CS 152
Computer Architecture and Engineering
Lecture 24 -- Voxel Processing

2014-4-22

John Lazzaro

(not a prof - "John" is always OK)

TA: Eric Love

www-inst.eecs.berkeley.edu/~cs152/

Play:

Today: Processing volumetric data

- **Voxels:** Representation of volumetric density, used in medical imaging.

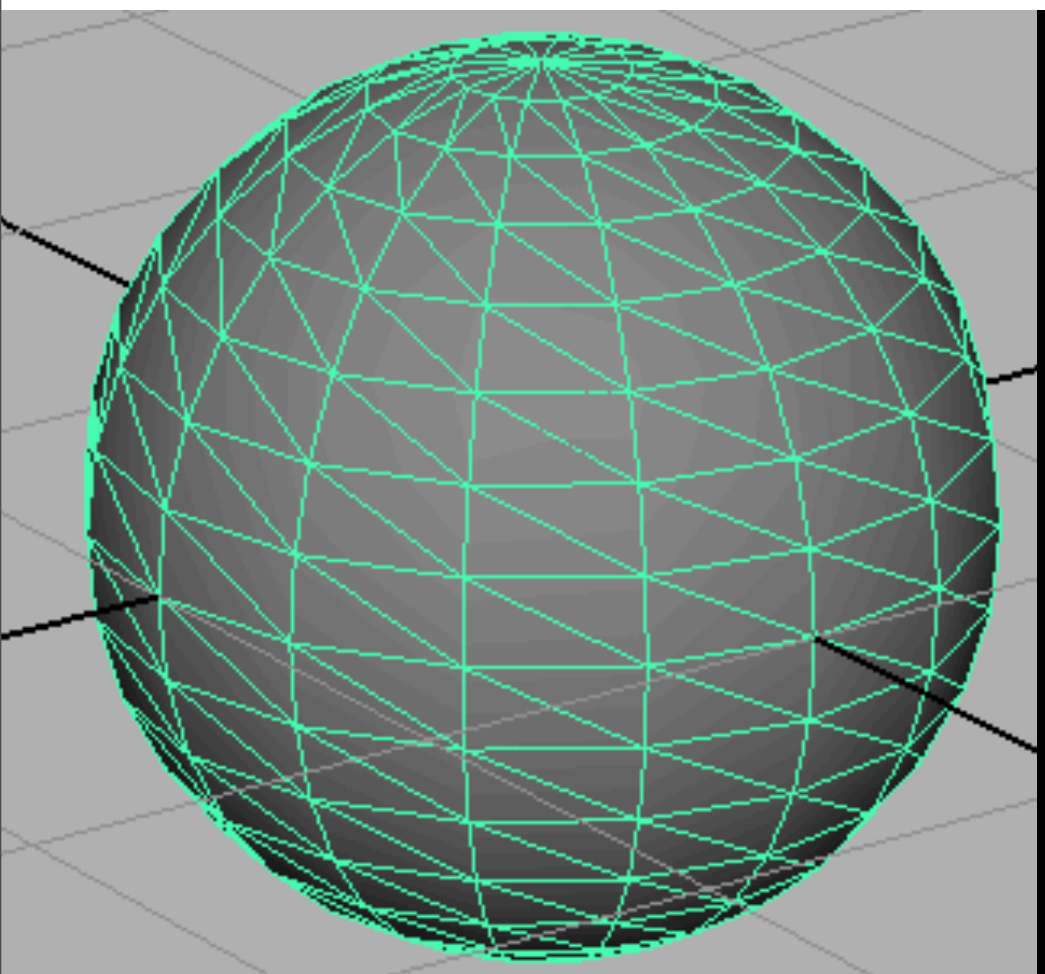
- **The Voxel Processor:** Marketed as a "Physician's Real-Time Workstation".

Short Break

- **Architecture:** The design of the voxel processor, a 256^3 real-time system.

Maps: Look at the surface of the Earth

Sometimes, the surface of an object is not the interesting part.



Computed tomography (CT) medical imager data ...

"Surface view"
of the skull
of a head
trauma patient.

Massive nose
damage.

To plan surgery,
doctors need
to look inside.

Always wear your seat belt ... and shoulder restraint.

CT "slices" through the head.

CT measures "density" of small volumes.

White pixels are bone (most dense)

"Soft" brain tissues are in shades of gray.

Empty space is black.

To begin: How does CT work?

Conceptually ...

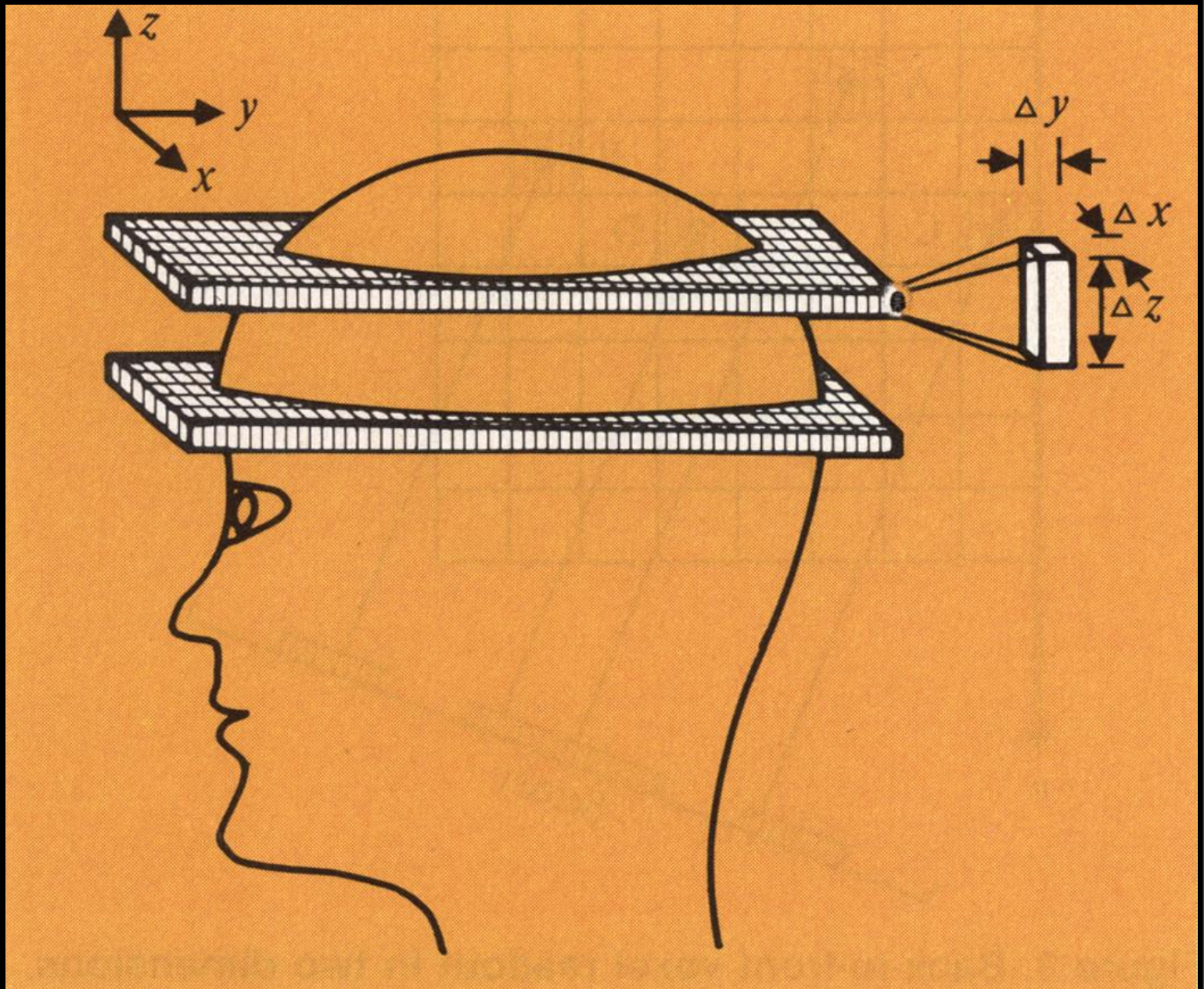
(1) Slice off thin sheets of the skull.

(2) Chop slices into 2-d arrays of cubes.

(3) Weigh each cube.

(4) Create 3-D weight matrix: d[x][y][z].

2-D slices of Voxels (volume elements)



d[x][y][z]? Density: weight per unit volume

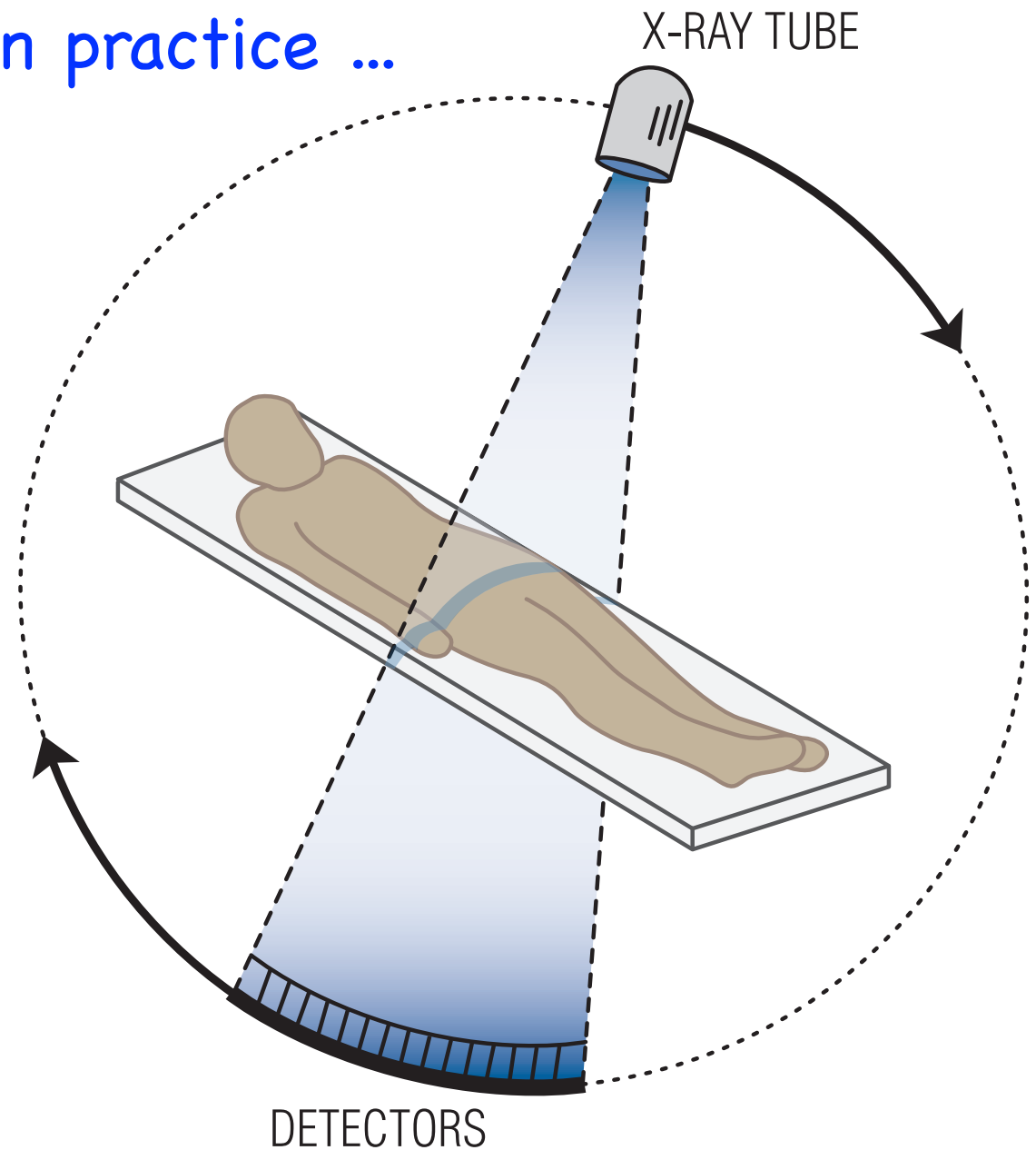
X-ray source and detector array are mounted on opposite sides of a rotating hoop.

A motorized table moves the patient through the hoop.

X-ray absorption correlates with material density.

Algorithms convert raw sensor data into a voxel array.

In practice …

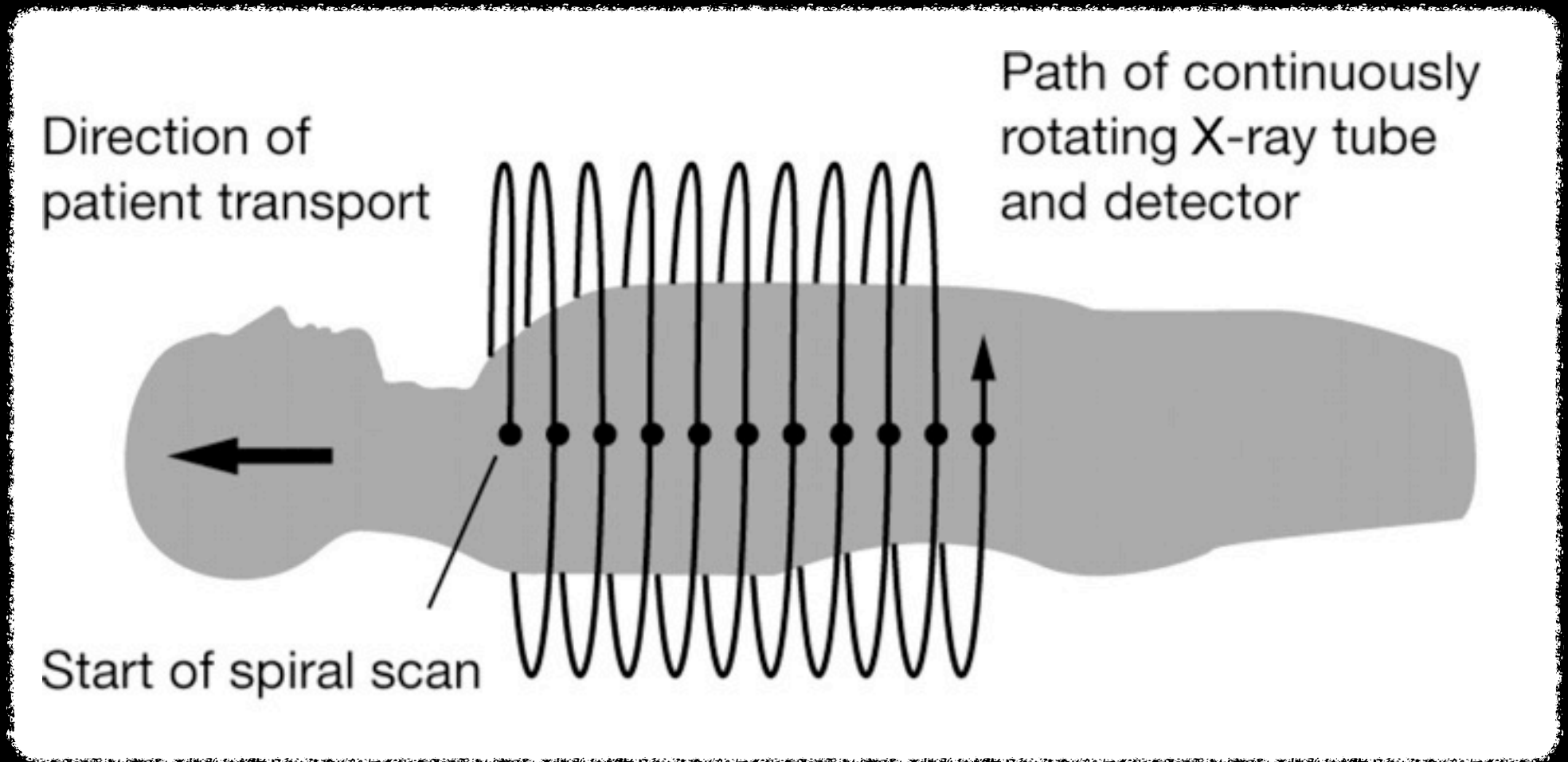


The patient is exposed to a fan-shaped x-ray beam and the projected image is detected on a thin, semi-circular digital x-ray detector.

A CT machine …

Technician safety is crucial: the stray X-rays a technician receives over a career increases his or her cancer risk significantly.

The raw data …



Processed to create densities …

After processing ...

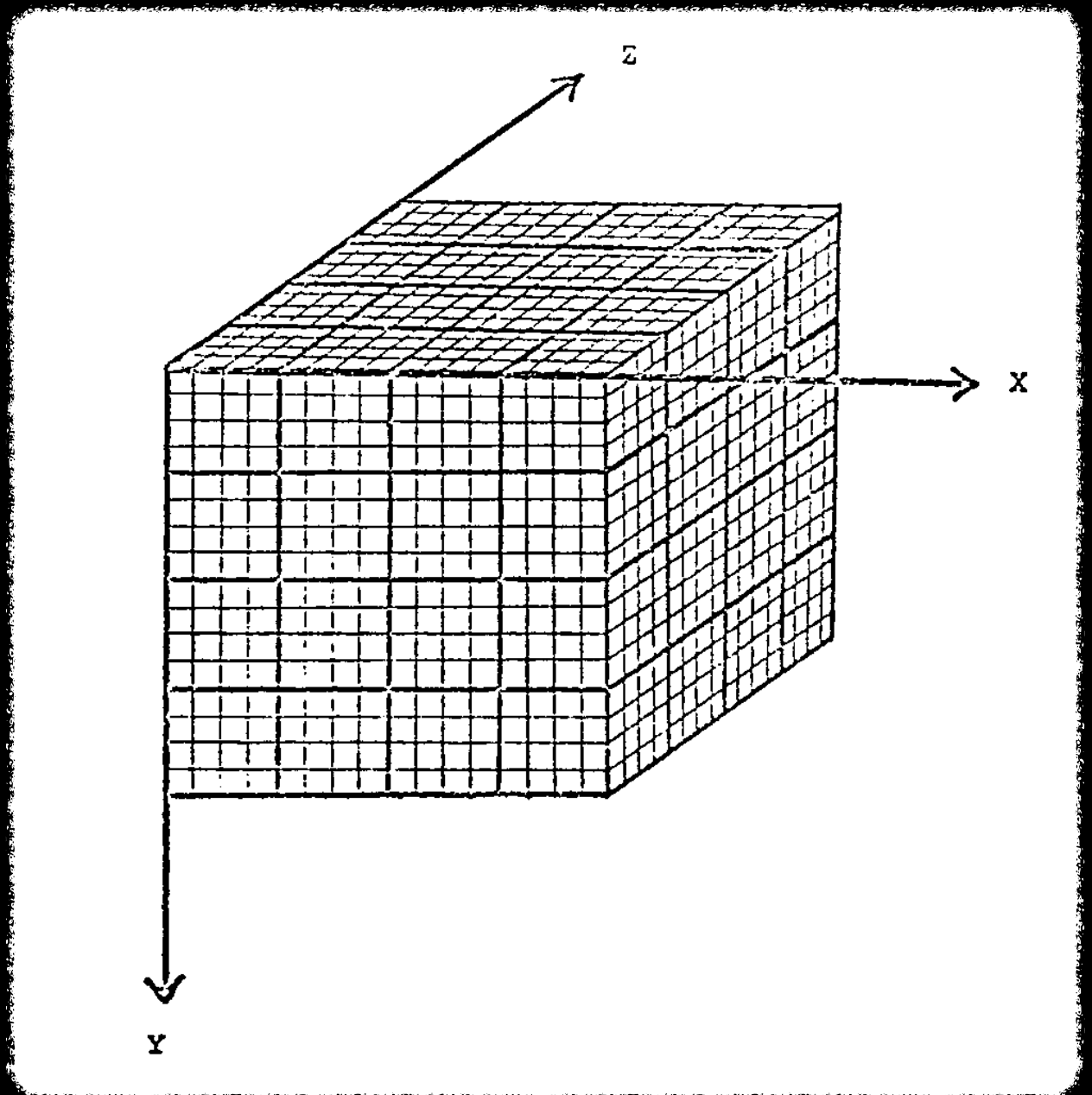
A 3-D matrix of cubes, in object space (X,Y,Z).

8-bit density value stored for each cube (0 = "air").

256^3 = 16 MB = 10 inch cube (for 1mm voxels)

0.125 mm voxels? 8 GB

Interesting to computer architects because n^3 grows so quickly!

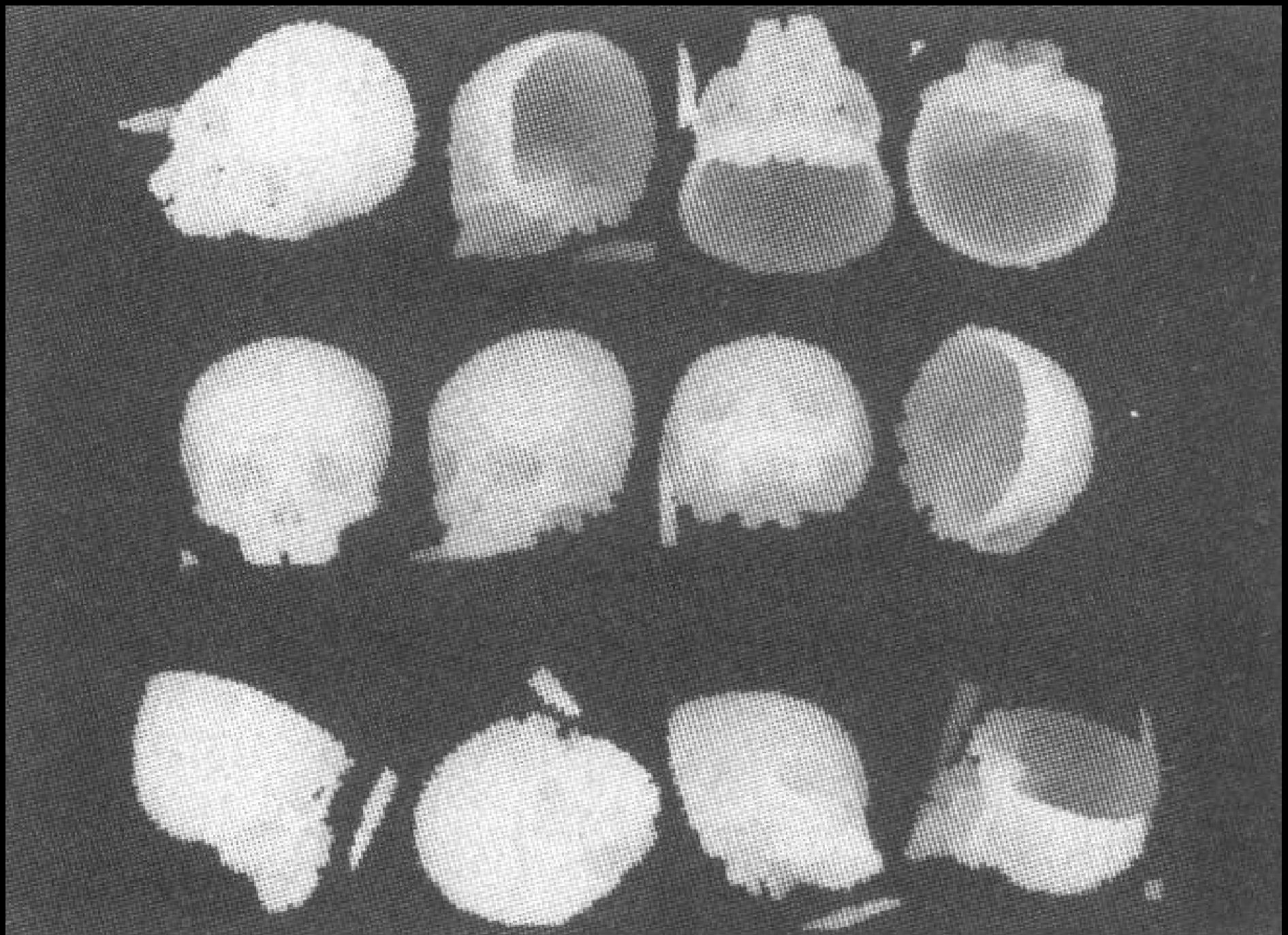


Real-time interaction for surgical planning ...

For some applications, rendering a standard static view is sufficient ...

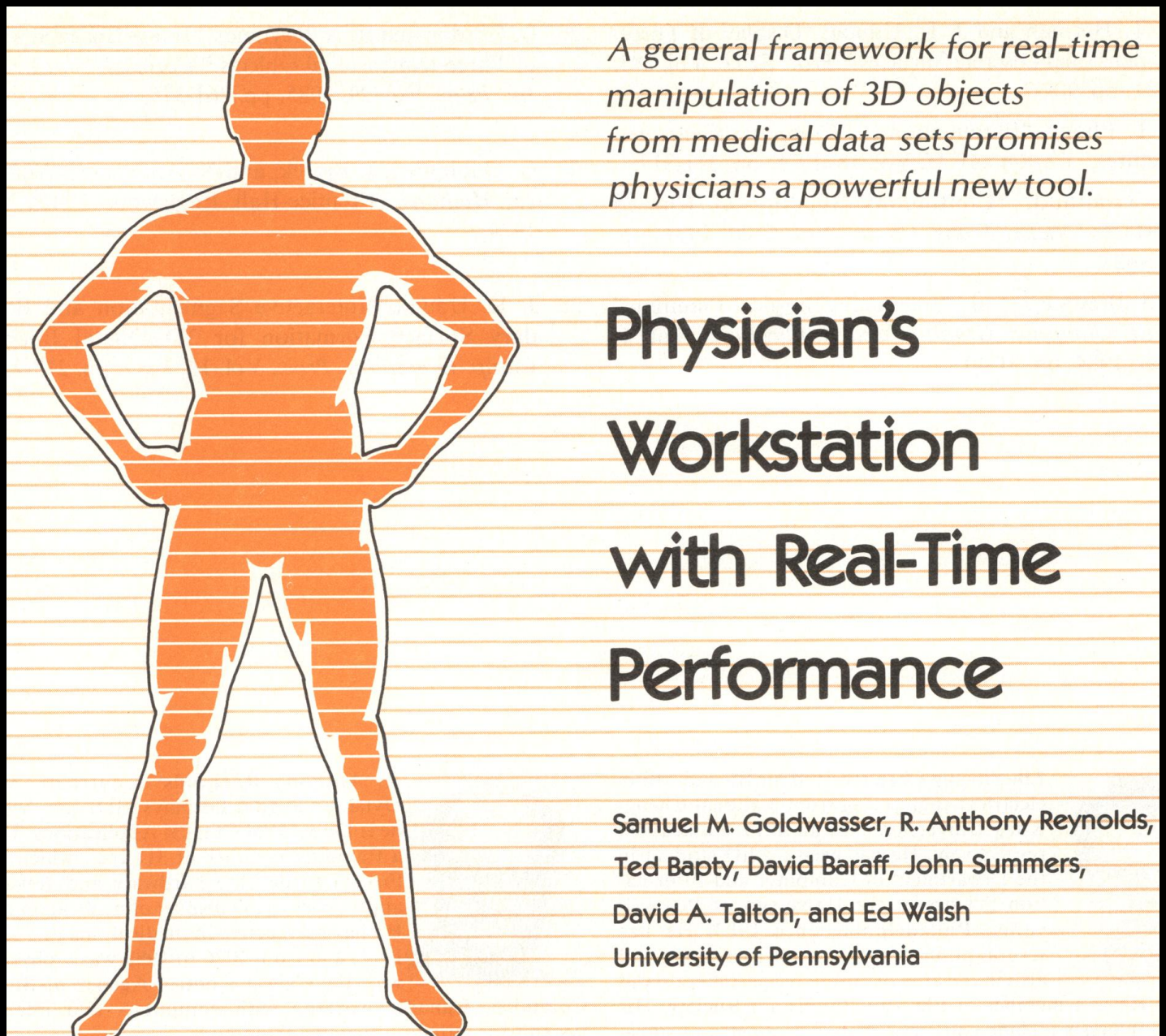
But for planning surgery, physicians want real-time interaction.

Trackball rotation of a 256^3 skull at 30 frames/sec



We describe this computer, that manipulated large voxel databases in their native form, @ 30 frames/sec.

1980s academic research project that became a company that shipped products that were used in hospitals through the 2000s ...



A general framework for real-time manipulation of 3D objects from medical data sets promises physicians a powerful new tool.

Physician's Workstation with Real-Time Performance

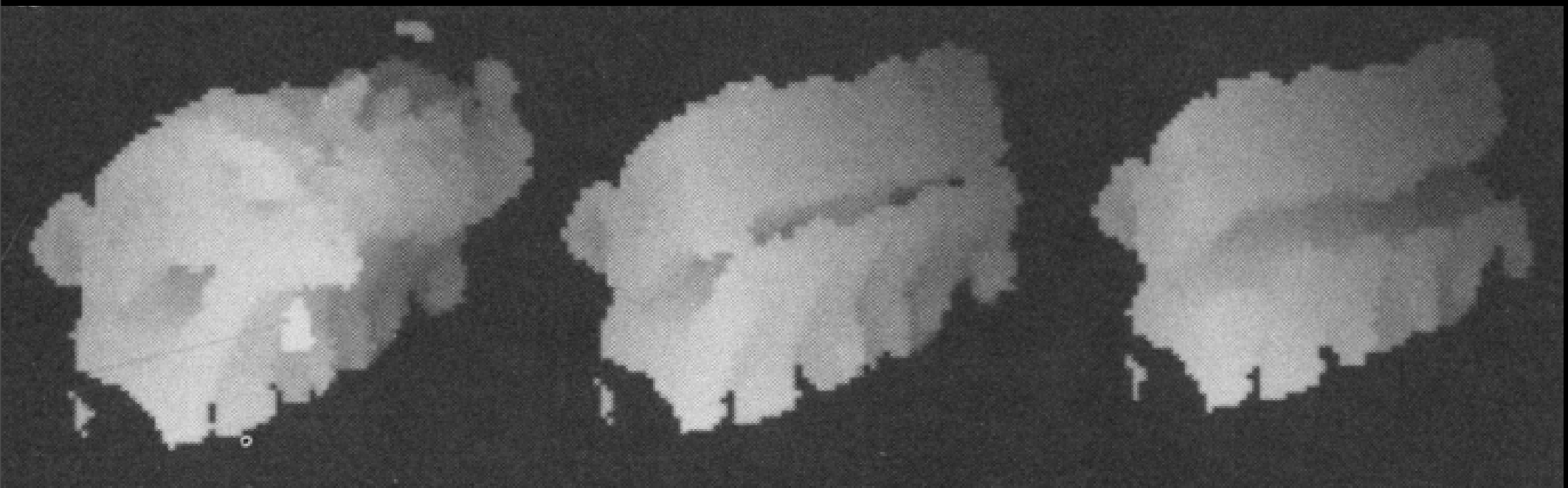
Samuel M. Goldwasser, R. Anthony Reynolds, Ted Bapty, David Baraff, John Summers, David A. Talton, and Ed Walsh
University of Pennsylvania

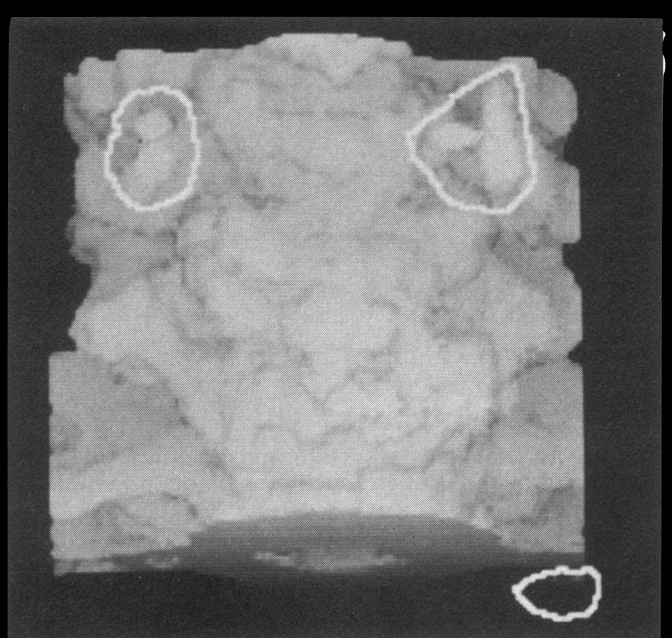
Human spine dataset

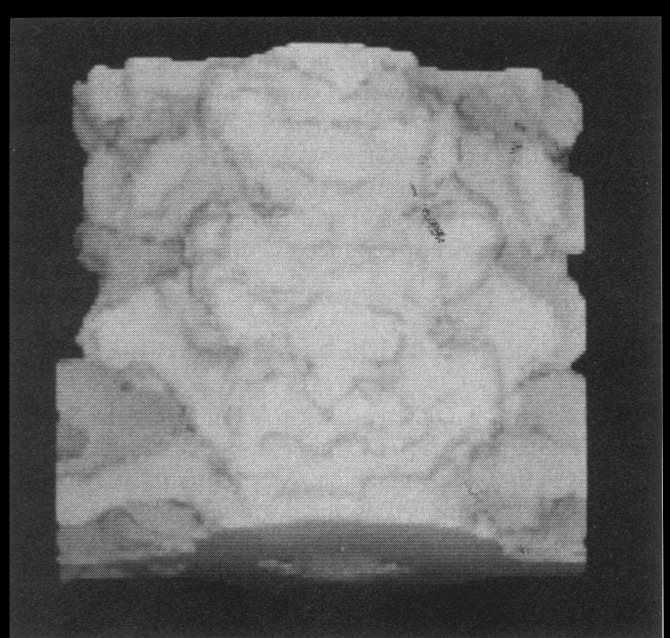
For some applications, rendering a standard static view is sufficient ...

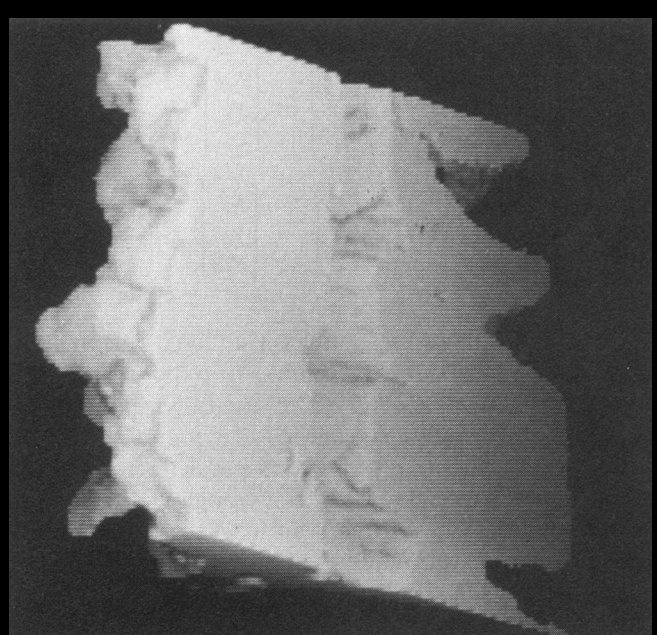
But for planning surgery, physicians want real-time interaction.

Slicing through a human spine @ 30 frames/second using a "virtual knife" (graphics tablet)



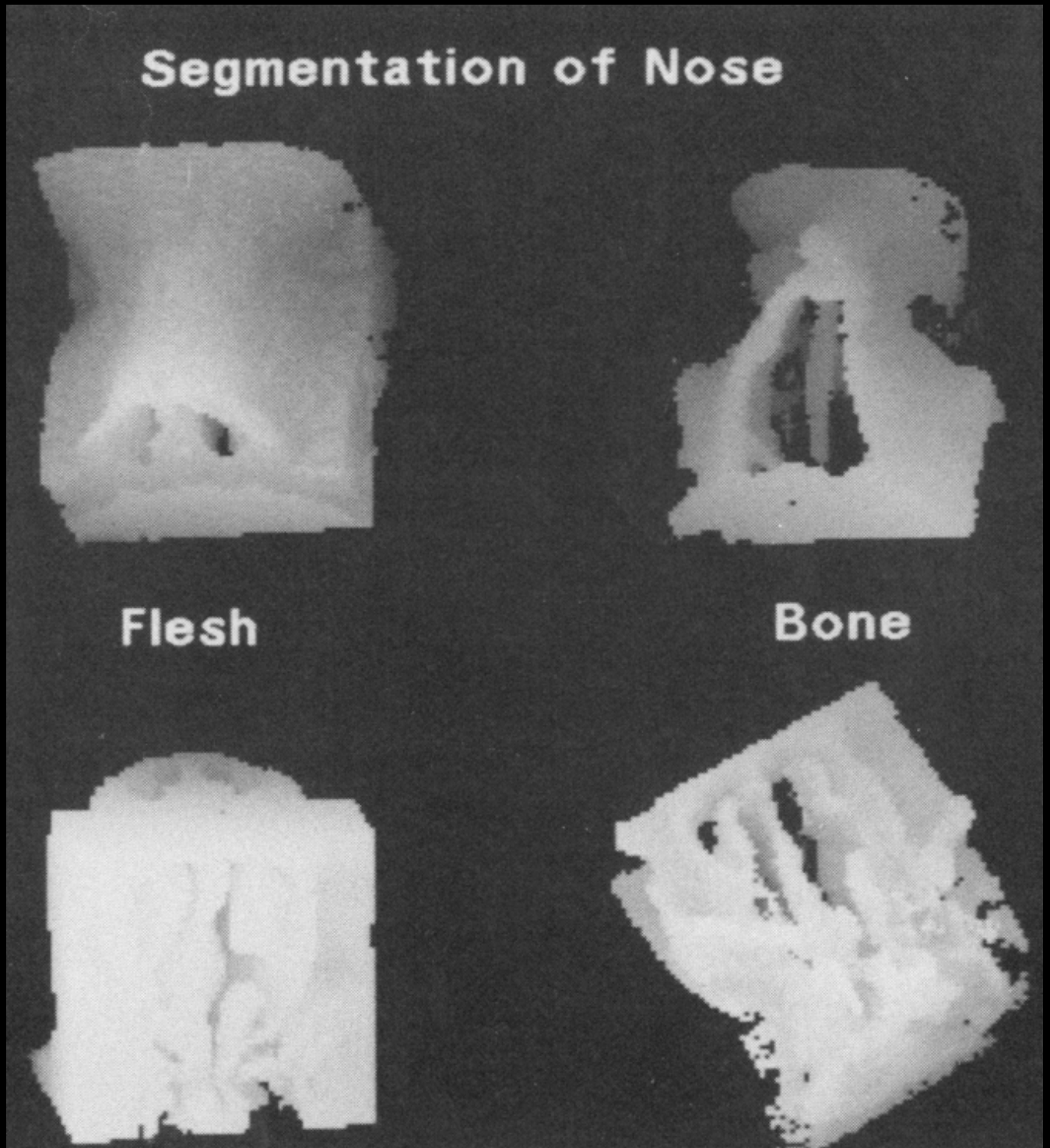




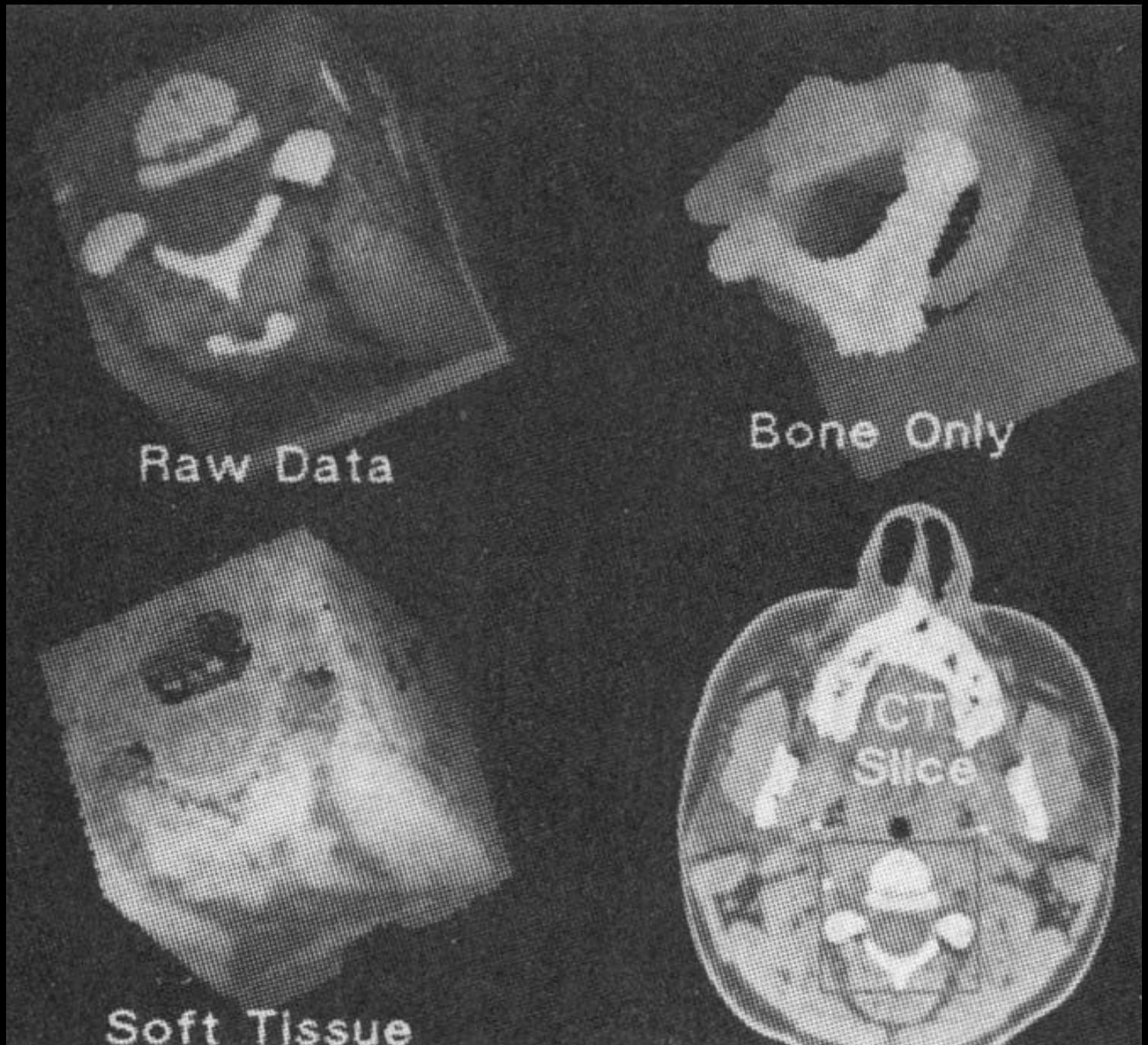


Interactive thresholding to segment the bone from the flesh of a broken nose.

Specify the gray-scale ranges to show or hide with trackball, @ 30 frames/sec.



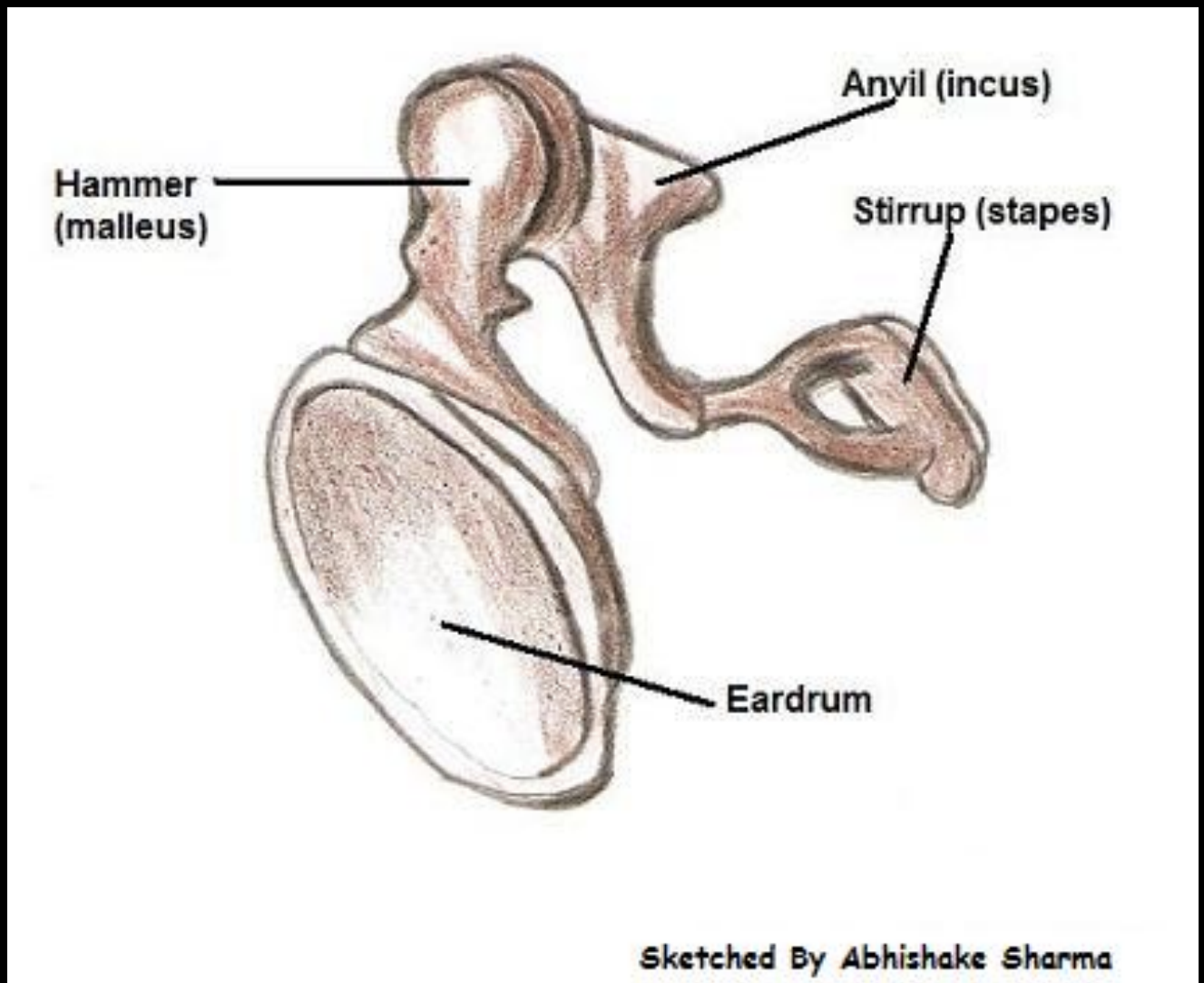
Interactive thresholding to segment the bone from the flesh of the top of the spinal column.



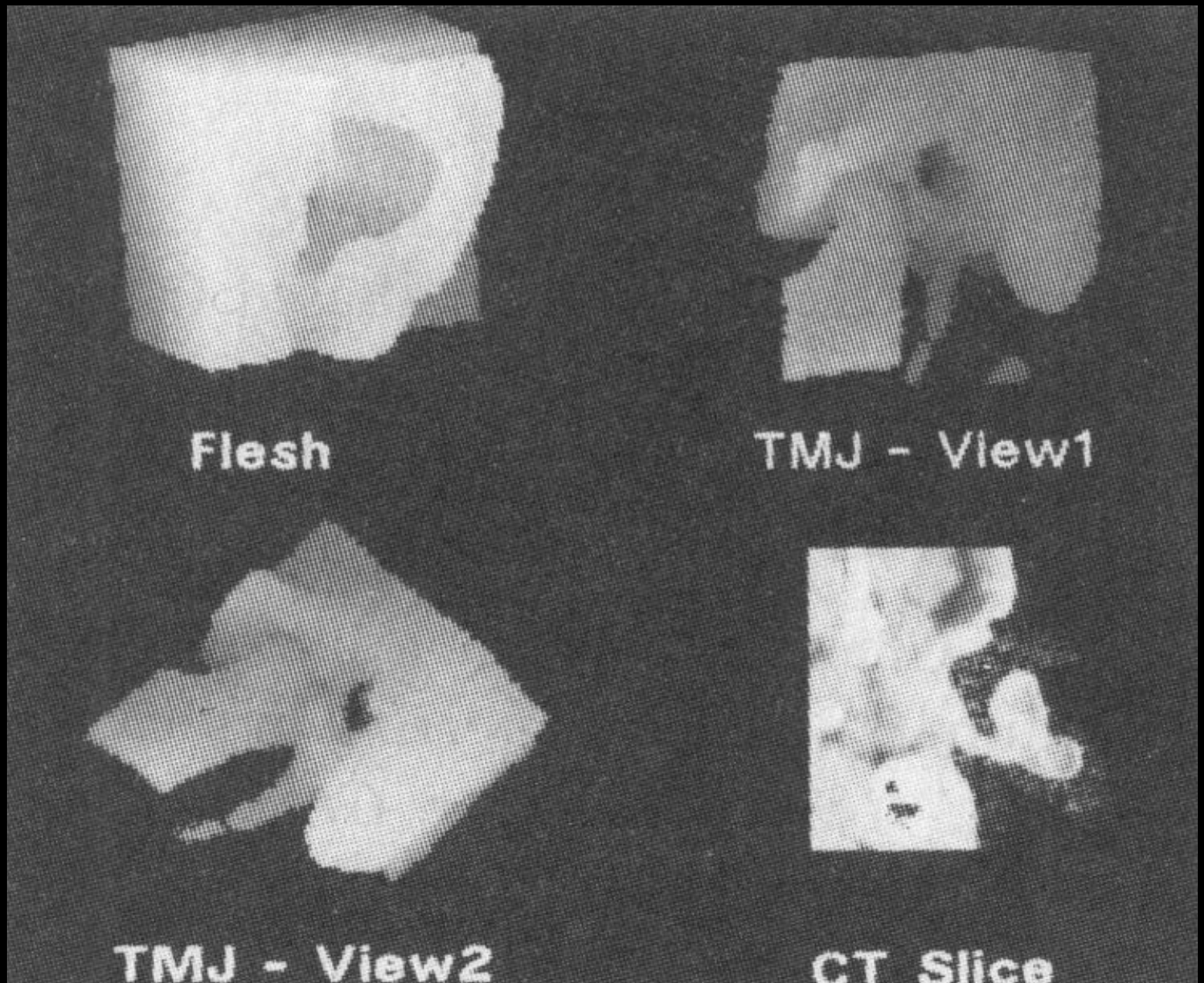
In some babies, these bones are malformed and don't transfer acoustic energy to cochlear.

Microsurgery can fix the problem before the baby is ready to learn to talk.

Tiny bones in the middle ear.

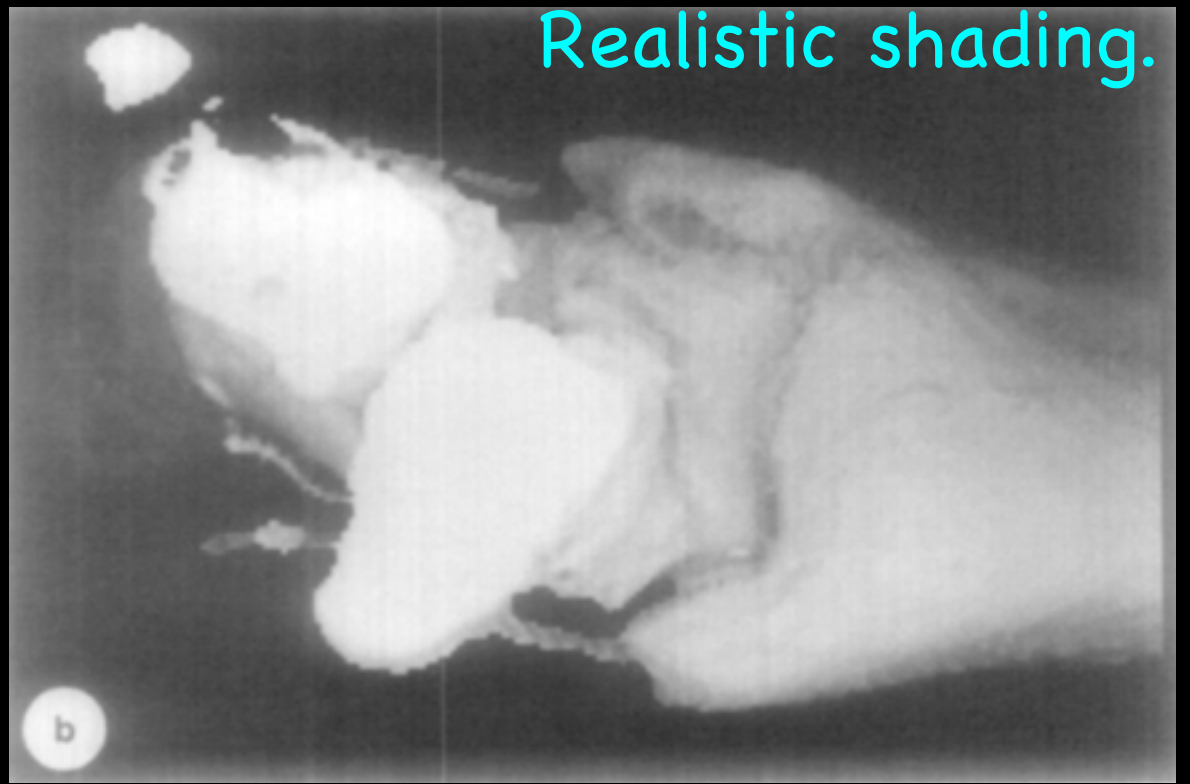


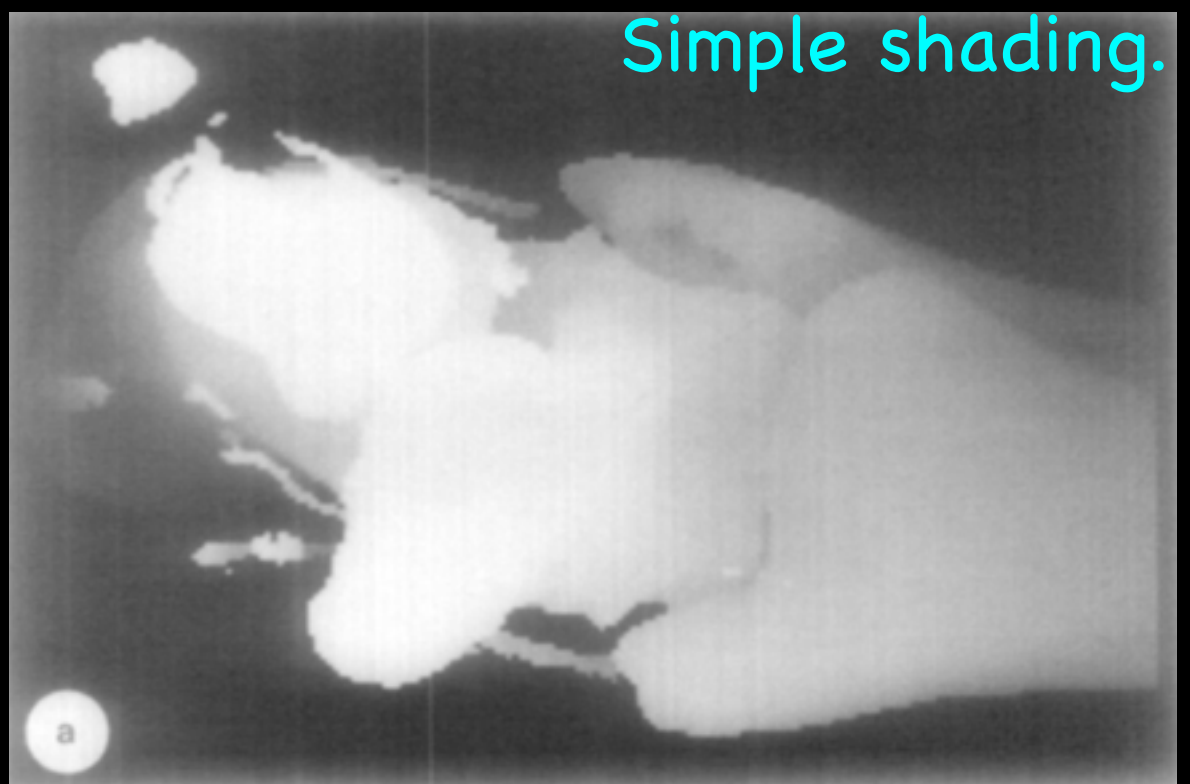
Interactive thresholding to segment the bones of the middle ear.



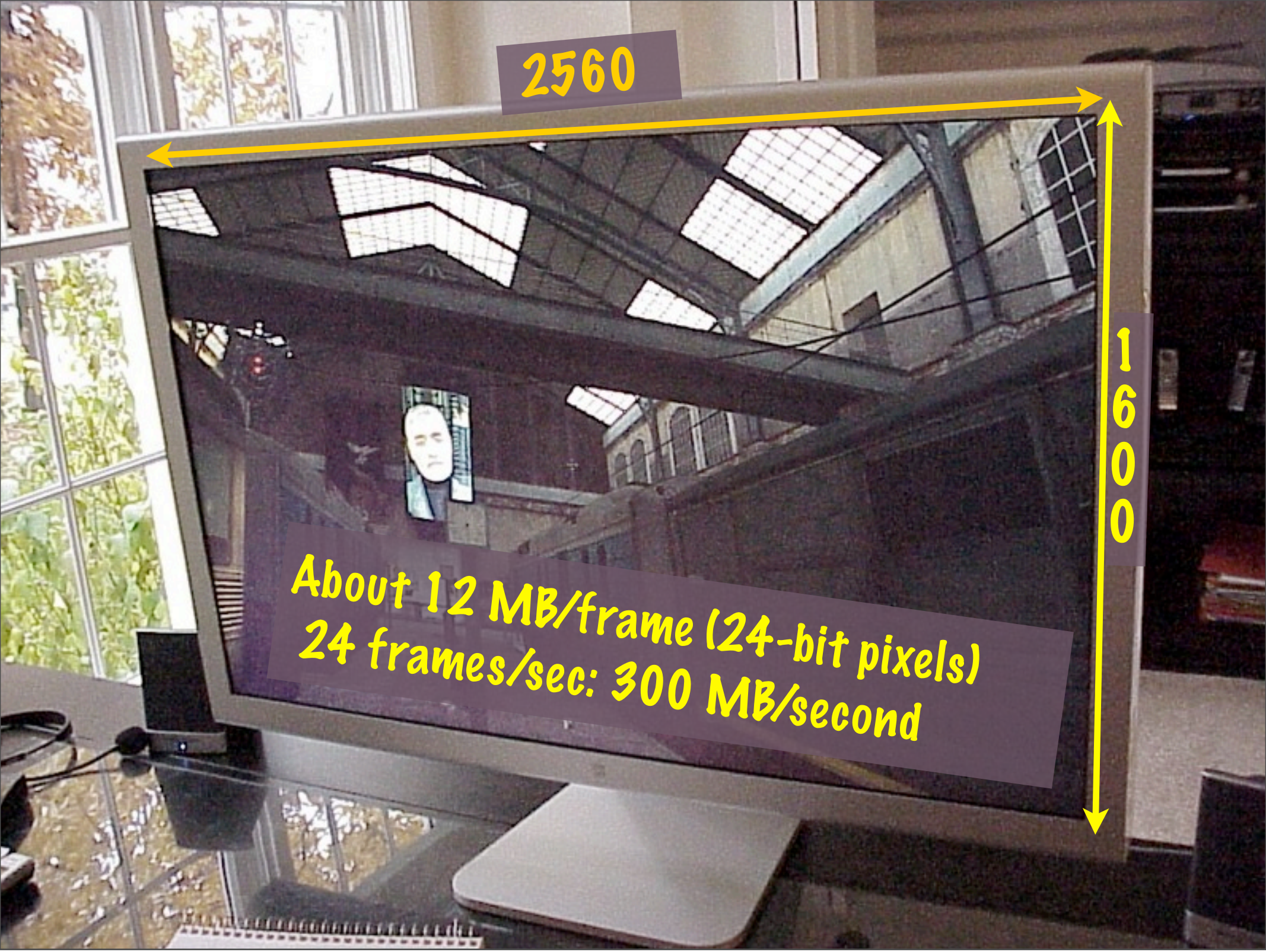
Realistic shading models are applied in real-time.

Essential to let physician see details of bone damage to prepare for surgery.





Algorithms


2560
1600
About 12 MB/frame (24-bit pixels)
24 frames/sec: 300 MB/second

A frame buffer for a normal 2-D display ...

PCIe Bus Port

300 MB/s easy to sustain.

Goal: Virtual knife cuts seen on screen with the 1 / frame-rate latency of 33 milliseconds.

12 MB Frame Buffer A

12 MB Frame Buffer B

Control Logic

DVI Formatter

D/A

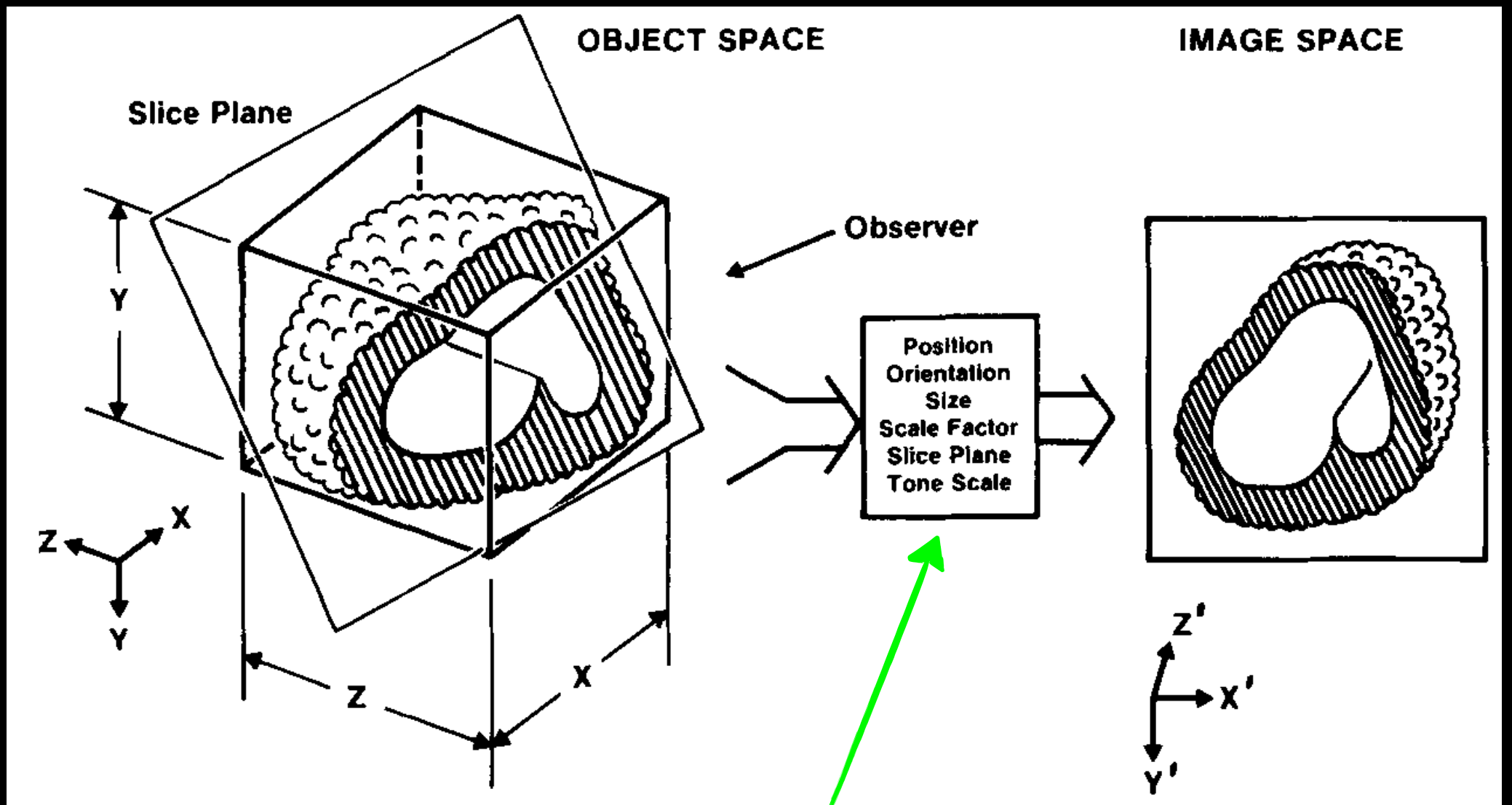
Display Out

Double Buffering:

CPU writes A frame in one buffer.

Control logic sends B frame out of other buffer to display.

30 times a second, read all 256^3 voxels from DRAM.
Do all "compute" on the way to the 2-D frame buffer.



Voxel data, X, Y, Z space.

The transforms to create the view.

Screen image X', Y' Z' space

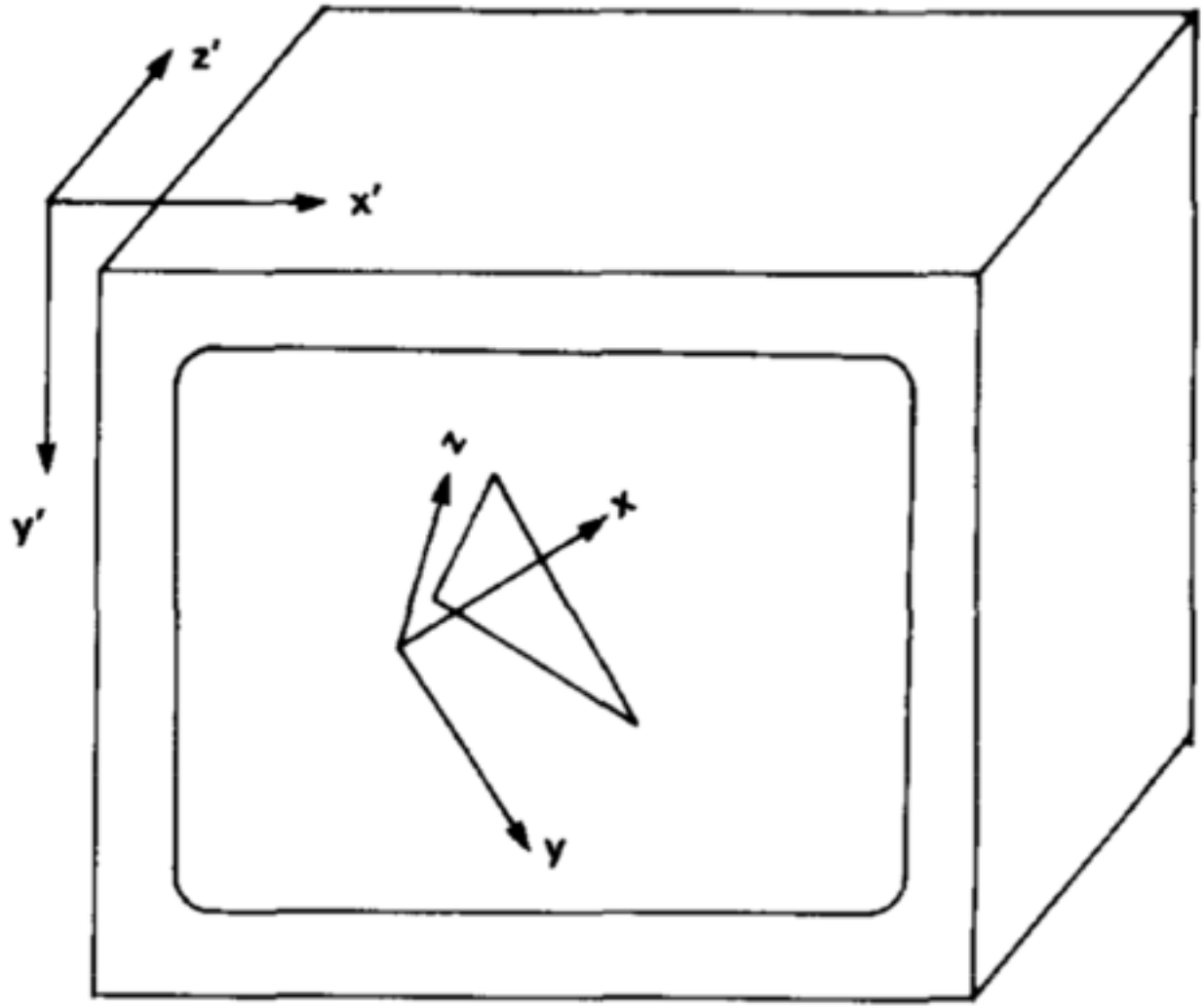


FIG. 1. Object space (x, y, z) and image space (x', y', z').

The user specifies rotation, scaling, and object distance from screen with track ball …

The specification ($R_{\{X',Y',Z'\}}$ and L_0) appears in the object-to-screen matrix equation:

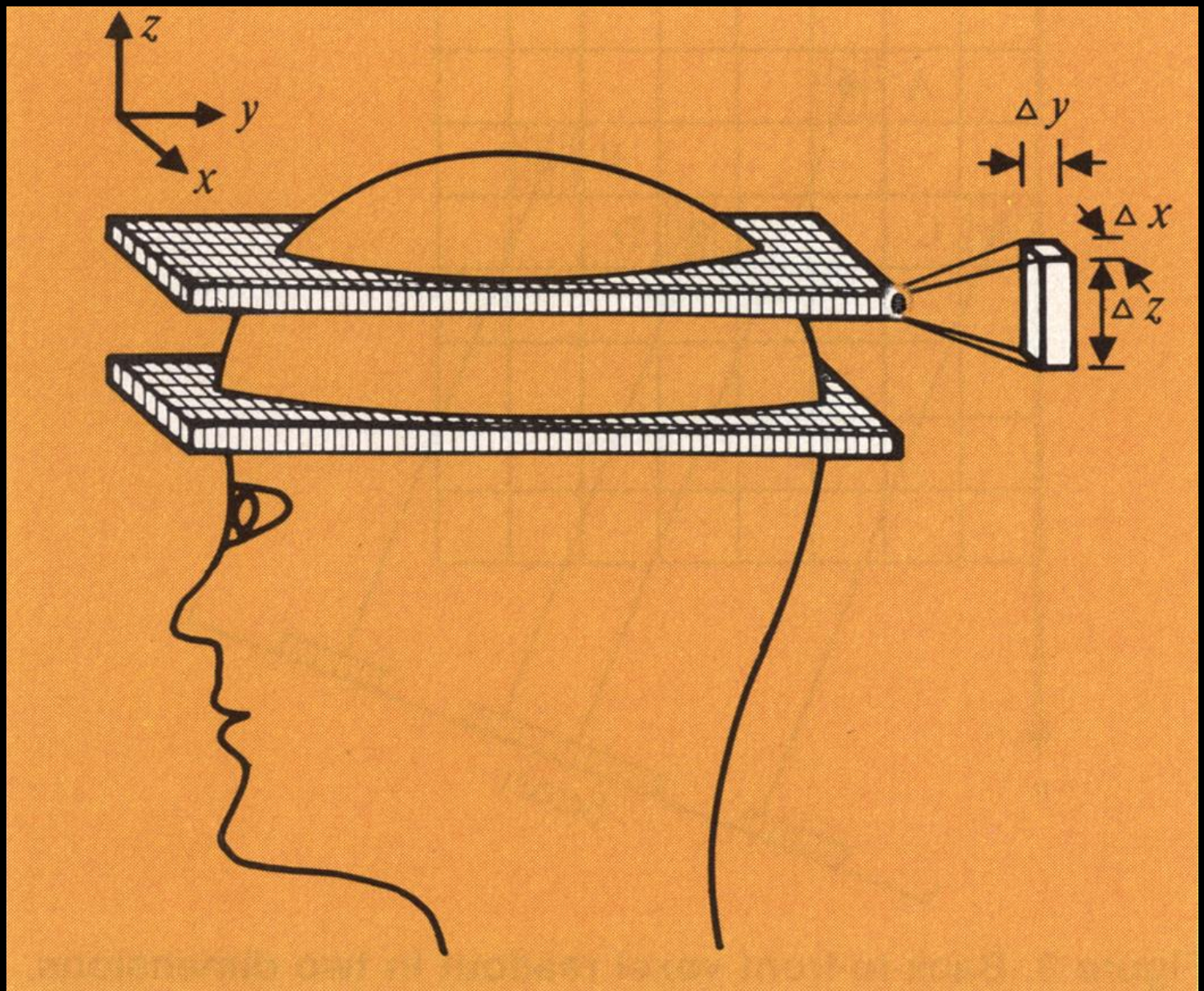
$$\mathbf{C}_k' = R_{Z'} \cdot R_{Y'} \cdot R_{X'} \cdot \mathbf{C}_k + \mathbf{L}_0, \qquad 1 \le k \le 8.$$

C_k: The position of a voxel in the object (X, Y, Z)

C'_k: The pixel to light up (X′, Y′) and the distance of the voxel from the screen (Z′, used for shading).

However, a voxel should not light its pixel if another voxel that lights its pixel is closer to the screen …

Solving this "hidden surface" problem (2-D case)



Algorithm

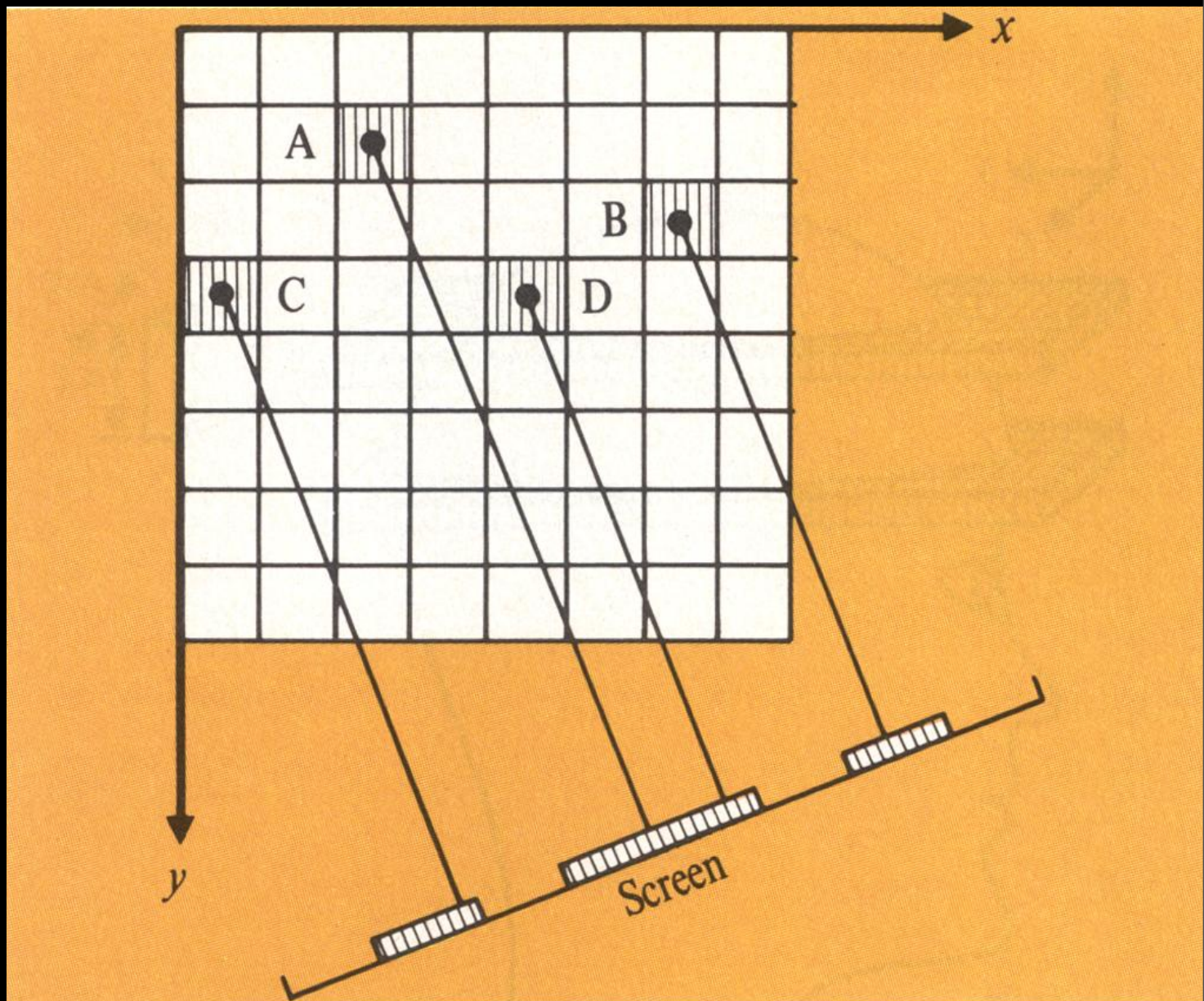
Read voxels from the slice in "Back to Front" order.

Do screen pixel writes for all non-zero voxels.

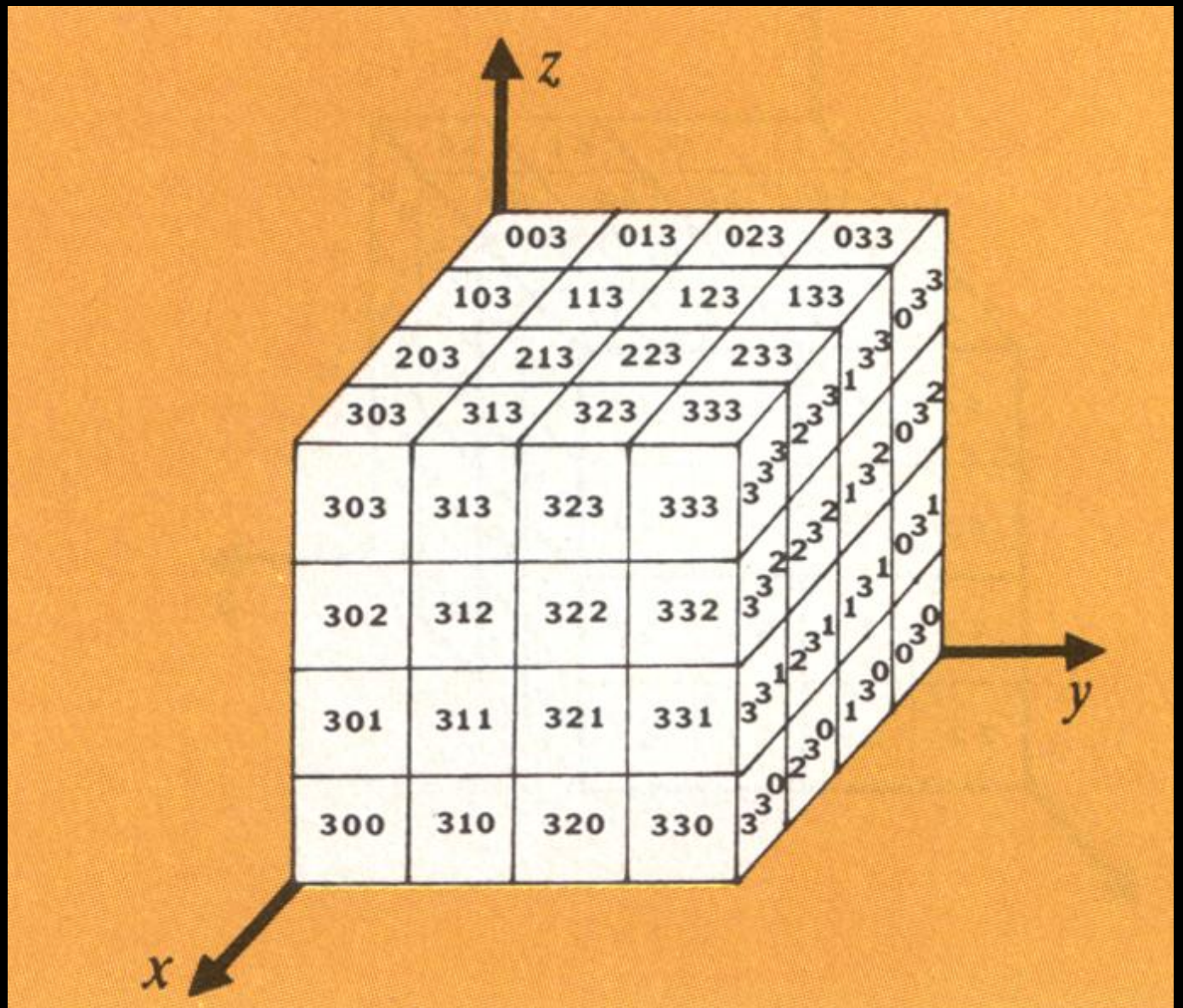
Back: Furthest from screen in X′, Y′, Z′ space.

Front: Closest from screen in X′, Y′, Z′ space.

Why this works: Pixel written by voxels that should be hidden are overwritten by pixel writes by voxels in front of it.



Back-to-Front Algorithm extends to cubes …



Cubes numbered in order of readout …

And works for sub-cubes --> parallel hardware

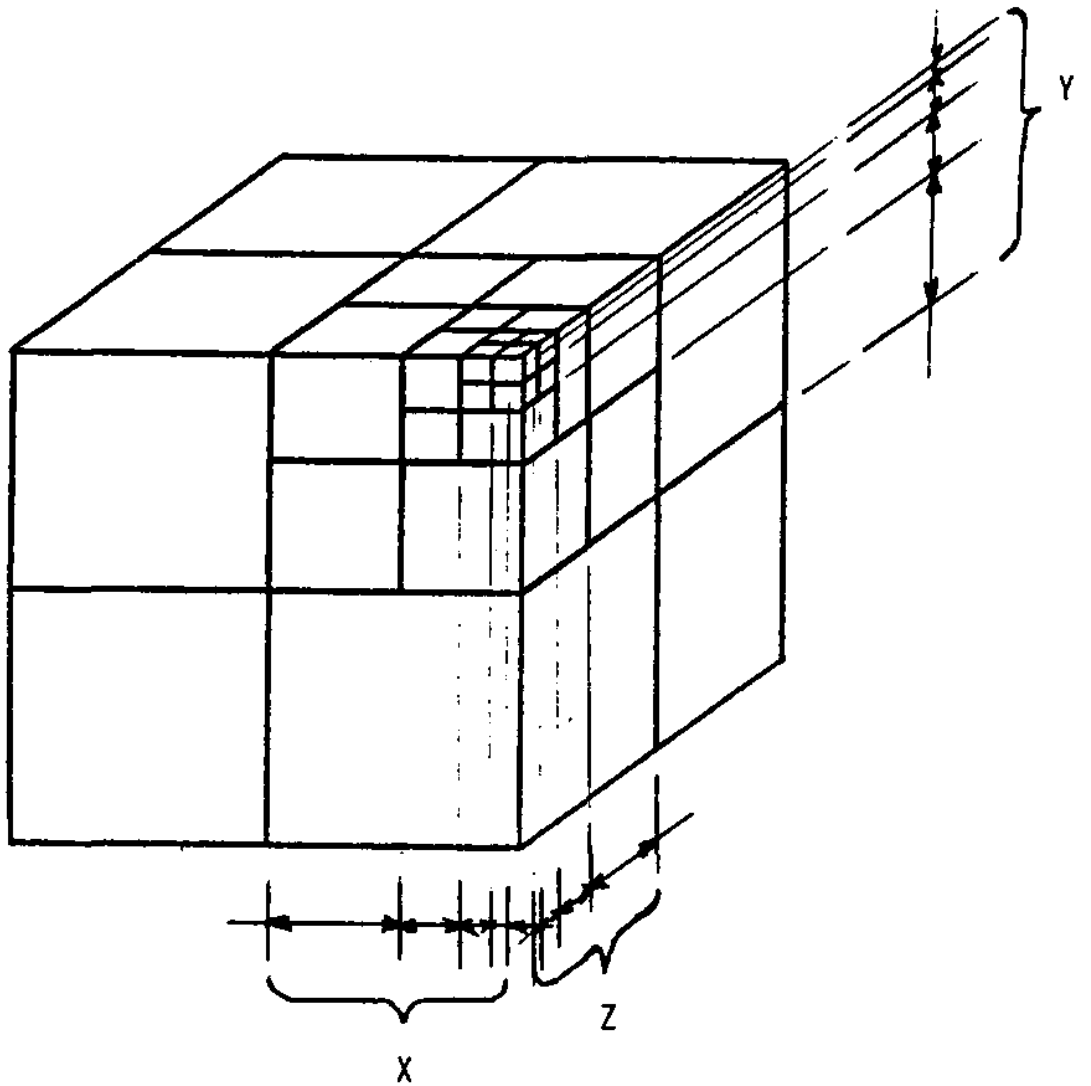
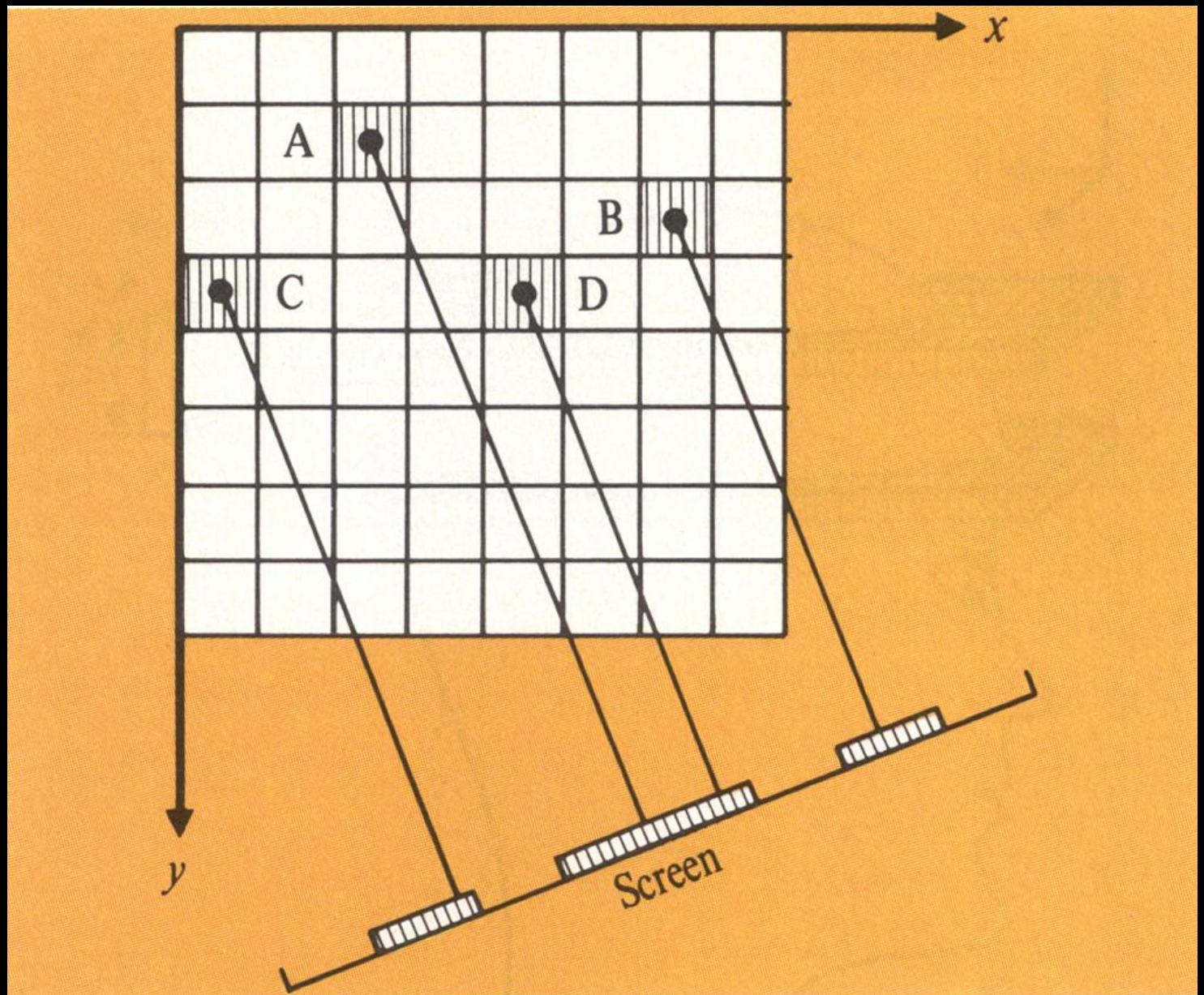


Figure 5 - Recursive Decomposition and Position Computation

Largest sub-cubes can be read out in parallel to create "mini-screen images" which are merged in priority order.

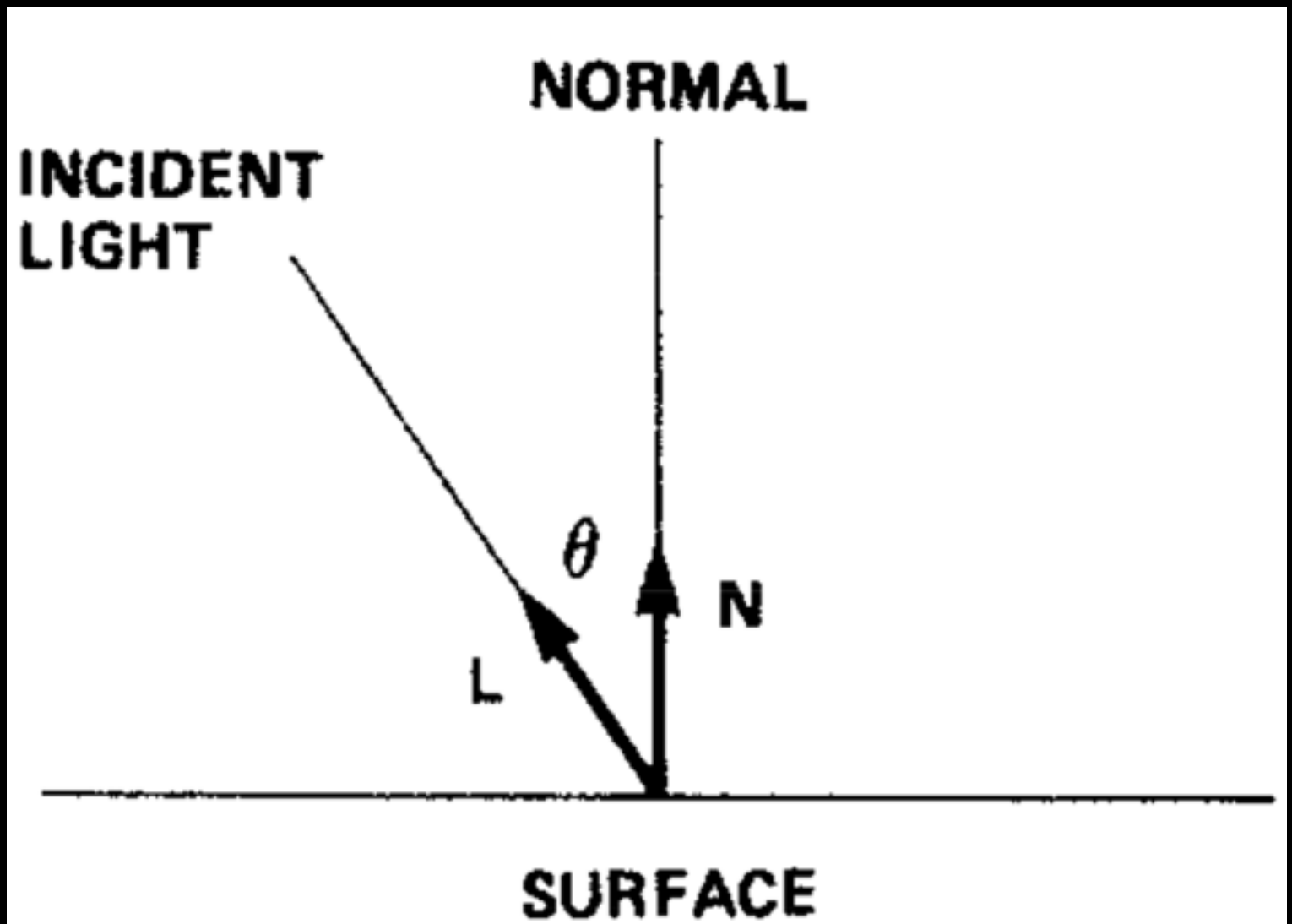
Shading

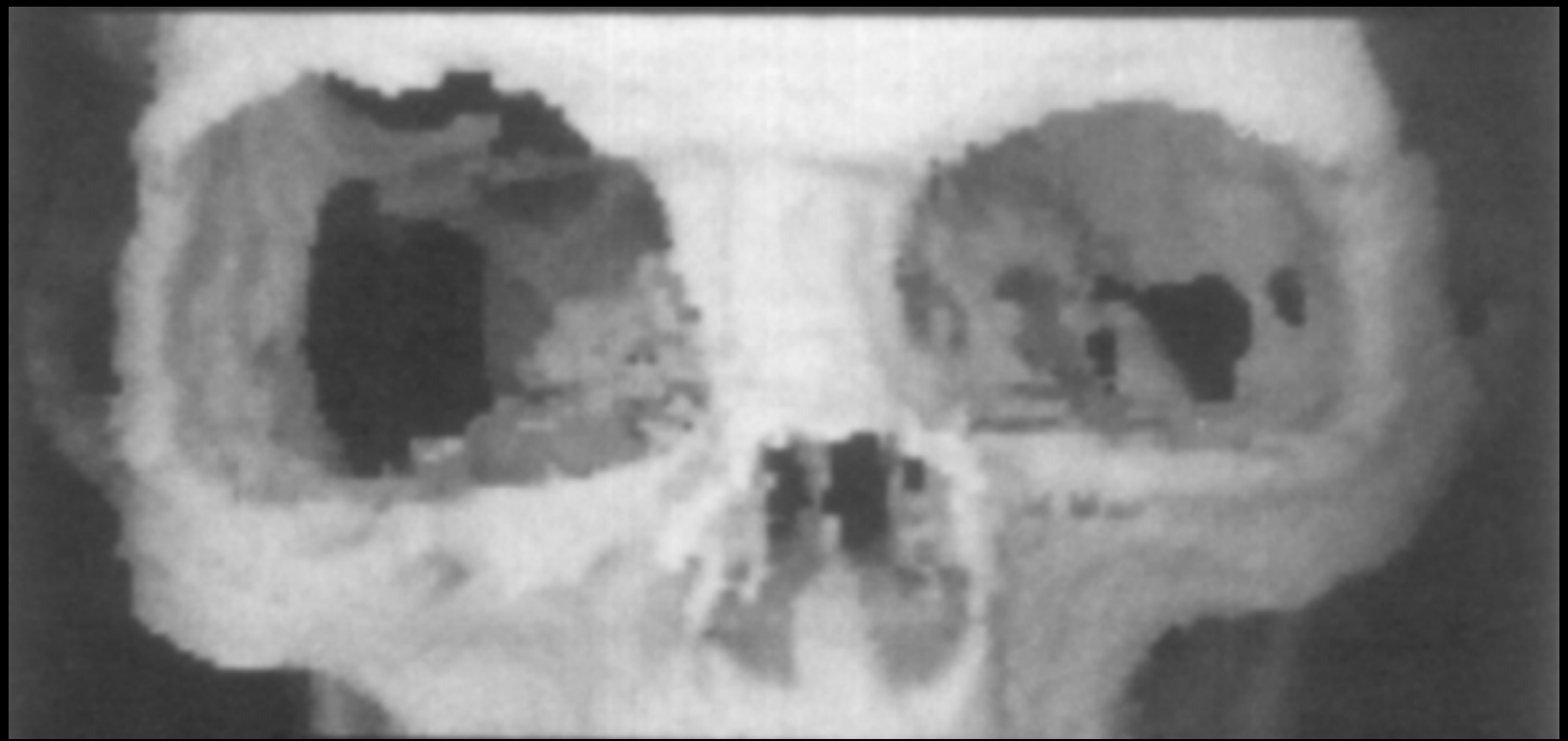
(1) Distance of voxel to screen "shades" pixel.
(2) Curvature near voxel also "shades" pixel.



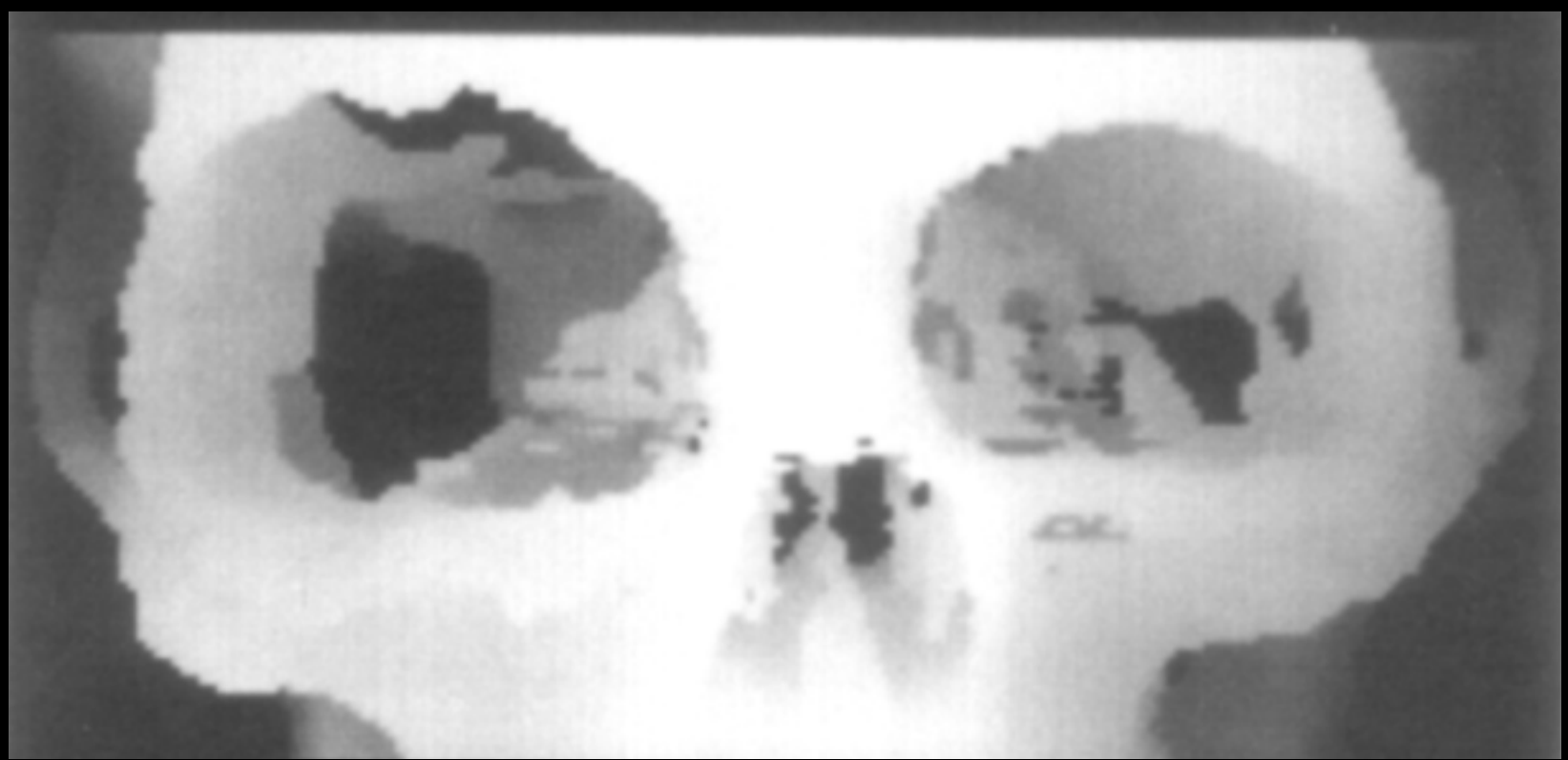
Shading

(1) L derived from distance from screen.
(2) Distances of neighboring voxels sets N.
(3) Darken pixel as a function of cos(θ)





cos(θ) shading.



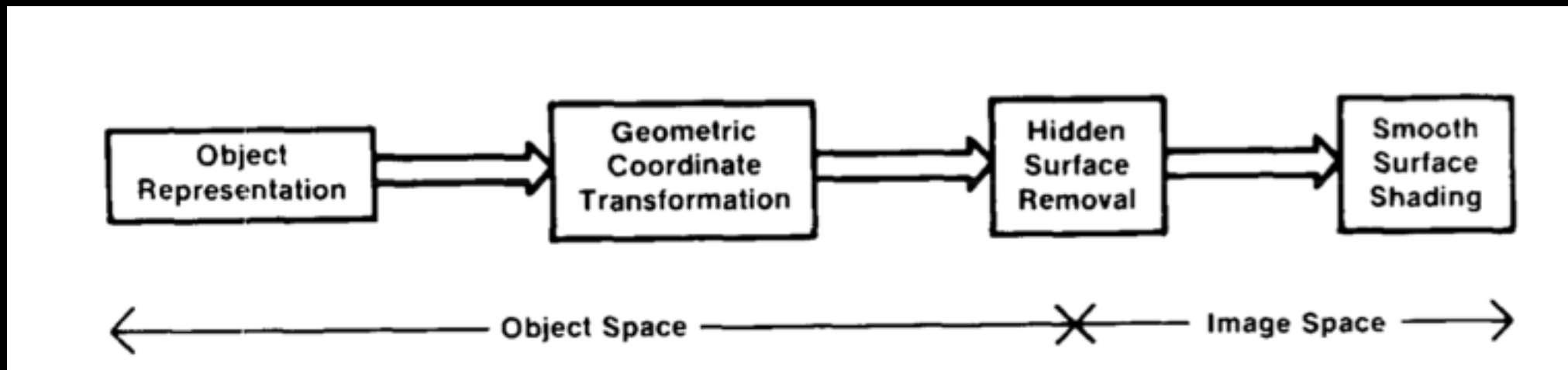
Simple shading.

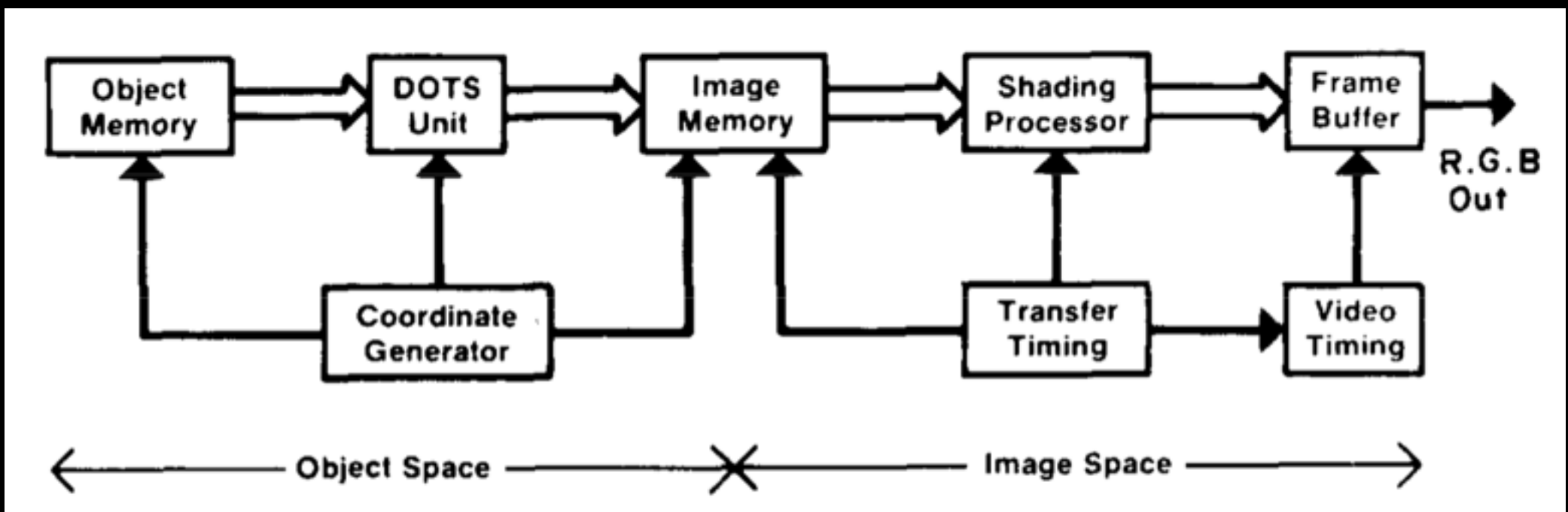
Break

Play:

Voxel Processor Design

Top-level block diagrams





64 PE-Mems: Each renders a sub-cube into a mini-image

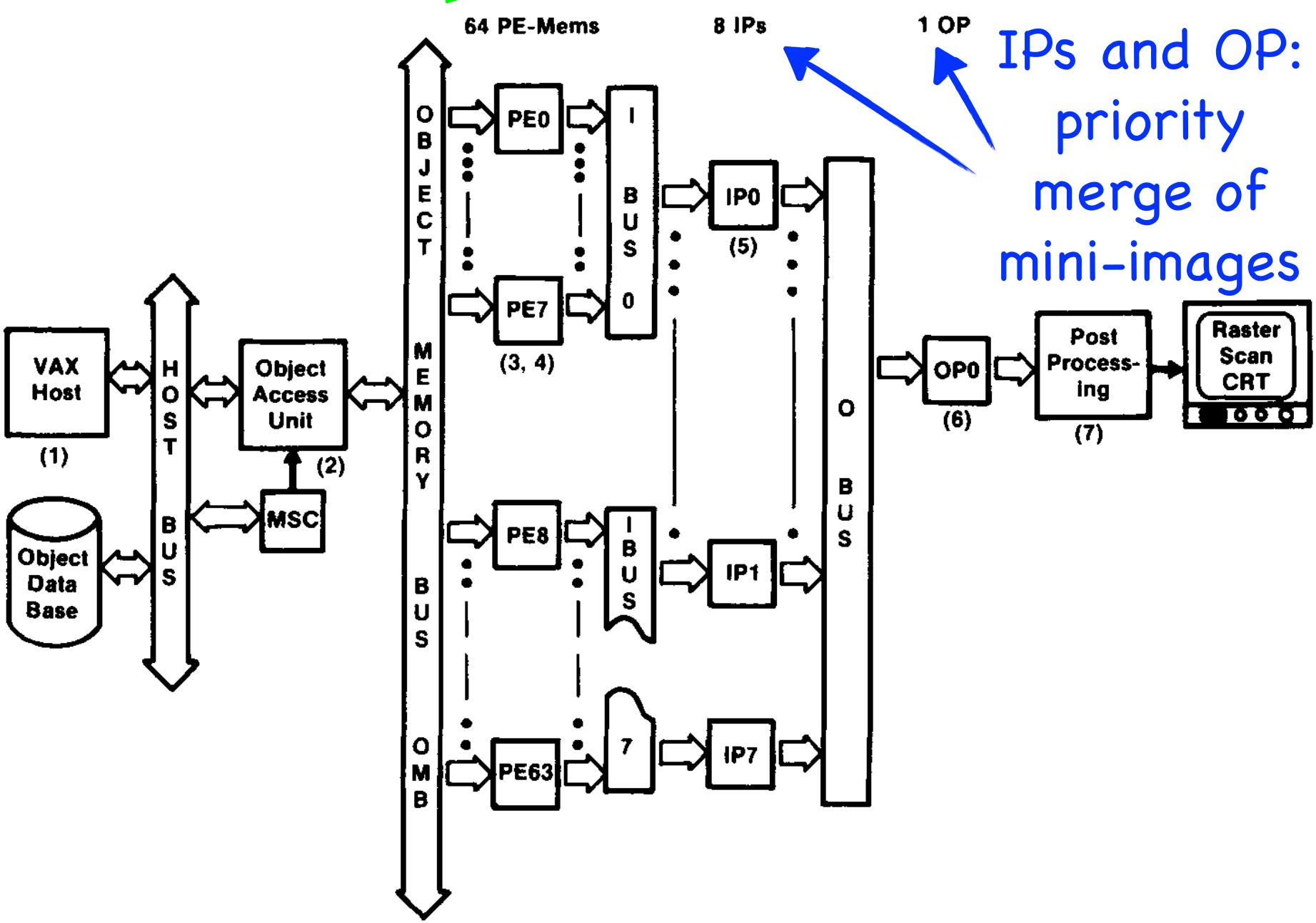
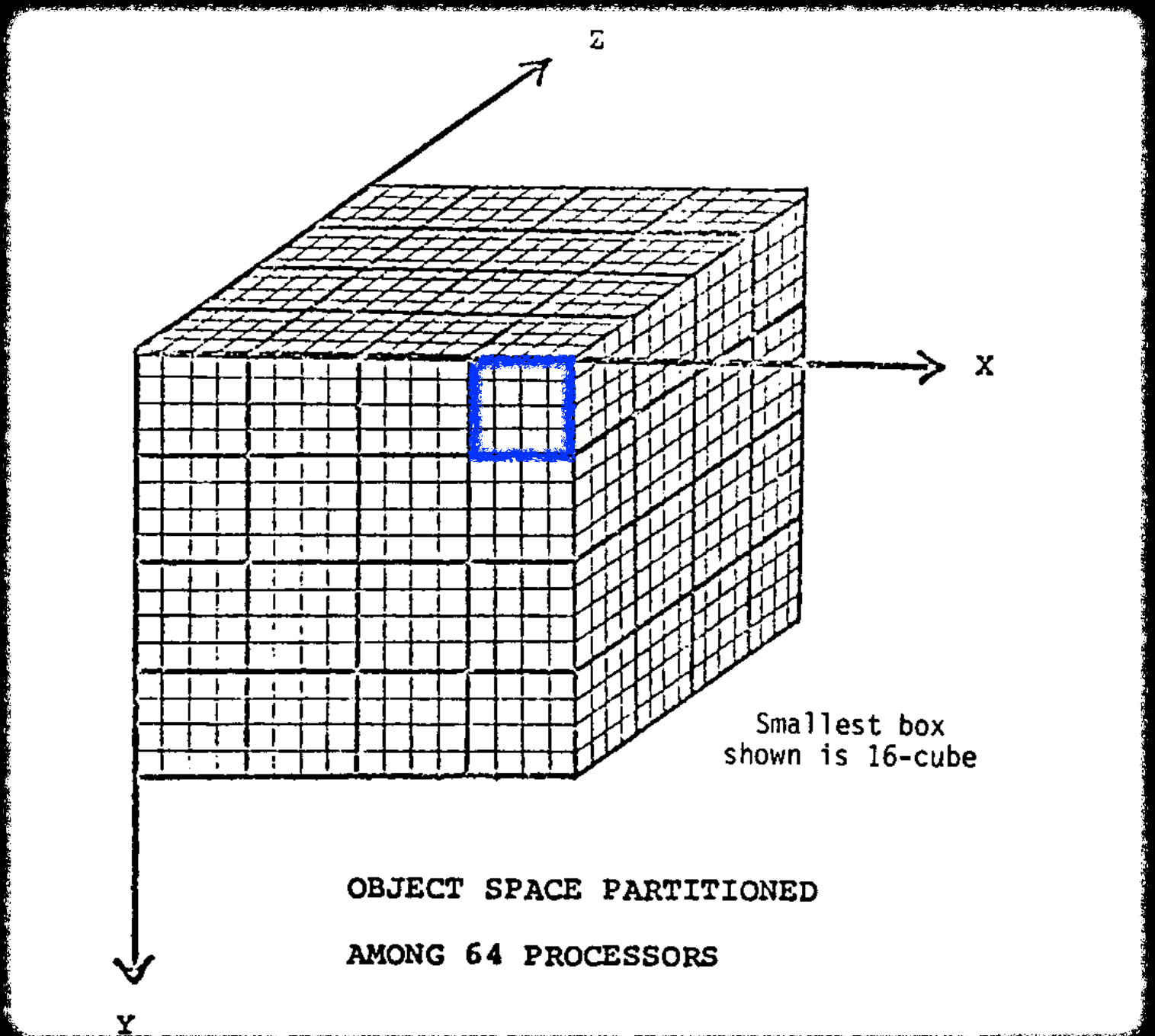


FIG. 5. Voxel Processor overall hardware organization.

64 PE-Mems: Location of processors on 256-cube



PE-Mem design ...

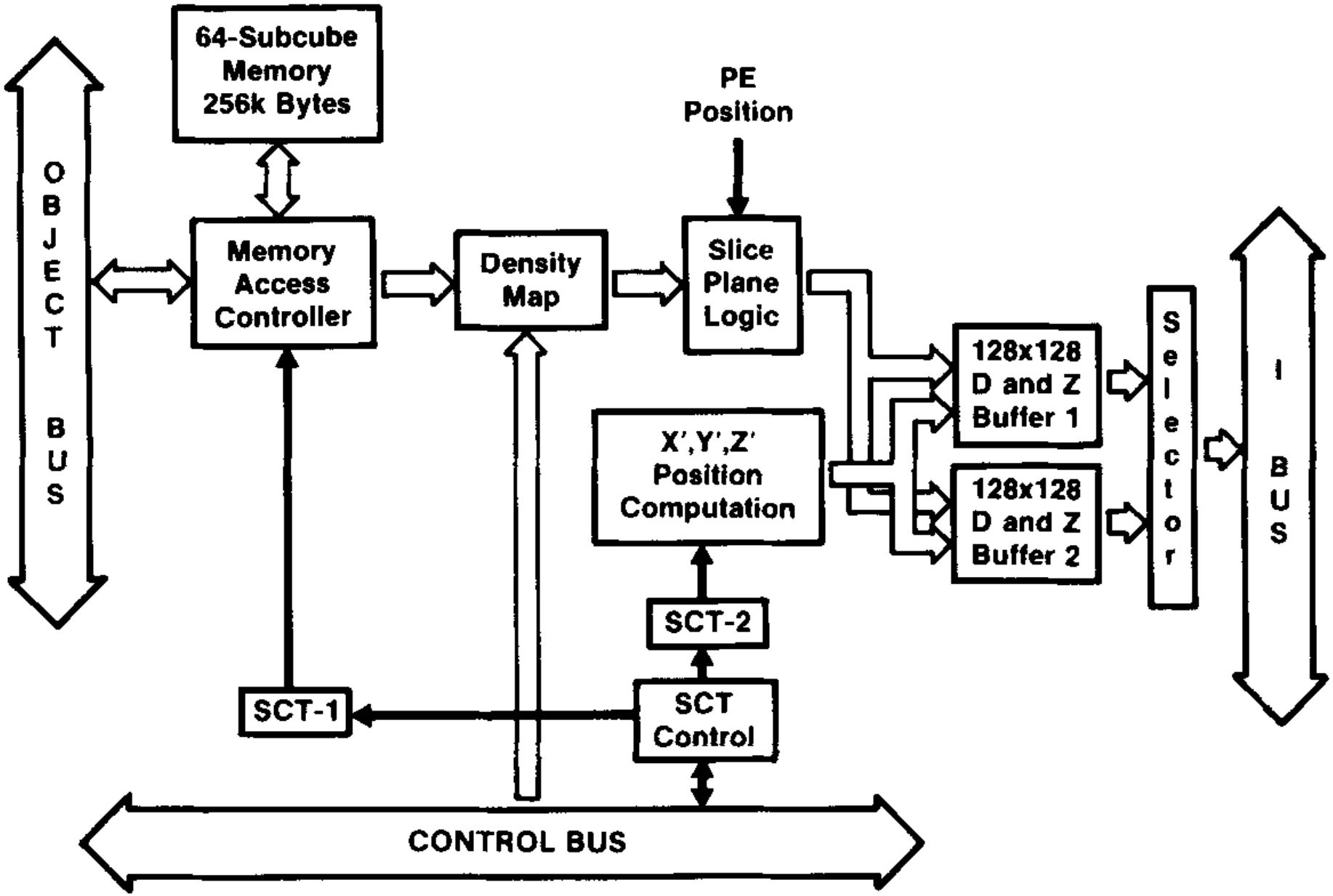


FIG. 8. Processing element and 64-cube memory organization.

Back-to-front cube indexing. (X', Y', Z') calculation.

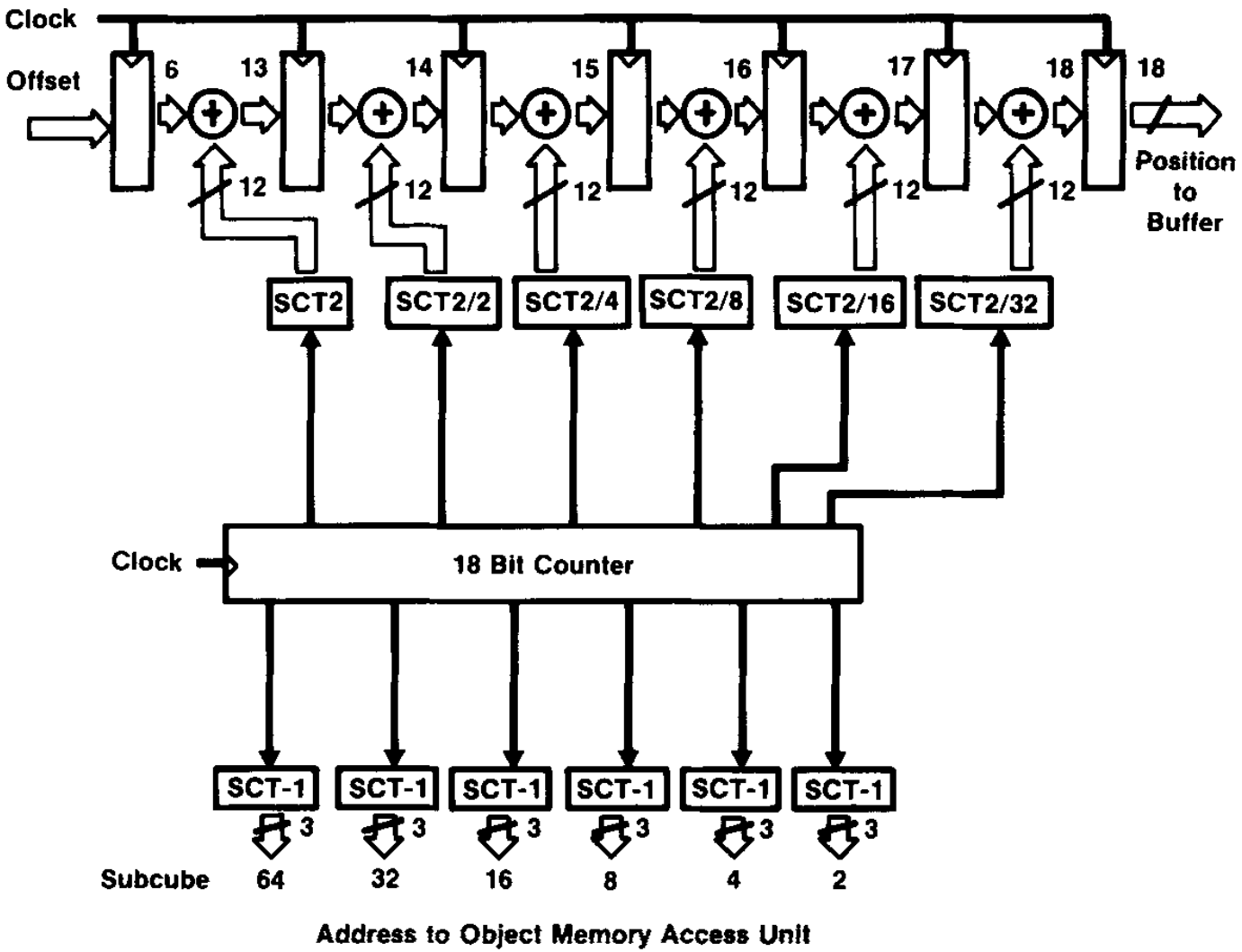


FIG. 9. Arithmetic pipelines for X, Y, Z position computations. Object Memory (voxel) addresses are generated from SCT1; Image Memory (pixel) addresses are generated from SCT2. All the object memory address lines are shown, but only the X arithmetic pipeline is shown. The Y and Z pipelines are identical to X.

Priority merging of 8 PE mini-images ...

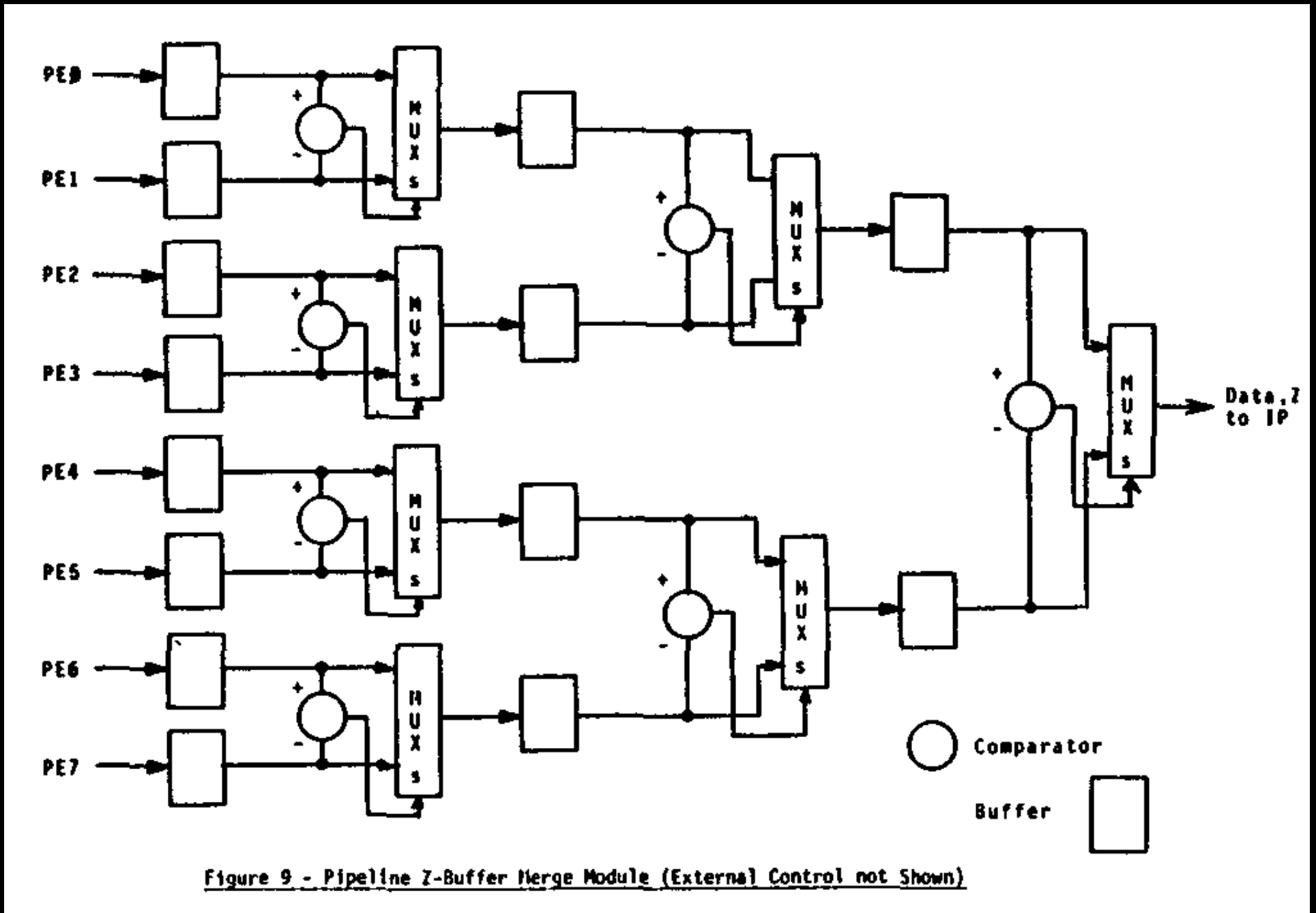
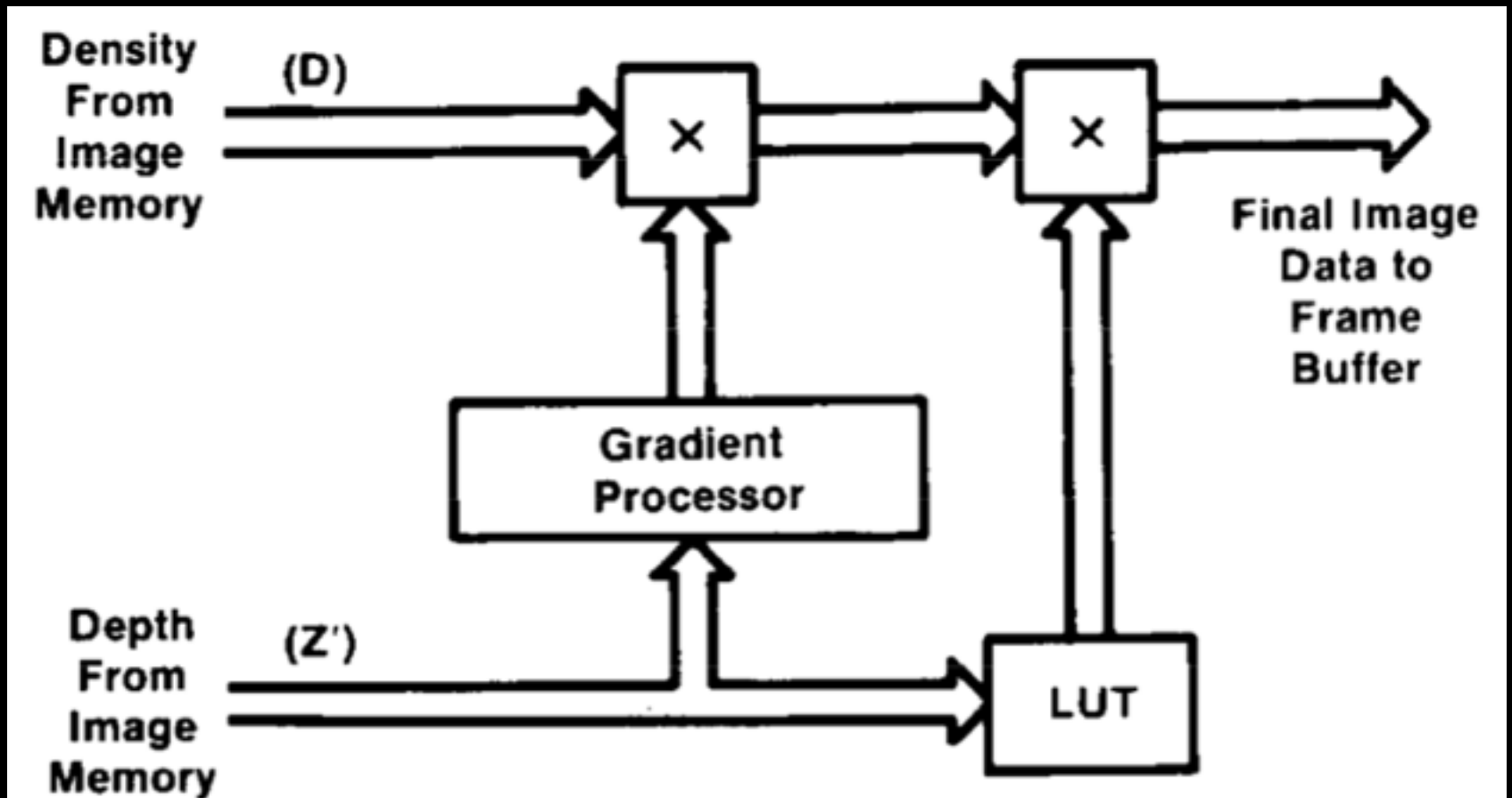


Figure 9 - Pipeline Z-Buffer Merge Module (External Control not Shown)

Shading ...


Density
From
Image
Memory
(D)
×
×
Final Image
Data to
Frame
Buffer
Gradient
Processor
Depth
From
Image
Memory
(Z')
LUT

Memory …

To store and display any set of objects within a 256-cube object space requires 16M bytes of high speed RAM (assuming 8 bit quantization for each point). While this may seem to be an extremely large amount of high speed memory, it should be recognized that the steep decline in MOS memory prices is expected to continue for some time. Even at current prices, the cost of the overall display device (which is dominated by the cost of this memory) should be relatively small compared to the cost of a complete medical imaging system such as a CT scanner.

Since the object space is divided into 64 equal subcubes, each PE requires 256K bytes of associated memory. To keep the individual processors busy, we assume a memory bandwidth of 100 nsec/read access. With 256K MOS dynamic RAMs which have static column access capability, this can be accomplished using only eight devices for each 64-cube. Such DRAMs require only one full access time to load their internal register with an entire row (512 bits). Subsequently, any bit in the entire row can be randomly accessed rapidly. Conveniently, one "bit-hyperplane" (one bit per cube for an entire 8-cube) can be stored in each row. Some additional logic would be needed to meet the DRAM refresh requirements and the host would be locked out from accesses to the object memory while image generation is in progress.

Timing ... 3 frames of latency, due to pipelining.

1. The time required by each PE to generate a subimage from the 64-subcube is 256K × 100 ns or approximately 25.6 msec.

2. The time required to merge groups of 8 subimages into a 256 × 256 intermediate buffer is 8 × 12321 × 100 nsec or approximately 10 msec.

3. The time required to merge 8 intermediate buffers into the output buffer is 8 × 49284 × 100 nsec or 39.4 msec.

Thus, the limiting time is the last, corresponding to a frame update rate of 1000/39.4 or approximately 25 frames/second. Note that because of the pipeline latency, however, a response to a change in orientation will require a total of three frame times to become visible.

What happened to this product?

- Company was bought by a medical imaging scanner company (Picker) who incorporated it into the product line through the 2000s. Picker was acquired by Philips.

- 256-cubes became possible to do in real time, and the interest in larger cubes (which would still require custom hardware) was insufficient.

- I believe there is still an opportunity to build machines of this nature, if one finds real-time applications for very large voxel datasets.

On Thursday

The last "regular" lecture ...

or, perhaps a different topic.

Th 4/24	Programming to the Machine

Have fun in section !