoda†, C. Reinhart†, J. Hesser†, R. Männer†¶, H.-P. Meinzer*, H.-J. Baur*

† Lehrstuhl für Informatik V, Universität Mannheim, Mannheim, Germany

¶ Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Univ. Heidelberg, Heidelberg, Germany

* Abt. Med. u. Biolog. Informatik, DKFZ Heidelberg, Heidelberg, Germany

Abstract

Architecture and applications of a massively parallel processor are described. Volumes of 256×256×128 voxels can be visualized at a frame rate of 10 Hz using volume oriented visualization algorithms. A prototype of the scalable and modular system is currently set up. 3D rotation around an arbitrary rotation axis, perspective, zooming, and arbitrary gray value mapping are provided in real-time. Multi-user access over high-speed networks is possible.

A volume oriented visualization algorithm is used that is tailored to the requirements in medicine [5]. With this algorithm, small structures of a size down to the pixel resolution, and structures without defined surfaces can be visualized as well as semi-transparent objects. One application of the system is therapy planning in heart surgery.

Introduction

Data drawn from computer tomography (CT) or nuclear magnetic resonance (MR) allow in principle to look inside the patient's body using visualization of these data. Today's imaging machines have a resolution on the mm scale so that many details important for diagnosis and therapy can be recorded. However limitations of the visualization process have up to now narrowed the application area of 3D visualization of such data. Typical visualization problems to be solved in the medical context are e.g. tissues that have no defined surface (e.g. ramify into its surrounding healthy tissue) or blood vessels whose size is close to the limit of imaging resolution. Also, the visualization of semi-transparent objects, where objects can be visualized within their anatomical context, has shown to give decisive information for successful diagnosis or therapy.

Volume visualization is currently done using two different approaches: surface oriented and volume oriented algorithms.

Surface oriented methods are standard for nearly all 3D visualization problems. The surfaces of the relevant objects are extracted from the data according to some criteria, and—for visualization—incoming light interacts with these surfaces only [1,2,3]. With dedicated vector hardware, these models can be calculated very efficiently. However, surface oriented methods leave the typical problems in medical visualization of the kind described above unsolved.

In contrast, in volume oriented methods [4,5] each voxel is illuminated and contributes to the final 2D projection. Although they do not suffer from the problems of surface oriented methods, they have not yet found widespread application in medicine. The reason why they were hindered from introduction into clinical practice is their immense computational demand which is about three orders of magnitude higher than with surface oriented methods. The calculation of one 2D projection with the Heidelberg Raytracing Model [5] e.g. currently takes about 5 minutes for data volumes of 256×256×128 voxels using a standard workstation like a SUN Sparc2.

The aspect of computational demand becomes decisive when real-time visualization is required. Real-time calculation of arbitrary views not only enables the physician to capture the real 3D shape of the objects under study but also allows interactive optimization of visualization parameters.

Despite the demand for real-time volume visualization in medical applications, only recently have parallel architectures been suggested for real-time visualization using volume oriented methods, and few prototypes are currently under development [6,7,8,9].

While real-time visualization hardware using surface oriented visualization algorithms is state-of-the-art (Reality Engine (SGI), HP, SUN, IBM, PIXAR) [10,11,12], these systems are far from providing real-time visualization when running volume based algorithms. One reason is the sheer computational demand of ≈5-10 GFlops, the other reason is the data communication problem in parallel systems, when views of the data volume from different angles are desired.

Up to now no commercial system is available for real-time visualization with volume oriented algorithms. The urgency to support the physician in his interpretation of 3D data from CT or MR lead us to the design of a massively parallel machine dedicated to this task, which will be embedded into a clinic information system to provide full use of other information techniques.

Below the Heidelberg Raytracing Algorithm is described, which is the underlying volume visualization approach. This is followed by a detailed description of the architecture of the parallel computer system and finally, the application areas of the system are described.

Algorithm

As our basic visualization algorithm, we use the Heidelberg Raytracing Model of Meinzer *et al.* [5]. It has been modified so that rotation around an arbitrary axis, perspective, scaling of original gray values, background identification, stereo view, and other features became possible.

The Heidelberg Raytracing Model is a volume oriented algorithm which uses ray tracing and Phong shading (with x and y gradient only) for the generation of projections of the 3D data cube. The algorithm operates with 2 light sources that emit parallel light. One light source lies in the direction of the viewer, one 45° apart.

Before raytracing, the data cube is rotated and the volume is resampled such that the viewer looks along the y axis into the rotated data cube. After the rotation, the bundle of light rays from both sources enters the volume parallel to the x-z plane (see Fig. 1). Then, it is possible to calculate the interaction between light and optical density slice by slice (reflected light does not enter into other planes), and the amount of light which is emitted into the viewer's direction (y) gives the resulting image.

For perspective view, x-z planes, which are perpendicular to the viewing direction, are resized after rotation. The planes that are next to the viewer are expanded, while those that are apart from the user are shrinked (s. Fig. 2).

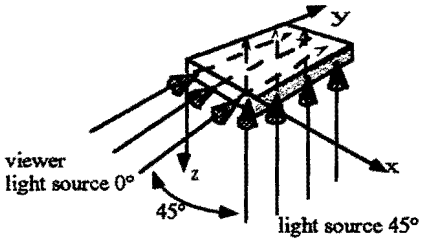

Fig. 1: Schematic of lighting sources, viewer and rotated data cube.

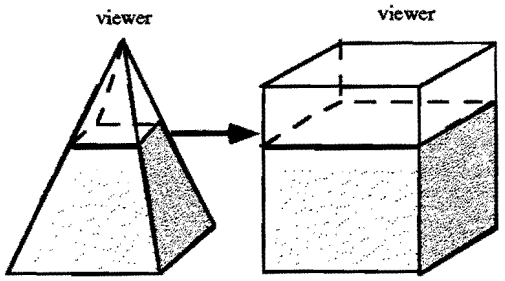

Fig. 2: Schematic of the transformation for the projective view. A pyramid out of the rotated data cube is mapped onto a cube.

In order to allow arbitrary changes of the transparency of objects, the values of the voxels in the data cube can be mapped by a look-up-table (LUT) before rotation and raytracing. We have therefore the following sequence of operations before the raytracing begins:

```
apply LUT to data cube ->
rotate data cube ->
affine transform of data cube for perspective
```

During raytracing, each light ray is first attenuated according to the density of the voxel passed through. The energy deposited in the voxel due to absorption determines the maximal amount of light that is scattered in the viewing direction; no scattering components to other voxels are considered (low albedo).

Light from the viewing direction is scattered only by diffuse scattering while light from 45° is scattered by diffuse and specular scattering. Self-luminosity proportional to the light intensity is the fourth component of the scattered light. A Phong shading model [13] is used to calculate the first three components whereby the required x and y gradient is realized by a x and y Sobel operator [13]. The norm of the gradient vector is taken as measure for the probability for a surface. It is multiplied with the specular scatter component.
To this basic algorithm a background identification step has been added that allows the separation of the background from dark fields in the image. When the light intensity leaving the most distant depth coordinate in an x-z plane is above a certain threshold, a marker is set which indicates that the viewer can see the background behind the data cube.
Finally, stereo view of the data is provided by producing two images with a 4° inclination angle.

Visualization System

In this section, the software and hardware concept of our visualization system are described. Rotation has been identified to consume as much as 80% of the computation time for visualization. Being a fixed operation for which the best possible algorithm is already known, the rotation algorithm is mapped directly into hardware (rotator). Raytracing—which should be modifiable—is performed by an array of digital signal processors (DSPs).
One rotator and 16 DSPs constitute one module; linear scaling of the system's performance is achieved by addition of further modules. Four modules are required for real-time visualization of volume data of 256×256×128 voxels.
The modules are attached to a VME bus which is connected to a host system. The host system provides the base for the control software and the graphical user-interface. Alternatively, the host system can operate as a server allowing multi-user access over high-speed networks.
Our real-time system with four modules will be integrated into one crate. At peak speed, 160 million voxels/s can be rotated and a floating point performance of 2.5 GFlops is achieved.
Below the three components of the system, the rotator, the raytracing processor, and the software concept are described in detail.

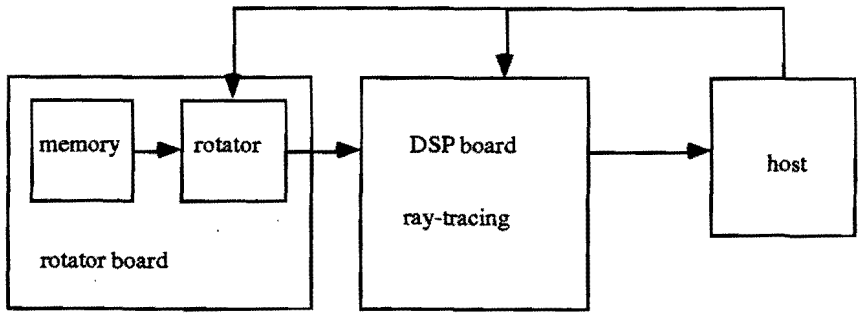

Fig. 4: Schematic of the hardware.

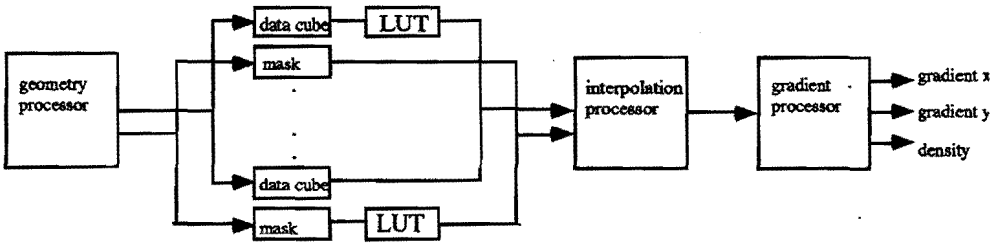

Fig. 5: Schematic of the rotator board.

Rotator Board

In principle, the rotation and the volume sampling of the data cube can be realized in two ways. Either only light sources and viewer are rotated, or the data cube is rotated while light source and the viewer are fixed [7]. The first choice has the advantage that the position of the light sources can be chosen freely without affecting the performance. However, for each ray the fraction of energy deposited in each voxel has to be computed by interpolation. For two light sources this is twice the number of time critical interpolations compared to our choice where only the data cube is rotated[1].

In principle, the rotation and the volume sampling of the data cube can be realized in two ways. Either only light sources and viewer are rotated, or the data cube is rotated while light source and the viewer are fixed [7]. The first choice has the advantage that the position of the light sources can be chosen freely without affecting the performance. However, for each ray the fraction of energy deposited in each voxel has to be computed by interpolation. For two light sources this is twice the number of time critical interpolations compared to our choice where only the data cube is rotated[2].

For the parallelization of the rotation operation, two different approaches are reported in literature. Either the data are distributed over many memories which are attached to processor elements over interconnection structures [8], or a dedicated hardware rotator is used [9,6]. The first approach has the advantage that no data duplication is necessary, but it suffers from the problem of the interconnection network that states a bottleneck due to high latency times. Moreover, communication problems will occur when projection and zooming are allowed. As the rotation is a fixed operation, a direct mapping of the algorithm into hardware has the advantage that this operation can be executed very fast, and only a small number of such rotators is necessary for real-time visualization.

For our system, we chose the latter solution of a hardware rotator. Different approaches for the rotation algorithm have been investigated, rotation using distortion matrices [6], trilinear interpolation [5], and Gaussian interpolation masks [15]. For the last method, the gray value of the voxel v in the rotated coordinate system is interpolated by the gray values of its 8 neighbors v_r' in the original coordinate system. Here, a 64×64×64 mask w with values $w_{i\,j\,k} \sim \exp(i^2 + j^2 + k^2)$, (i, j, k = 0,...,63) is used for the interpolation weights of the v_r', where i/64, j/64, k/64 indicate the x, y, and z distance between v and the corresponding v_r' [3]. The discretization error is invisible if a discretization with 6 or more bits for each coordinate is used.

Distortion matrices smooth the data significantly causing information loss, which is not the case for the other two methods. Compared to trilinear interpolation, the least artifacts (Moiré patterns) are obtained by mask interpolation using a Gaussian filter mask, which we use for our system.

In the following the realization of the rotator board as a parallel pipeline processor is described. The board consists of three parallel pipeline processors, a geometry processor, an interpolation processor, and a gradient processor.

The geometry processor calculates the addresses of the voxels to be interpolated using the rotation matrix, the matrix for the perspective, and the zooming factor. The addressees are fed into the eight-fold divided data RAM containing the data cube. Each RAM unit contains only 1/8th of the whole data such that each eight-neighborhood can be accessed in parallel[4]. The eight values from the RAM are mapped onto density values using a freely modifiable LUT. Besides, the geometry processor additionally generates eight interpolation weight addresses for weight memories. Using eight density values and eight weights the interpolation processor interpolates the density values and produces the density of one rotated voxel.

Finally, the gradient processor applies the Sobel operator for x and y direction, which serve as the x and y gradient for Phong shading.

All three pipeline processors are controlled by a local bus master that is additionally responsible for the communication with the host system e.g. to read parameters for rotation, perspective, and zoom.

The data produced by the rotation board—density value plus x and y gradient—is transferred over a 48 bit wide unidirectional rotator-DSP bus with a peak transfer rate of 240 MB/s. 16 DSPs are attached to this bus whereby only the selected DSP receives the data.

DSP Board

The DSP board is a multiprocessor board that executes the raytracing part of our algorithm. Its flexibility allows the implementation of different visualization models, e.g., the Heidelberg Raytracing Algorithm [5] or the volume visualization algorithm of Levoy [4].

One board consists of 8 digital signal processors (DSPs) and a local bus master CPU that handles all communication, i.e., the distribution of visualization parameters among the DSPs, and the transfer of parts of the final 2D image calculated by the DSPs. That way the full performance of the DSPs can be dedicated to the raytracing itself.

Data from the rotator board arrives over the rotator-DSP bus and are temporarily stored in a FIFO. Using the internal parallelity of the DSP (two floating point operations together with two move operations) the ray tracing and scattering of the light is calculated. Additional devices perform data format mapping (16 bit integer to floating point) and the branch operations, which are unsuited for DSPs but nevertheless substantial to our algorithm. Running at a speed of 40 MHz, each DSP with its additional devices calculates one line (256 pixel) of the final 2D image within 50 ms (≈66 million operations/s).

---

1 In our case we use 0° and 45° light sources. Here, the ray passes through the center of each voxel on its way, and therefore no partitioning between neighboring voxels is necessary.

2 In our case we use 0° and 45° light sources. Here, the ray passes through the center of each voxel on its way, and therefore no partitioning between neighboring voxels is necessary.

3 Cases where the distance = 1 for one coordinate are handled separately.

4 Each RAM unit contains one combination of even/odd data cube addressed voxels. To allow fast r/w accesses, each unit is subdivided into two banks that each contains half the data. This way a ping-pong readout is possible. Accesses on the same address are suppressed and the old data is held on the line; addresses that lie out of the data cube range (determined by software) produce a 0 as data value. Using these additional features, the mean access time can be reduced by more than a factor of two.

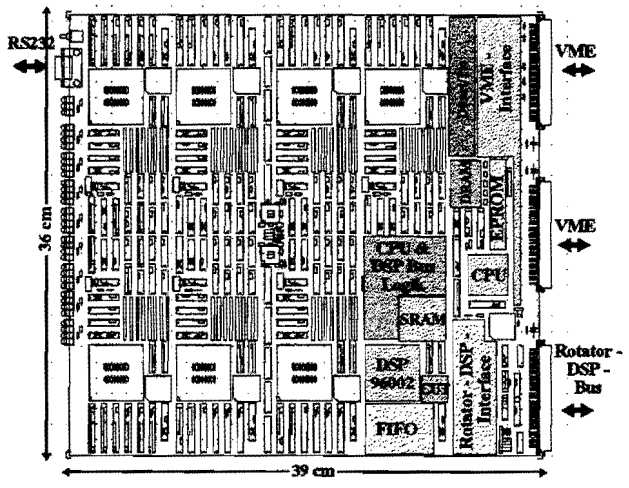

Fig. 6: The DSP board

Communication structure and User Interface

The control software of the visualization system is organized into two layers, a user-interface layer UIL and a communication layer CL. The UIL handles the user input over input devices like a mouse or keyboard, and converts these signals into control parameters like rotation angles etc. These signals are transferred to the communication structure that handles all communication between host and visualization system. Due to the separation of both layers, remote access to the visualization system is possible. Here, the UIL operates e.g. on an X-terminal and communicates over a high-speed network with the host computer where the communication layer resides.

The UIL provides three different interaction components to the user. With the first component the user can modify freely the weights of the reflection parameters (diffuse and specular parts for each light source). The second component allows arbitrary rotation, moving, and zooming of the data cube; with a 3D mouse as input device. Finally, the third component enables the user to directly modify the data cube. Arbitrary cubic regions of the original data cube can be selected for visualization. Moreover, the UIL supports simple 3D segmentation. Our segmentation approach is based on an arbitrary mapping of feature vectors onto density values. Feature vectors which consist of elements like gray value, local variance, or texture measures are calculated and mapped to a density value for each voxel before visualization.

Besides standard I/O functions—loading the data cube, storing the result image—, a recording feature has been added. This feature allows to record the visualization result while manipulating the control parameters, e.g., reflection parameters, rotation parameter etc.

For visualization each 100 ms the user input is transferred to the communication layer. This layer maps the parameters obtained from the UIL into the control structure used by rotator and DSP board, and it handles all communication protocols between host and visualization hardware. DSP and rotator board read actively their parameters from the structure and calculate the resulting 2D projection according to the underlying algorithm. The 2D projection is transferred to the communication unit which passes the data through to the UIL for representation on the screen.

Medical Applications

The system described here has a wide range of applications in the medical field. With the user interface UIL, it becomes possible for the physician to use the real-time rotator/raytracer as a 'digital endoscope', enabling him to navigate through any hollow organ found in the human body. Moreover, if resolution of the CT and MR scanners is accordingly improved, it is conceivable to simulate operations like catheterizing a hollow organ or a vessel in the digital, virtual world. This opens a wide field of application in education and training of physicians.

More precisely, heart surgery, e.g., can benefit substantially from the work described here. The surgeons would appreciate an interactive method of presenting cardiac morphology in a way that allows detailed preoperative study. This intravital study would support planning and execution of surgical procedures in several aspects [16]:

- the presentation of the morphology is directly transferable to the in vivo situation, in contrast to the intraoperative situation in which the cardioplegic heart lies in the thorax with an altered form,
- complete preoperative information reduces the time of intraoperative evaluation and thus the time of aortic cross-clamp time,
- preoperative decision about the optimal surgical access to the heart increases surgical success rate,
- in complex malformations, complete and reliable diagnosis can be facilitated.

Another field of application in which we are presently working is orthopedics. The meniscus of the knee, for example, can be diagnosed with the tool presented here.

Presently, we are developing the user interface to our real-time visualization system which can be intuitively operated by e.g. the cardiac surgeon. A first prototype will use a 3D mouse for the interactive navigation through the digital data volume, and zooming in/out can be set with a slider.

The non-invasive 'digital endoscope', applied to study the anatomy and morphology of organs without harm to the patient (in fact, the patient can leave the hospital after the CT or MR scan was taken) is a first step to be taken in the direction of introducing virtual reality to medicine. Here, volume based rendering of the data has undoubtedly advantages against vector based models which are today standard in virtual reality applications. For reliable diagnosis, the texture and morphology of organ or vessel surfaces is essential and has to be presented as realistically as possi-

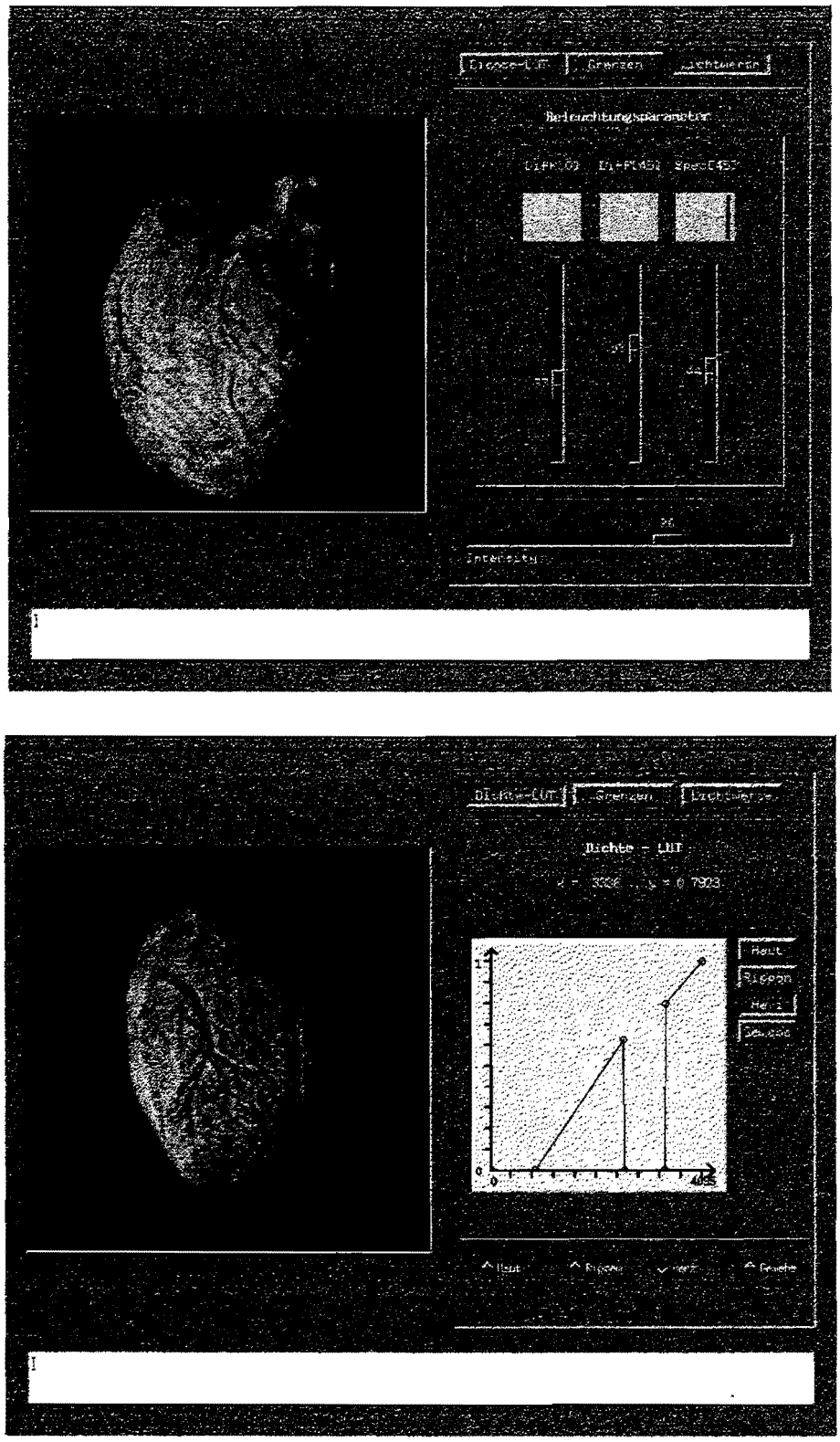

Fig. 7: Two modules of the user interface for the modification of the light components and the gray value mapping.

ble. In this sense, vectorized models seem to be of limited application in the medical field.

Once real-time visualization is available, the next step in the direction of virtual reality in medicine is interactive manipulation of the data. Appropriate tools for resection or implantation of material

have to be designed in order to provide a realistic means of preoperative simulation of surgical actions. The volume data model has to be extended in order to be able to specify features like tissue elasticity or flexibility for every voxel in the data cube.

Conclusions

Our visualization system will be operable as a prototype by autumn 1994. Its application in clinical routine is planned for 1995.
A full system amounts to about 100 kECU and has currently three limitations, the z gradient is not used, only two fixed light sources are considered, and the size of the image memory in the rotator processors limit the maximal cube size. We are currently investigating a system without these limitations.

Acknowledgments

We gratefully acknowledge the support of R. W. Günther and M. Drobnitzky from RWTH Aachen who provided the MR data set of the heart shown in Fig. 7.

References

[1] A. Watt, M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice.* ACM Press, NY, 1992.

[2] G.T. Herman, D. Webster. *Surfaces of Organs in Discrete Three-Dimensional Spaces.* In G.T. Herman, F. Natterer (Ed.): Mathematical Aspects of Computerized Tomography, Springer, Berlin, 1980, pp. 204.

[3] M.W. Vannier, J.L. Marsh, M.H. Gado, W.G. Totty, L.A. Gilula, R.G. Evens. Clinical applications of 3-dimensional surface reconstruction from CT-scans. Electromedica 4 83, 1983, pp. 121.

[4] M. Levoy. Design for a real-time high-quality volume rendering workstation. Conf. Proc. of the Chapel Hill Workshop on Volume Viszalization, Chapel Hill, North Carolina, May 18-19, 1989, pp. 85.

[5] H.-P. Meinzer, K. Meetz, D. Scheppelmann, U. Engelmann, H.J. Baur. The Heidelberg Ray Tracing Model. IEEE Comp. Graphics&Appl., Nov. '91, pp. 34.

[6] J. Lichtermann, G. Mittelhäußer. Eine Hardwarearchitektur zur Echtzeitvisualisierung von Volumendaten durch "Direct Volume Rendering". Workshop Visualisierungstechniken, Anwendungen und Entwicklungstendenzen, Stuttgart, 1991.

[7] J. Lichtermann, priv. comm.

[8] W.L. Nowinski. A SIMD Architecture for Medical Imaging. Lecture Notes in Computer Science 634, Parallel Processing: CONPAR 92 - VAPPV, Springer, 1992.

[9] W.L. Nowinski. Design for a ray casting integrated circuit. Institute of Systems Science, National University of Singapore, Singapore, 1992.

[10] N. Ewart, L. Thayer; B. Fleming; D. Voorhies. Three Ways to 3-D. Byte, Nov. '93, pp. 215.

[11] R. Yoshida, T. Miyazawa, A. Doi, T. Otsuki. Clinical Planning Support System—CliPSS. IEEE Comp. Graphics&Appl., Nov. '93, pp.76.

[12] SGI priv. comm.

[13] Foley, van Dam, Feiner, Hughes. *Computer Graphics: Principles and Practice.* Add&Wesley, Reading, MA, 2d. ed., 1990.

[14] G. Zorpette. The power of parallelism. IEEE Spectrum, Sept. '92, pp. 28.

[15] P.-E. Danielsson , M. Hammerin. High Accuracy Rotation of Images. In Graphical Models and Image Processing, Vol. 54, Nr. 4, July 1992.

[16] C.F. Vahl, H.-P. Meinzer, S. Hagl. Three-dimensional Presentation of Cardiac Morphology. Thorac. cardiovasc. Surgeon, Vol. 39 (Suppl.), 1991, pp. 198