Reconstructing Solids from Tomographic Scans -- The PARCUM II System --

D. Jackèl

TU Berlin, Institut für Technische Informatik
Franklinstr. 28-29, D-1000 Berlin 10
and
GMD-Forschungszentrum für Innovative Rechnersyteme
und -technologie an der TU Berlin (FIRST)
Hardenbergplatz 2, D-1000 Berlin 12

W. Strasser

Universität Tübingen
Wilhelm-Schickard-Institut für Informatik (WSI)
Graphisch-Interaktive Systeme (GRIS)
Auf der Morgenstelle 10 C9, D-7400 Tübingen

The computer-aided design of mechanical parts is supported by sophisticated geometric modelers and visualized by high-performance raster graphics systems allowing for a realistic display. The geometric modeler accepts the designer's inputs and converts them into a 3D model. In general, the designer has total control of the object description defining his design. But in contrast, the situation is different when dealing with existing physical objects, e.g. natural objects such as the human body, for which an explicit 3D model is required. For instance, in many applications the input information is a sequence of 2D tomographic scans. In this case the task is to combine both the interactive CAD-mode of construction as well as the scan-based mode of reconstruction in an integrated system, such that an unique 3D object representation is achieved and can be supported by hardware efficiently.
Here we describe a cellular space representation scheme which is supported by a voxel-oriented graphics system --the PARCUM II System--.

1. Introduction

The construction of 3D objects from 2D information is a discipline of both Computer Graphics as well as Computer Vision. In most cases, both use interactive computer graphics as a tool. The relationship between computer graphics and computer vision is clarified in figure 1.

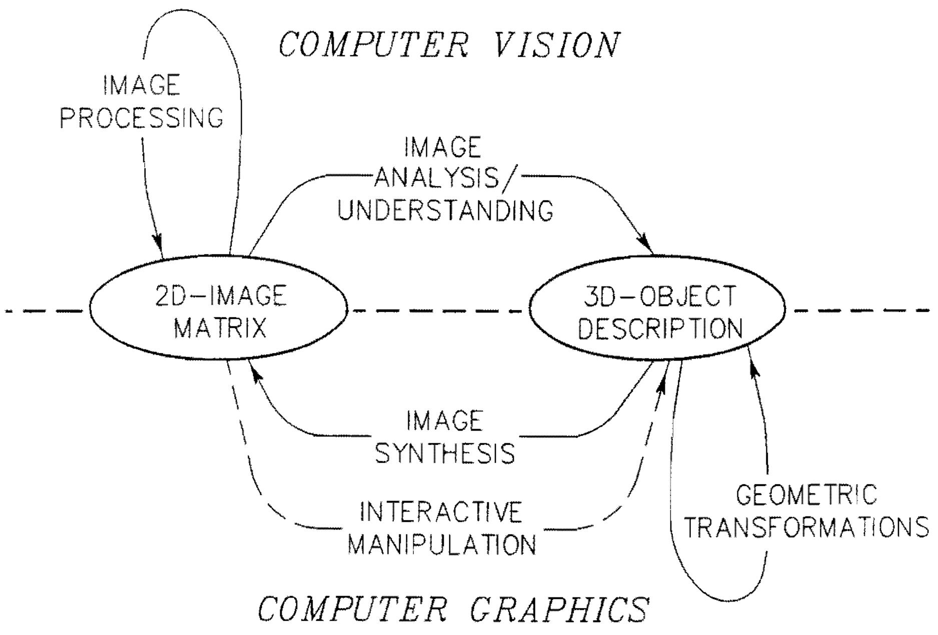


Figure 1: Relationship between the computer graphics and computer vision disciplines.

Computer graphics deals with the synthesis of images from mathematical descriptions or data structures and, moreover, provides tools to interactively manipulate the source data. In image processing, the source data is an image which is transformed into another image, e.g. an enhanced image. The transformation parameters are totally defined by the application. Image analysis may be conceived as the inverse process of image generation: it is the process of finding task-specific descriptions from images. In many image analysis and image understanding applications it is aimed at finding 3D object descriptions which allow the computation of all properties of interest. The reconstruction of 3D objects from computer tomographic scans may be understood as the inverse operation of a scan line hidden surface algorithm whose working principle is shown in figure 2.

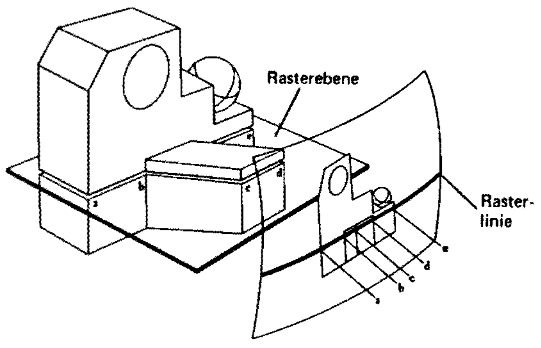


Figure 2: Working principle of scan line hidden surface algorithm [GRE82].

The reverse operation is to aggregate the scan lines of PIcture ELements (PIX-ELs) --by sophisticated methods [NAT85] not to be discussed here-- to spatial matrices of VOlume ELements (VOXELs) to be put on a stack in proper order. The result is a 3D object in a cellular space representation (see figure 3).

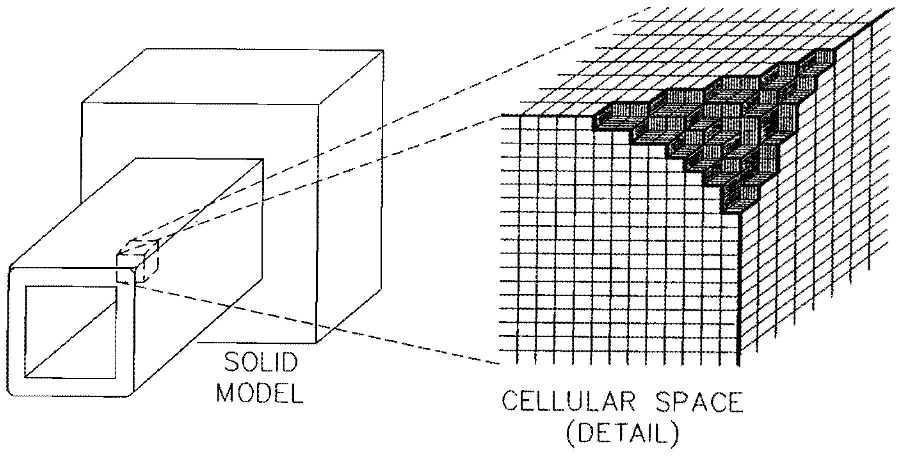


Figure 3: Cellular space representation.

If the stored representation is maintained in a physical 3D- or VOXEL memory, all operations known from solid modeling can be performed very efficiently. In the last few years some display architectures based on cellular space representations, as proposed in [GOL84], [KAU86], and [OHA85], were designed for executing the geometric modeling and display processes. These VOXEL based systems can be expanded by integrating image processing, geometric modeling, and display processes. [JAC87] describes a "Processor ARchitecture based on a CUbic

Memory" (PARCUM II) --an earlier version of the system was published in [JAC85]-- which is especially designed for all three purposes mentioned above. Figure 4 depicts the main components of this system, which is able to store binary as well as non-binary VOXELs. Non-binary VOXELs are important for using PARCUM in the field of biomedical applications --for details see [STI87]. In this case, the PARCUM II System can be considered as a component of a "Medical Work Station" (MWS) as proposed by Lemke [LEM86]. In the following, the main components of the PARCUM II System are described.

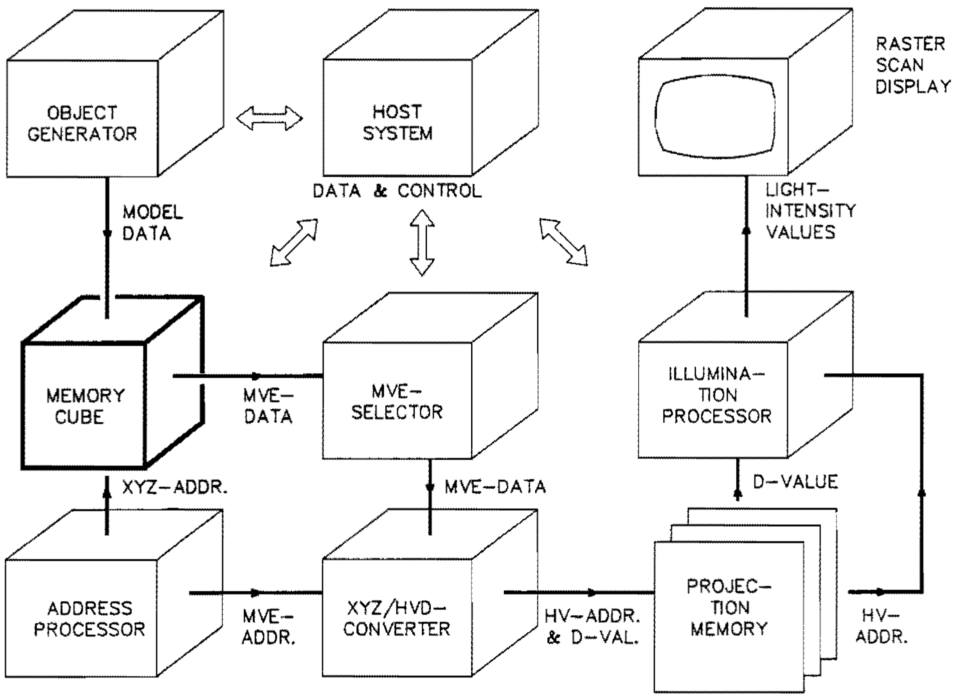


Figure 4: The architecture of PARCUM II-System.

2. The PARCUM II System

Memory cube

The core of the PARCUM II System is a 3-dimensionally organized memory known as a *memory cube*. For the purpose of *non-binary* representations of spatial images, the address space is reduced to 256^3 cells, whereby each cell has assigned a 7-bit attribute (e.g. anatomic label, VOXEL density ect.). The organization of the memory system permits the simultaneous reading and writing of a cubic 64-bit data set, called Macro Volume Element (MVE) (see figure 5).

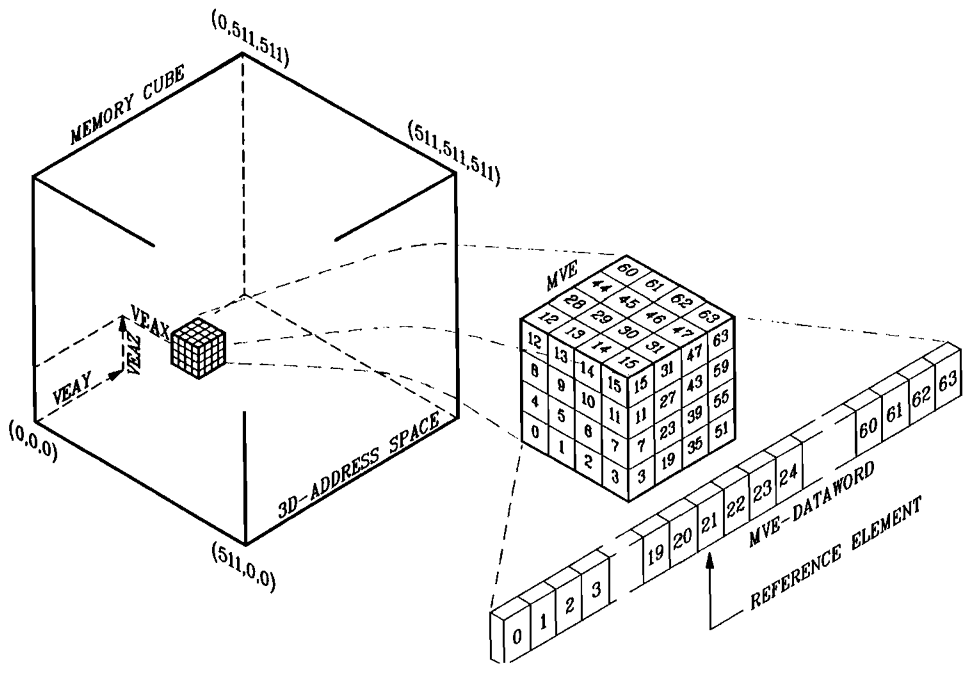


Figure 5: Principle of reading/writing a MVE from/to the 3D address space of the memory cube.

The MVE-address is partitioned in 9-bit (binary reps.) or 8-bit (non-binary reps.): *XVEA*, *YVEA*, and *ZVEA* address parts, which due to their spatial order are called address coordinates. To obtain the 3D data access features, the memory system is subdivided into 64 modules with a capacity of 2-Mbytes each. As shown in figure 6, a 21-bit address path is assigned to each of these memory modules. These addresses are formed by means of combining three subaddress busses: X_l, Y_m, and Z_n of the set X_1,..., X_3; Y_0,..., Y_3; and Z_0,...., Z_3, which are arranged in the form of a spatial grid.

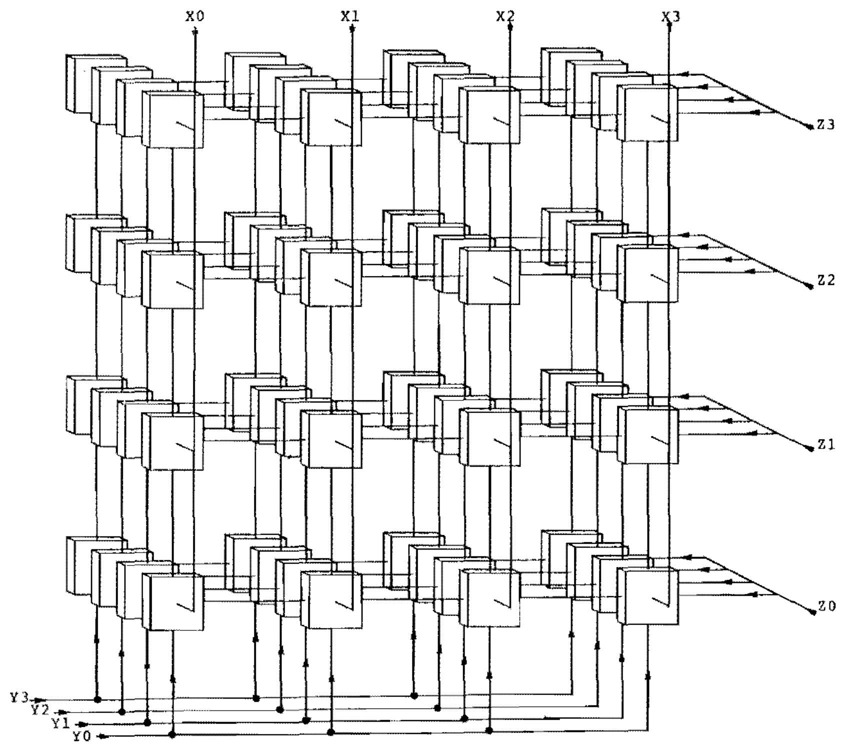


Figure 6: Arrangement of the subaddress busses X_1 to Z_3.

On the basis of the guidelines shown in table 1a-c, a *converter unit* generates the X_1 to Z_3 assignments derived from the above mentioned *XVEA*, *YVEA*, and *ZVEA* address parts. The generation of a sub-bus address assignment is controlled by the least significant *VEA* bits (*LS-VEA* bits) *xvea* (0); *xvea* (1); *yvea* (0); *yvea* (1); *zvea* (0) and *zvea* (1).

[*xvea*(1)	*xvea*(0)]	address modification of *XADR*				shift parameter	
						READ	WRITE
[0	0]	*XADR*	*XADR*	*XADR*	*XADR*-4	-1	1
[0	1]	*XADR*	*XADR*	*XADR*	*XADR*	0	0
[1	0]	*XADR*+4	*XADR*	*XADR*	*XADR*	1	-1
[1	1]	*XADR*+4	*XADR*+4	*XADR*	*XADR*	2	-2
	subaddr.bus→	X_0	X_1	X_2	X_3		

(a)

[*yvea*(1)	*yvea*(0)]	address modification of *YADR*				shift parameter	
						READ	WRITE
[0	0]	*YADR*	*YADR*	*YADR*	*YADR*-4	-1	1
[0	1]	*YADR*	*YADR*	*YADR*	*YADR*	0	0
[1	0]	*YADR*+4	*YADR*	*YADR*	*YADR*	1	-1
[1	1]	*YADR*+4	*YADR*+4	*YADR*	*YADR*	2	-2
	subaddr.bus→	Y_0	Y_1	Y_2	Y_3		

(b)

[*zvea*(1)	*zvea*(0)]	address modification of *ZADR*				shift parameter	
						READ	WRITE
[0	0]	*ZADR*	*ZADR*	*ZADR*	*ZADR*-4	-1	1
[0	1]	*ZADR*	*ZADR*	*ZADR*	*ZADR*	0	0
[1	0]	*ZADR*+4	*ZADR*	*ZADR*	*ZADR*	1	-1
[1	1]	*ZADR*+4	*ZADR*+4	*ZADR*	*ZADR*	2	-2
	subaddr.bus→	Z_0	Z_1	Z_2	Z_3		

(c)

Table 1a-c: Converting guidelines to generate the address assignment for the subaddress busses X_0 to Z_3.

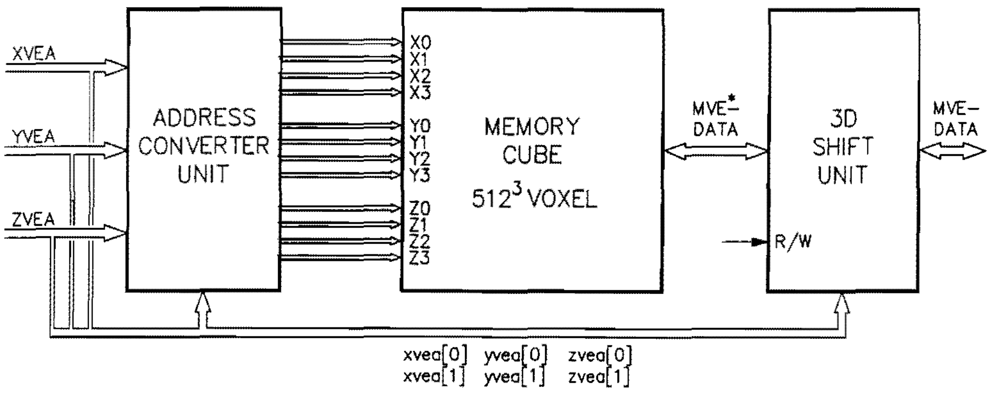


Figure 7: The architecture of the 3D memory system.

Likewise, depending on the values of the above mentioned *LS-VEA* bits, the VOXEL data are cyclically interchanged in all three coordinate directions after every reading access. In order to reverse this VOXEL interchanging process, a 3*D shift unit*, which is also controlled by the *LS-VEA* bits, is necessary. For writing accesses, the cyclical VOXEL interchange is performed beforehand. Figure 7 illustrates the architecture of the memory system.

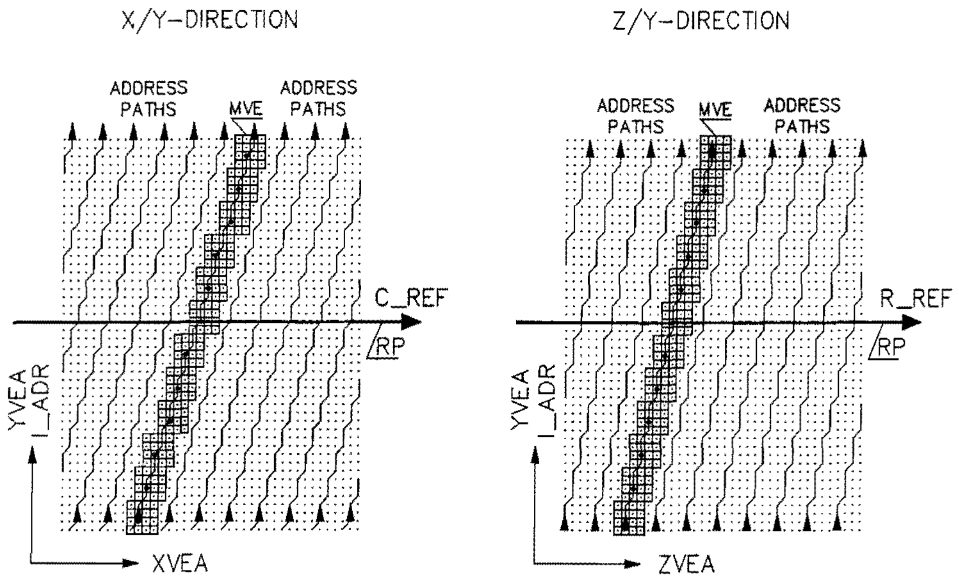


Figure 8: Surface-VOXEL detection by parallel aligned address paths.

Address Processor.

The task of an address processor is to detect all surface VOXELs of a *cellular* representation which are elements of the solid object's volume model. For this purpose, the memory cube is penetrated by parallel address paths, shown in figure 8.

In each search iteration the number of VOXELs of a MVE, which are elements of the volume model, is determined. We obtain this number of VOXELs by adding all binary VOXEL values of the examined MVE:

$$QS(MVE) = VE(0) + \cdots + VE(63)$$

Depending on the $QS(MVE)$-value, the search process is controlled by the following rules:

$QS(MVE) = 0$:	- determine next $QS(MVE)$.
$0 < QS(MVE) < 64$:	- execution of next incremental address step; - processing the VOXEL elements by the selector unit and the XYZ/HVD converter; - determine next $QS(MVE)$.
$QS(MVE) = 64$:	- processing of the VOXEL elements by the selector unit and XYZ/HVD converter; - termination of the address path.

The origin of all the address paths is a so-called *reference plane* (RP) illustrated in figure 9. Because all the address paths are parallel, it is sufficient to calculate the address sequence of only one address path, the so-called *reference path*. The reference path must be calculated during the initialization of the address processor. All other paths can now be determined by a parallel shift of the reference path. In principle, each address path is defined by a column address axis (C_REF-direction), a row address axis (R_REF-direction), and an incremental address axis (I_ADR-direction). The precalculated row parts $R_p(I\_ADR)$ and column parts $C_p(I\_ADR)$ of all reference path coordinates are written into a C_p- and a R_p-memory table. Both *look ahead* memories are parts of an *address path generator*, shown in figure 10, producing an address coordinate sequence using the equation:

$$\begin{array}{lcl} I\_ADR & & I\_ADR \\ R\_ADR & = & R\_REF + Rp(I\_ADR) \\ C\_ADR & & C\_REF + Cp(I\_ADR) \end{array}$$

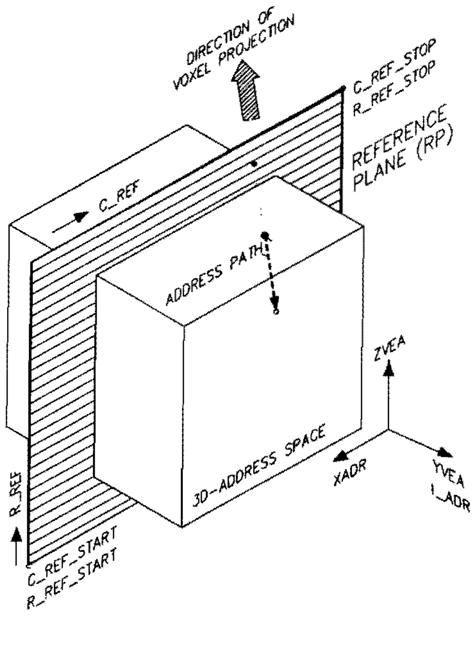


Figure 9: Principle of the surface VOXEL detection by linear address paths.

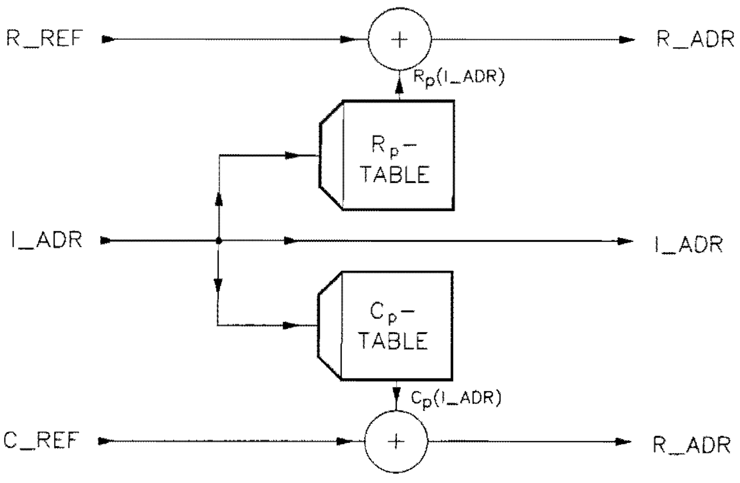


Figure 10: Principle of the address path generator.

The C_p and R_p tables are referenced by I_ADR. The memory table contents $R_p(I\_ADR)$ and $C_p(I\_ADR)$ will be calculated for each incremental address (I_ADR). By adding reference plane addresses C_REF and R_REF with $R_p(I\_ADR)$ and $C_p(I\_ADR)$, the parallel shift of the above mentioned reference address path is obtained.

Selector unit.

The task of the *selector unit* is to eliminate the VOXELs which are invisible from the position of the virtual viewer. In the following, the principle of this elimination process, called *VOXEL-reduction*, will be explained by the example of a 2D data field in a simplified way.

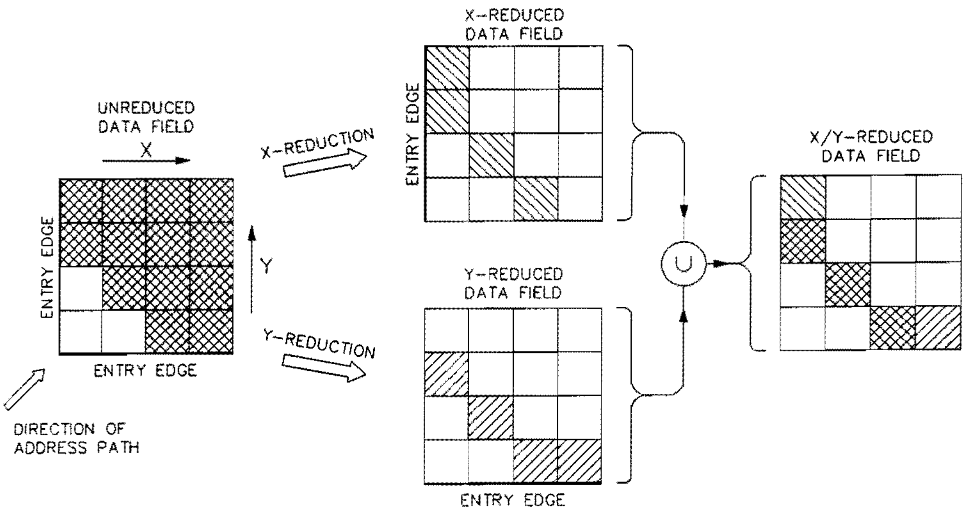


Figure 11: Principle of the VOXEL-reduction.

The reduction is carried out in the direction of the X and Y address axis with a group of elements from the two dimensional data field which has the same addresses. As shown in figure 11, there are 2*4 quadruple groups of data elements. Each of these groups will be processed both in the X and Y directions by means of a logical network. This network eliminates all elements from the 2D data field, except those nearest to the entry edges. The reduced X and Y data fields are thus obtained. The final result, i.e. the reduced X/Y field, is acquired through the boolean disjunction of the corresponding elements in the reduced X and Y fields. This principle can be applied analogously for MVEs by executing this reduction process in $XVEA$, $YVEA$, and $ZVEA$ directions with 3*16 quadruple VOXEL groups. In addition to the elimination of the hidden VOXELs, the *selector unit* has two additional functions. Firstly, for processing *non-binary cellular* representations, the VOXEL attribute values will be determined for the purpose of the so-called *threshold segmentation*. Secondly, for the visualization of object intersections, the selector unit marks all the HV-entries of the projected VOXELs (see principle of *XYZ/HVD*-Converter) of all the spatial address coordinates which are placed directly at the intersection planes of the displayed object.

XYZ/HVD Converter.

By means of a parallel projection, the *XYZ/HVD* converter maps the 3-dimensional address coordinates of the surface VOXELs onto a virtual projection plane. The spatial orientation of the projection plane is given by angles, α and δ, in a spherical coordinate system interactively set by the user. The *XYZ/HVD*-conversion is defined by the equations 1a-c:

$$H\_ADR = (YVEA - XVEA * \tan\alpha) * \cos\alpha \tag{1a}$$

$$V\_ADR = -(XVEA + YVEA * \tan\alpha) * \cos\alpha * \sin\delta + ZVEA * \cos\delta \tag{1b}$$

$$D = DIM * \sqrt{3} - (XVEA + YVEA * \tan\alpha) * \cos\alpha * \cos\delta + ZVEA * \cos\delta \tag{1c}$$

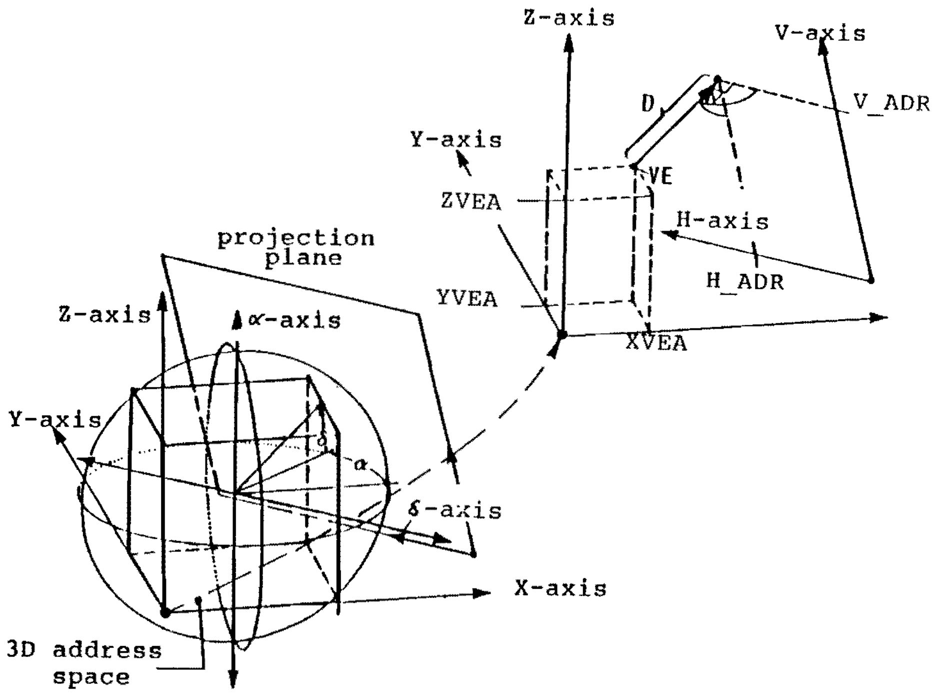


Figure 12: Principle of the projection process.

As figure 12 illustrates, the distance values (D) will be mapped onto the projection plane, which is realized as a 2D frame buffer, under their corresponding

(H_ADR, V_ADR)-address. The D-values are defined here as perpendicular distances between the virtual projection plane and the spatial address coordinates ($XVEA$, $YVEA$, $ZVEA$) of the surface VOXELs. If more than one VOXEL-address coordinates are mapped to the same (H_ADR, V_ADR)-address, each of the smallest value of the set of D-values has to be entered or reentered into the projection plane. The procedure used here is well-known as "Z-buffer hidden surface removal." After completion of the mapping process, we obtain a distance matrix $D(h, v)$ from the object's volume model which can also be considered a *depth map*.

As mentioned above, the XYZ/HVD-conversion is carried out by means of *look-ahead* tables. This is possible, because the angles α and δ remain constant during the entire conversion process. Therefore, depending solely on variable address coordinates, all multiplications in equations 1a-c can be substituted by sequences of values representing linear functions. These values are then loaded into the *look ahead* tables. For example, if the simple equation $c = a * b$ is given (b = constant), then we can simply write $c = f(a)$. This function can easily be implemented in hardware using *look ahead* tables. From equations 1a-c we derive the corresponding equations 2a-c. From equation 1a we get the equation 2a

$$H\_ADR = H\_TAB2(YVEA + H\_TAB1(XVEA)) \tag{2a}$$

with the table entries:

$$\begin{aligned} H\_TAB1\,(H\_ADR1) &= -H\_ADR1 * \tan\alpha \\ H\_TAB2\,(H\_ADR2) &= H\_ADR2 * \cos\alpha \end{aligned}$$

From equation 1b we find equation 2b

$$V\_ADR = V\_TAB4\{V\_TAB2[XVEA + V\_TAB1(YVEA)] + V\_TAB3(ZVEA)\} \tag{2b}$$

with:

$$\begin{aligned} V\_TAB1\,(V\_ADR1) &= V\_ADR1 * \tan\alpha \\ V\_TAB2\,(V\_ADR2) &= -V\_ADR2 * \cos\alpha * \sin\delta / \cos\delta \\ V\_TAB3\,(V\_ADR3) &= V\_ADR3 \\ V\_TAB4\,(V\_ADR4) &= V\_ADR4 * \cos\delta \end{aligned}$$

and finally from equation 1c we get the equation 2c

$$D = D\_TAB4\{D\_TAB2[XVEA + D\_TAB1(YVEA)] + D\_TAB3(ZVEA)\} \tag{2c}$$

with:

$$\begin{aligned} D\_TAB1\,(D\_ADR1) &= D\_ADR1 * \tan\alpha \\ D\_TAB2\,(D\_ADR2) &= D\_ADR2 \\ D\_TAB3\,(D\_ADR3) &= -D\_ADR3 * \sin\delta / (\cos\alpha * \cos\delta) \\ D\_TAB4\,(D\_ADR4) &= D\_ADR4 * \cos\alpha * \cos\delta \end{aligned}$$

The table entries for the equations 2a-c are only valid if α and δ are within the range of values from $-45°$ to $45°$. Further details of this method are given in [JAC87]. Figure 13 shows the hardware realisation of the equations 2a-c.

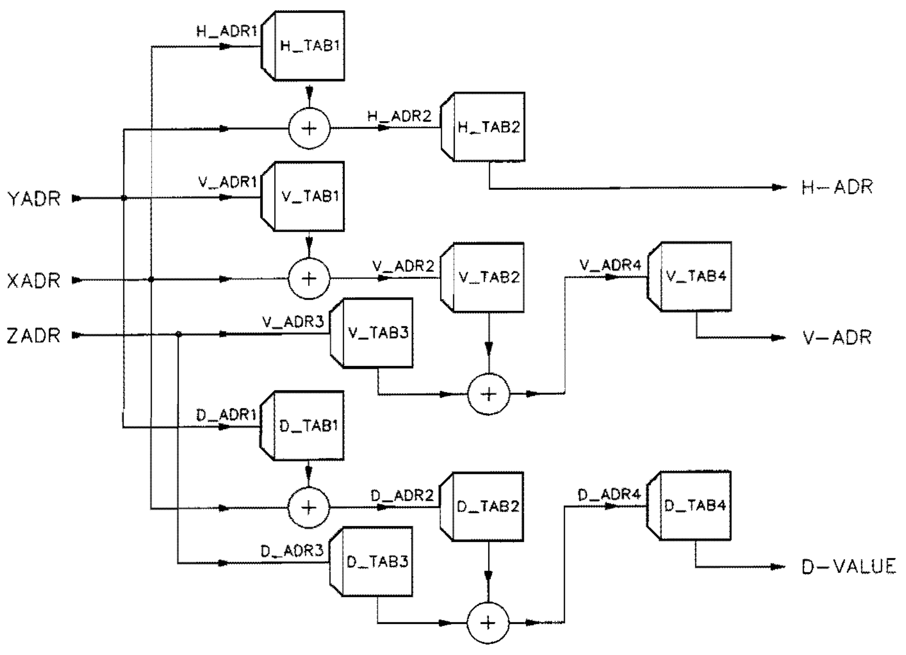


Figure 13: Principle of the *XYZ/HVD*-converter.

If the spatial orientation of the projection plane is changed, all the table entries must be updated. In order to obtain a sufficient short update time interval, it is, therefore, important to keep the table memories as small as possible. Thus we have to restrict the range of values for α and δ to $\pm 45°$. If α and δ are outside, all the VOXEL-address coordinates of the object must be transformed into the restricted range of values by an appropriate number of $\pm 90°$ rotations.

Illumination processor

The task of the illumination processor, whose principle is illustrated in figure 14, is to achieve a realistic display by shading the projected image of the volume model's VOXEL surface. In the following we explain the principle of its function, which is based on the so-called *gradient shading* method.

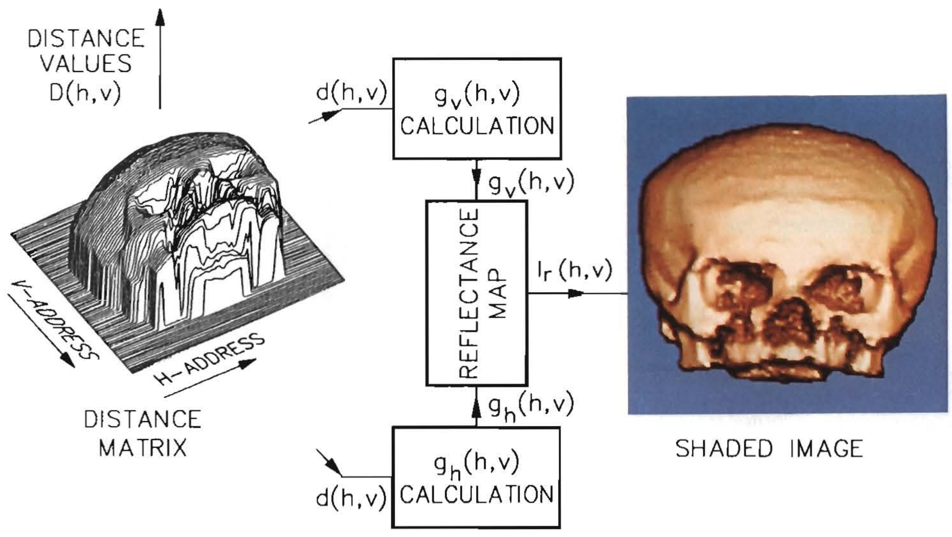


Figure 14: Principle of illumination processor.

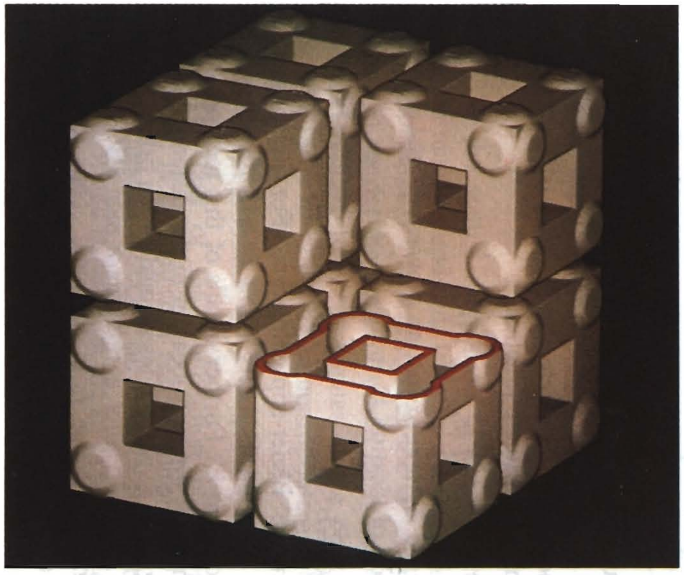
We consider the 2D distance value matrix $D(h, v)$ as a set of one-dimensional row and column functions $D_v(h)$ and $D_h(v)$. Firstly, the partial derivative must be calculated by differentiating $D_v(h)$ and $D_h(v)$. For determining the partial derivative at the location k of the one-dimensional discrete function

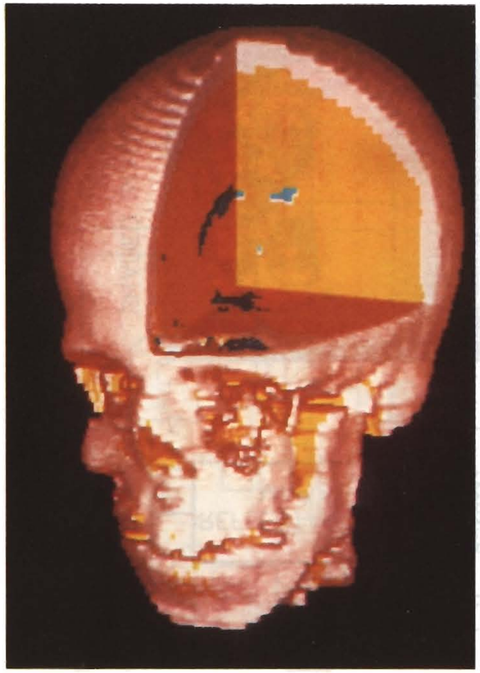
$$D_v(h_i) = f(h_0), f(h_1), \dots, f(h_k), \dots, f(h_{n-1}), f(h_n)$$

a differentiable function must be defined by approximating f over a partial set of control points within the neighbourhood of location k

$$f(h_{k-m}), \dots, f(h_k), \dots, f(h_{k+m}).$$

The method used here is based on the Bézier-approximation. For defining a differentiable function, the number of control points $(2m+1)$ will be chosen depending on the local curvature at location k. Further details of this method are given in [JAC87]. According to this method, for each location in the distance value matrix $D(h,v)$ the partial derivatives g_h and g_v will be determined in the orthogonal directions along columns and rows. By addressing a *reflectance map* [HORN77] with a g_h/g_v pair, a reflected light intensity value I_r is obtained. Thus for each possible g_h/g_v pair, the reflectance map contains the precalculated I_r value. Consequently, all entries of the distance matrix $D(h, v)$ are assigned appropriate $I_r(h, v)$ values which **in toto** make up a shaded image of the projected 3D object on the raster screen. The restrictions of this gradient shading method are discussed in [JAC87]. Figure 15a-c shows some shading results.





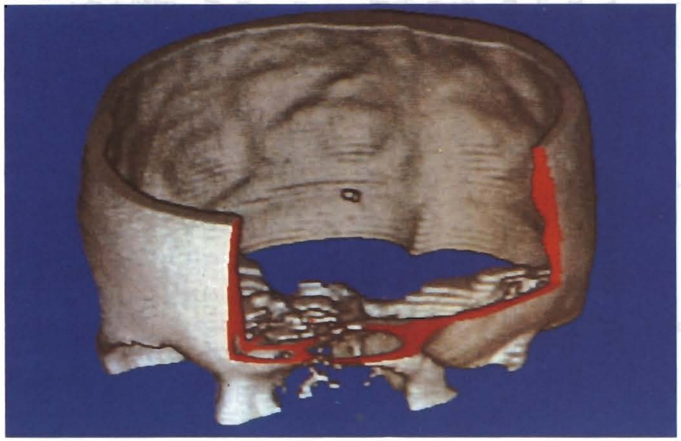


Figure 15:

a) *(upper left)* Sectional view of hollow bodies; constructed by CSG method; spatial resolution 512^3 VOXELs.

b) *(lower left)* Sectional view of a human skull; reconstructed from 37 tomographic scans; spatial resolution 512^3 VOXELs.

c) *(upper right)* Sectional view of a human head; spatial resolution 256^3 VOXELs (non binary CS-reps.); reconstructed from 67 thresholded tomographic scans; blue: ventricular system, yellow: brain tissue, white: skull bone.

Object generator

The object generator is the hardware part of a solid modeling system and enables the interactive generation of volume objects inside the memory cube. At the last stage of a modeling system, the generator has been designed to generate volume primitives, e.g. polyhedras, from *half spaces*. In order to generate volume models, according to the well-known "Constructive Solid Geometry"-Method (CSG), such volume primitives must be combined with binary set operations. To generate volume primitives, the plane parameter sets

$$\{A_1, B_1, C_1, D_1\}, \ldots, \{A_n, B_n, C_n, D_n\}$$

of the half space equation

$$A_k \star XVEA + B_k \star YVEA + C_k \star ZVEA - D_k = 0$$

are necessary for initialising the object generator. This approach is a 3D-expansion of the Pixel-Planes concept [POU85].

Host system

Apart from the I/O-operations, process control of the PARCUM_II System is the task of the host system. Its further functions are to calculate the *look ahead* table entries and to control the entire system.

3. The Actual Working State

Presently, the PARCUM_II design concept has been completed. Tests of the essential hardware parts (e.g. memory cube, projection memory as well as integration within a PARCUM II experimental system as shown in figure 16) have also been made. This experimental system consists mainly of a 512^3-VOXEL memory cube and a MC68020 processor for PARCUM emulation. The emulation processor has no operating system, but is controlled by a second MC68000 system.

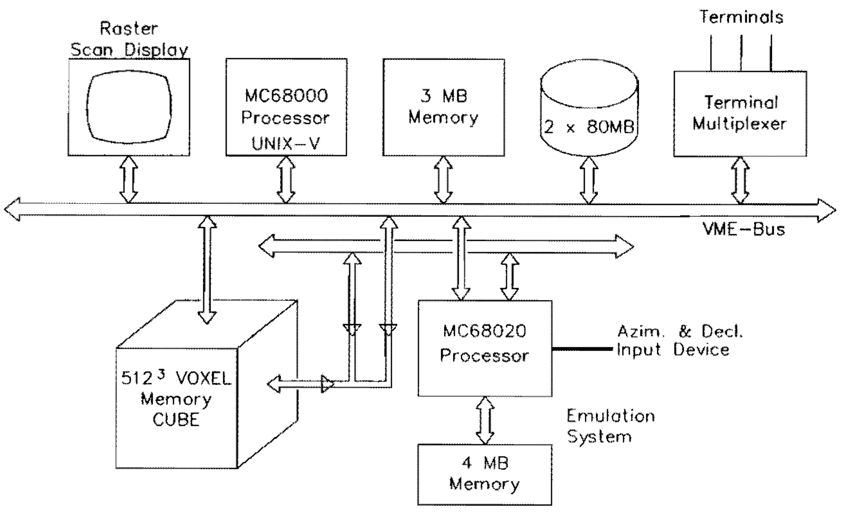


Figure 16: The architecture of the PARCUM II experimental system.

The emulated version of the PARCUM_II System is a compromise between supporting a) a flexible program environment and b) an acceptable run time for display processes. For the complete display processes executed by the experimental PARCUM_II System, the following CPU-run times are obtained:

110 seconds CPU-run time for object in figure 15a,
80 seconds CPU-run time for object in figure 15b and
38 seconds CPU-run time for object in figure 15c.

4. Acknowledgements

We wish to express our appreciation to Dr. Stiehl and J. Weida for the critical reading of the draft version, and the helpful preparing of this paper by Y. M. Guo. The Deutsche Forschungs-gemeinschaft (DFG) supported this research under a grant for the Sonderforschungsbereich SFB 0203 at the Berlin Technical University.

References

[GOL84] Goldwasser S.: "A Generalized Object Display Processor Architecture", IEEE Computer Graphics & Applications, October 1984.

[HOR77] B.K.P. Horn: "Understanding Image Intensities", Artificial Intelligence, Vol.8, No.2, pp.201-231.

[JAC85] Jackèl D.: "The Graphics PARCUM System: A 3D-Memory Based Computer Architecture for Processing and Display of Solid Models", Computer Graphics Forum, 4 (1985) 21-35.

[JAC87] Jackèl D.: "Eine Rechnerarchitektur zur Visualisierung und Verabeitung von Cellular- Space-Repräsentationen", TU Berlin, FB Informatik, Dr.-Ing. Dissertation (1987).

[KAU86] Kaufman A. : "Voxel-Based Architectures for Three-Dimensional Graphics", Proc. IFIP'86, Dublin, Ireland; September 1986.

[LEM86] Lemke H.U.: "Medical Work Stations in Radiology", in Höhne K.H. et al (Ed.), Pictorial Information System in Medicine, Springer-Verlag, (1986) pp. 421-431.

[OHA85] Ohashi T., Uchiki T., Tokoro M.: "A Three- Dimensional Shaded Display Method for Voxel Based Representations", Proc. EUROGRAPHICS'85, Nice, France; September 1985.

[POU85] Poulton, J. et. al.: "Pixel-Planes: Building a VLSI-based Graphics System", Proc. Chapel Hill Conference on Very Large Scale Integration, 1985.

[STI87] Stiehl H.S., Jackèl D.: "On a Framework for Processing and Visualizing Spatial Images", in H.U. Lemke et al (Eds), Proc. Computer Assisted Radiology CAR'87, Springer-Verlag, July 1987.