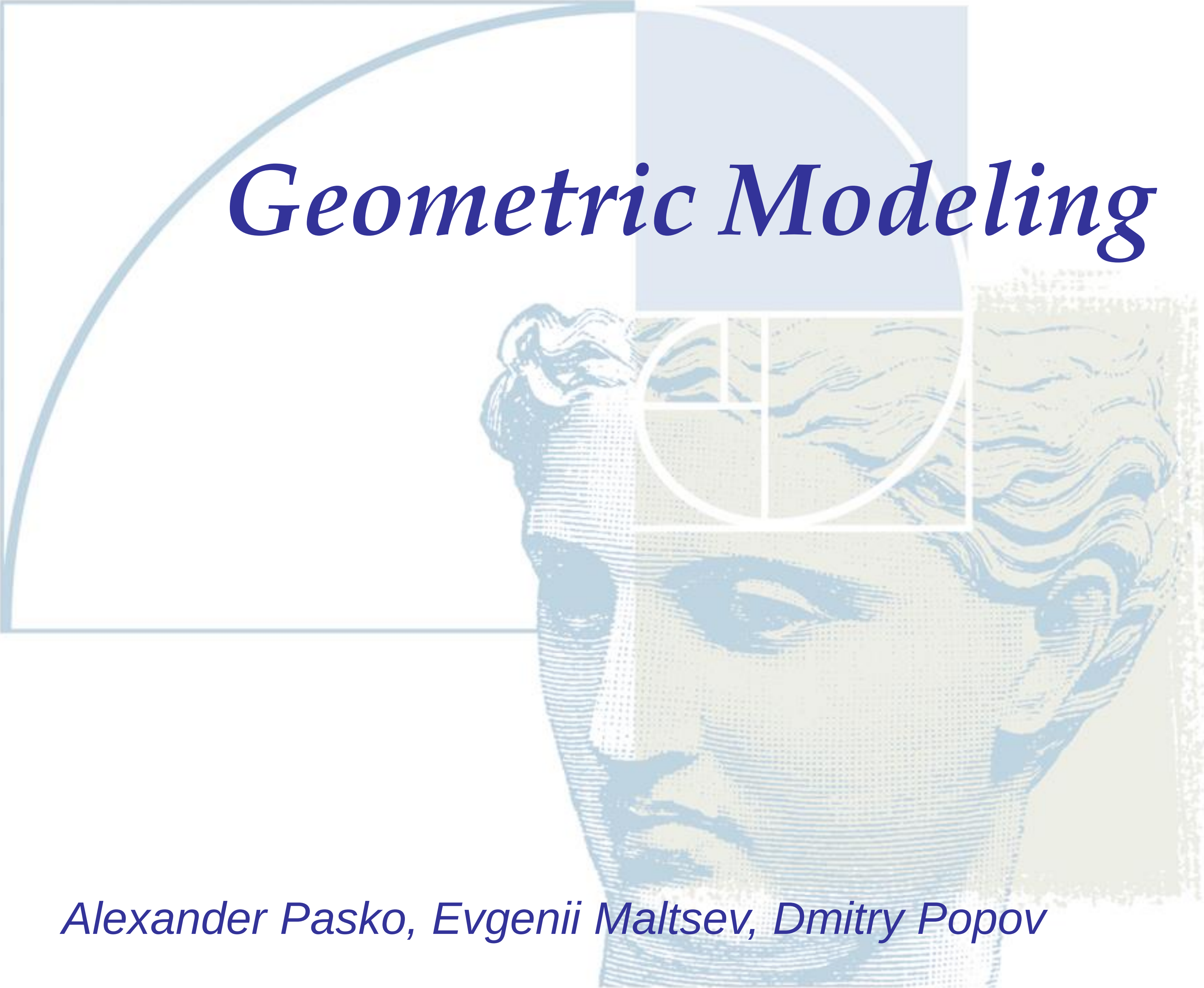
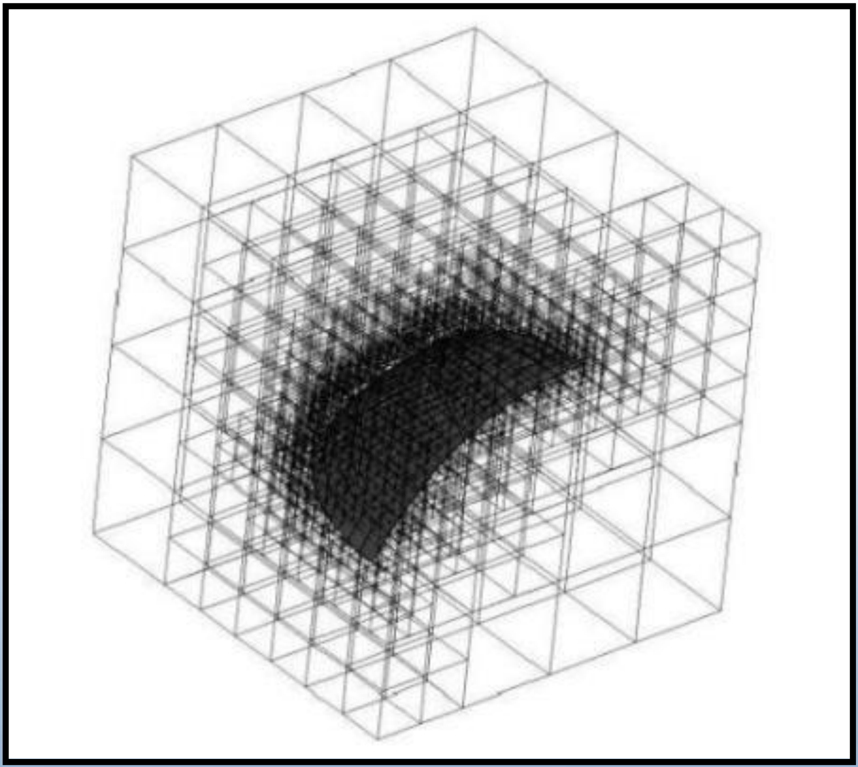
Geometric Modeling



Alexander Pasko, Evgenii Maltsev, Dmitry Popov

Solid Modelling

Spatial partitioning schemes



Tobias Weinzierl

Contents

1. Classification of spatial-partitioning schemes
2. Cell decomposition
3. Spatial-occupancy enumeration
4. Volumetric (voxel) objects
5. Quadtrees and octrees
6. Binary space-partitioning (BSP) trees
7. Ray representation

Classification of spatial-partitioning schemes

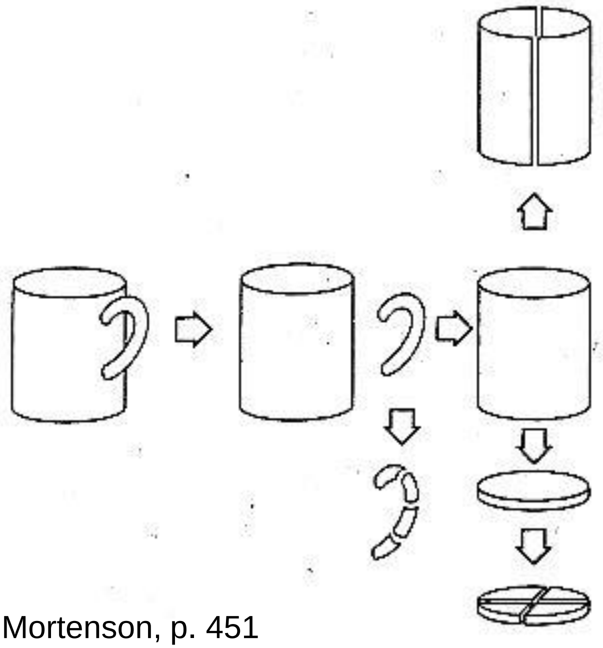
In spatial-partitioning (spatial decomposition) representations, a solid is decomposed into a collection of adjoining, non-intersecting solids that are more primitive than the original solid. The primitives may vary in type, size, position, parametrization, and orientation.

Classification of spatial-partitioning schemes

Primitives	**Representational scheme**
Cells (cube, prism, hyperpatch)	Cell decomposition
Cube	
fixed size	Spatial occupancy enumeration (voxel representation)
variable size	Octrees (quadtrees)
Planar halfspaces	BSP trees
Line segments	Ray-representation

Cell decomposition

An object is decomposed into separate pieces so that each piece of the final decomposition is easier to describe than the original object



Mortenson, p. 451

- Cells are primitive solids (block, prism, cylinder) or parametric tricubic solids (hyperpatches).
- Cells do not intersect and share vertex, edge or face.
- A solid is represented as a union of cells.

Cell decomposition: hyperpatches

A hyperpatch (parametric solid) is a set of points with coordinates given by continuous, three-parameter, single-valued mathematical functions of the form:

$$x = x(u,v,w)$$
$$y = y(u,v,w)$$
$$z = z(u,v,w)$$

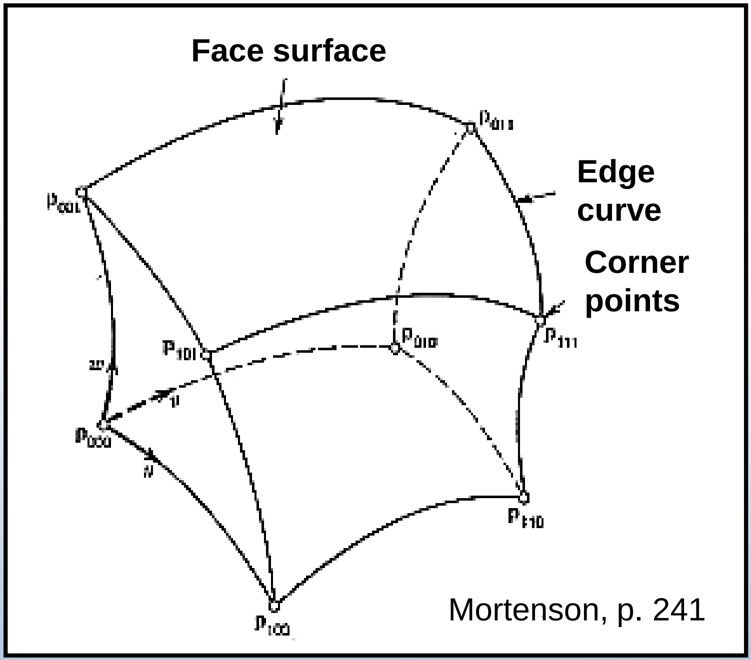
where $u, v, w \in [0,1]$.

Cell decomposition: hyperpatches

The algebraic form of the tricubic solid is given by the following polynomial equation:

$$p(u,v,w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} a_{ijk} u^i v^j w^k$$

$$\text{where } u, v, w \in [0, 1]$$



For x variable we obtain:

$$x(u,v,w) = a_{333x} u^3 v^3 w^3 + a_{332x} u^3 v^3 w^2 + \ldots + a_{000x}$$

Cell decomposition: hyperpatches

Composite hyperpatch solids are constructed of several hyperpatches with required continuity conditions:

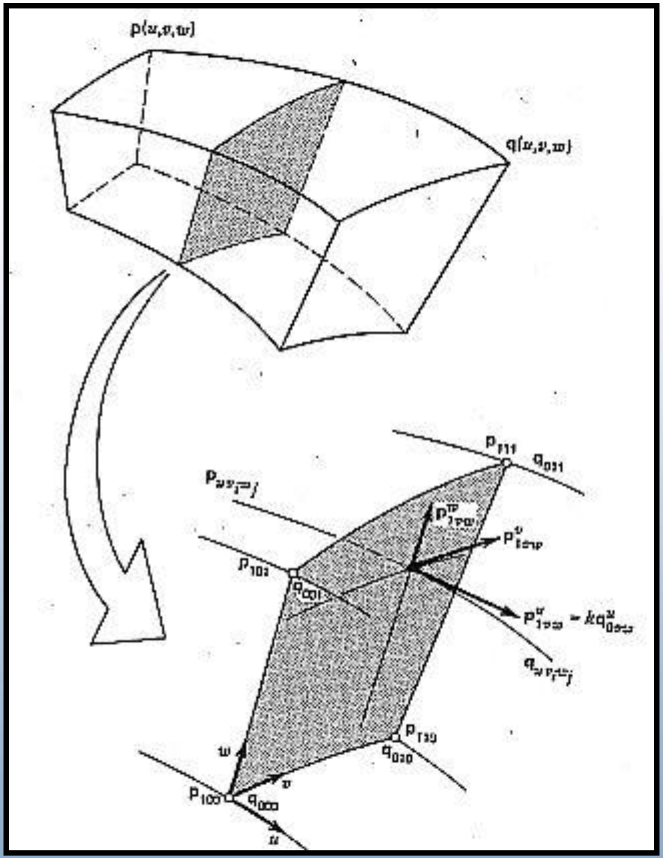
For two solids $\boldsymbol{p}(u,v,w)$ and $\boldsymbol{q}(u,v,w)$:

C0 continuity condition

$$\boldsymbol{p}_{1vw} = \boldsymbol{q}_{0vw}$$

C1 continuity condition

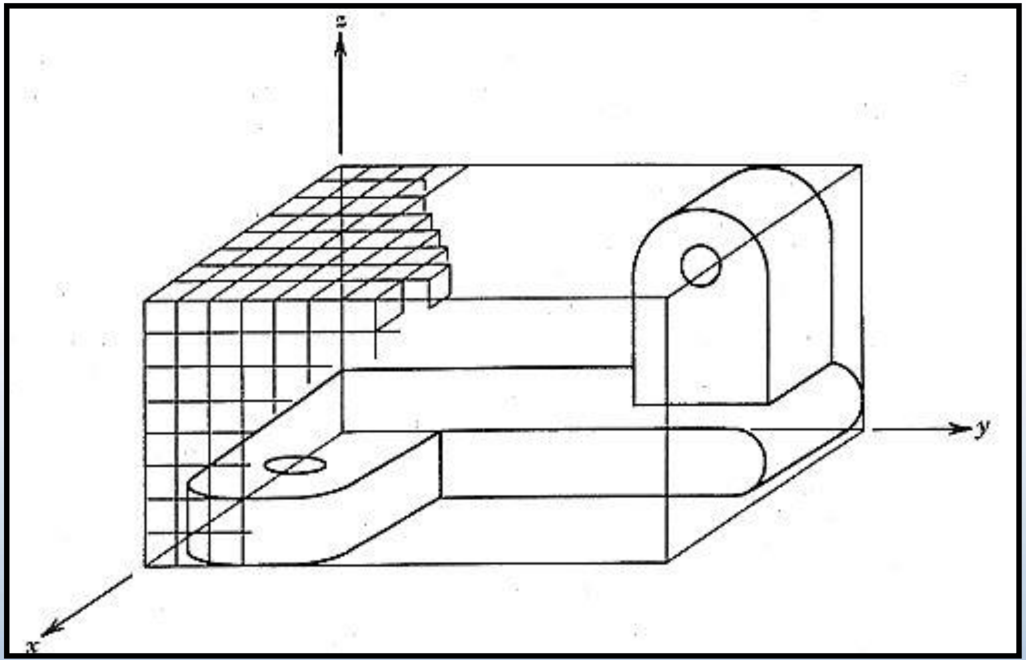
$$\boldsymbol{p}^{u}_{1vw} = \boldsymbol{q}^{u}_{0vw}$$



Mortenson, p. 254

Spatial-occupancy enumeration

- Special case of cell decomposition with identical cells arranged in a fixed, regular grid. The cell are often called voxels.



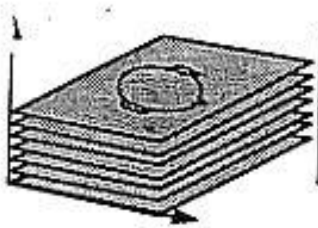
The most common cell type is the cube, and the representation of space as a regular array of cubes is also called a cuberrile.

Spatial-occupancy enumeration

- For every cell only it presence (1) or absence (0) in the grid is defined. A cell is presented in the grid if it is occupied by the object.
- Set- theoretic operation can be easily implemented.
- Disadvantages:

  - approximate model, no concept of “partial occupancy”
  - not affine invariant (moving, scaling, rotations)
  - deformations are also not straightforward
  - memory consuming (up to n^3 cells)

Volumetric objects



x_i	y_i	z_i	Density
0.00	0.00	0.00	243
0.00	0.00	0.12	175
.	.	.	.
0.00	0.00	1.00	186
0.00	0.12	0.00	187
.	.	.	.

$$F_{ijk} = F(X_i, Y_j, Z_k)$$

$i = 1, \ldots, N \quad j = 1, \ldots, N \quad k = 1, \ldots, N$

Medical Scanners, MRI, PET, ect.

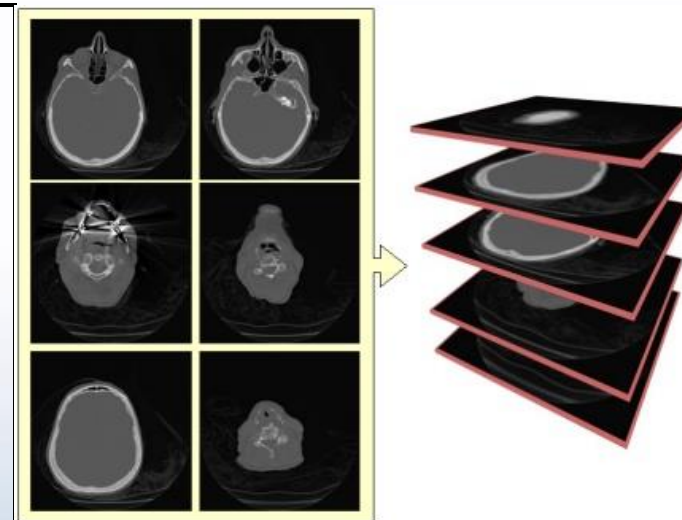


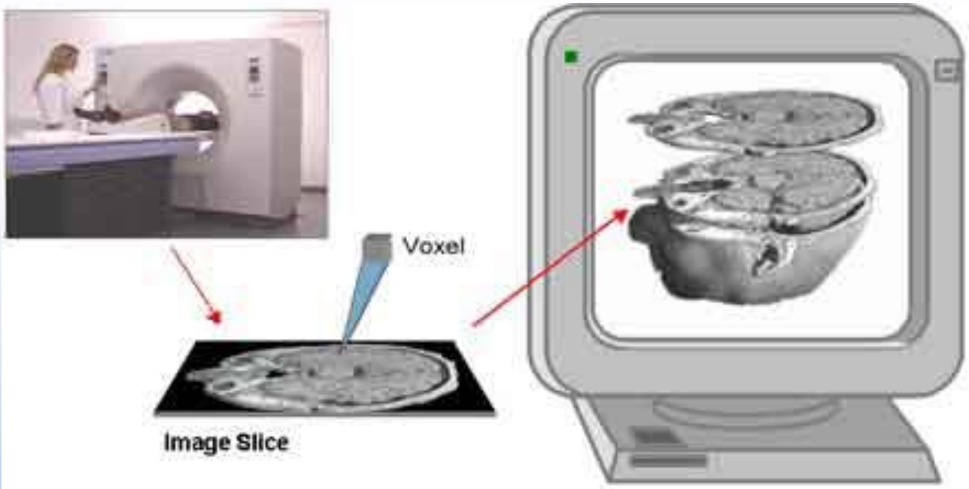
Image by Andrew Winter

Numerical value is defined in the nodes of 3D grids: density, temperature, pressure, and so on.

Object is defined by $F > 0$.

Source of data:

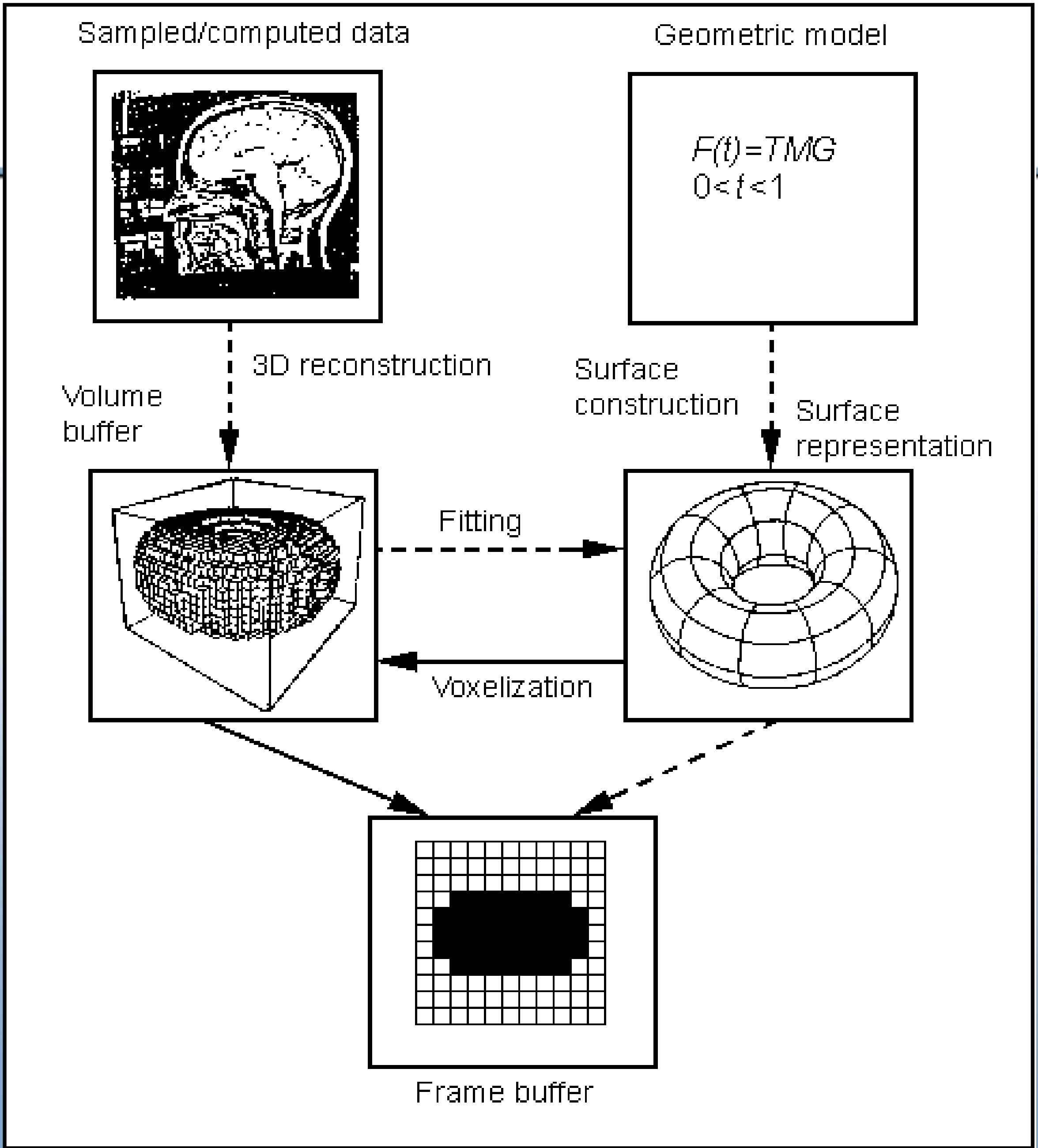
- Computer tomography
- Numerical simulation
- Manual sculpting (like 3D drawing with a brush)
- Voxelized geometric primitives:



$$F_{ijk} = 1,\ \ if\ \ f(x_i, y_j, z_k) \geq 0$$

$$F_{ijk} = 0,\ \ if\ \ f(x_i, y_j, z_k) < 0$$

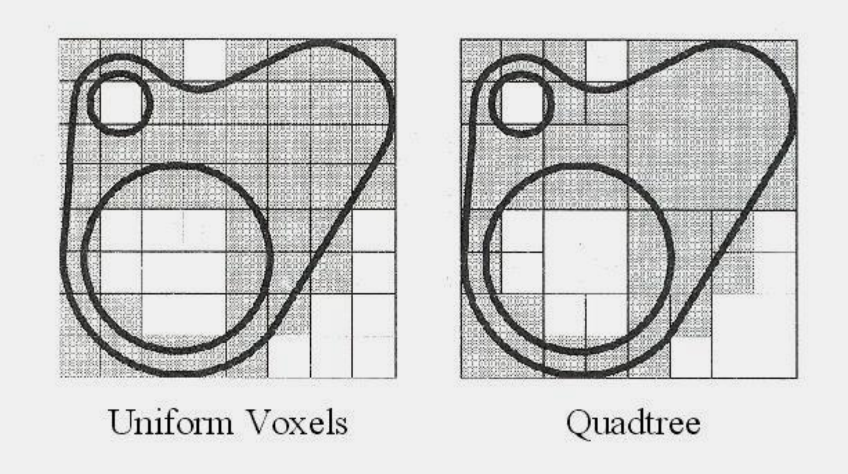
Volumetric objects



Quadtrees and Octrees

- Octrees are a hierarchical variant of spatial occupancy enumeration. Octrees are derived from quadtrees, a 2D image representation.

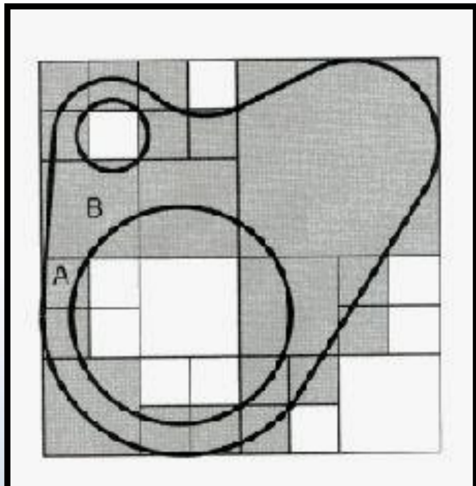
- A quadtree is derived by recurrently subdividing a 2D plane in both directions to form quadrants.

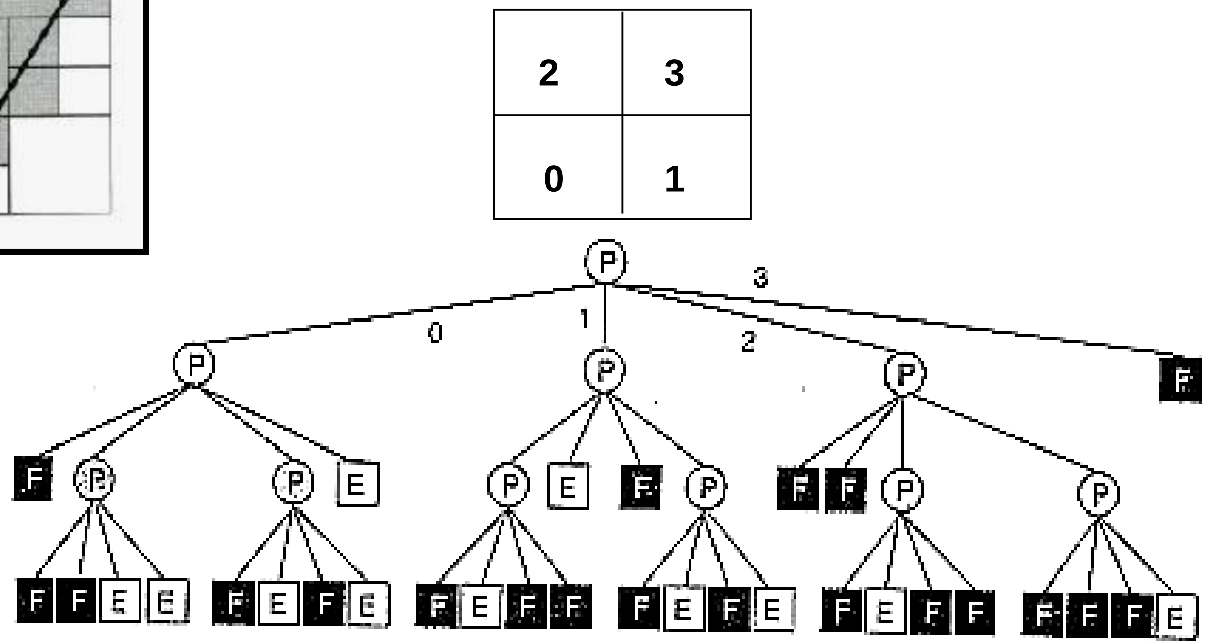


(a) (b)

An object represented using
(a) spatial-occupancy enumeration
(b) quadtree

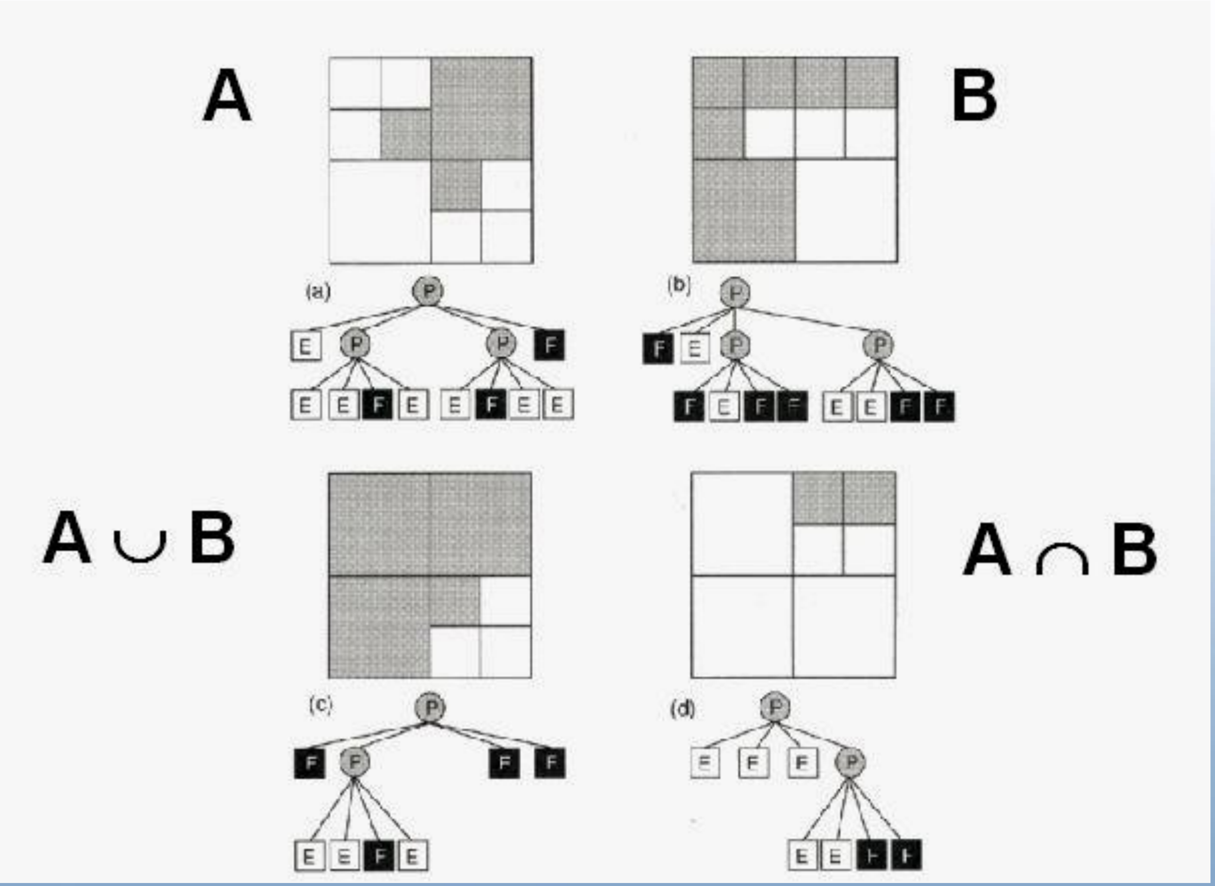
Quadtree data structure for the object





F = full; P = partially; E = empty

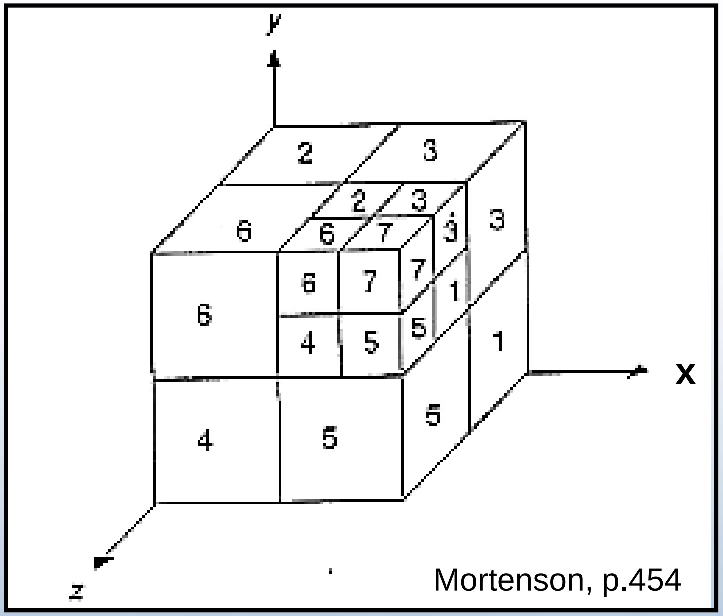
Boolean set operations on quadtrees



Octrees

Octrees: example

An octree scheme divides regions of 3D space (usually cubes) into octants and stores 8 data elements in each node of the tree.



Octree enumeration. Octant 0 is not visible

Procedure for generating octree: each octant is tested, and octant subdivisions continue until octants are homogeneous (full or empty).

Octrees

- Number of nodes in an octree is proportional to the object's surface.
- Operations: Boolean, rotation by 90 degrees, scaling by powers of 2, translations.
- Problem of aliasing under general transformation.

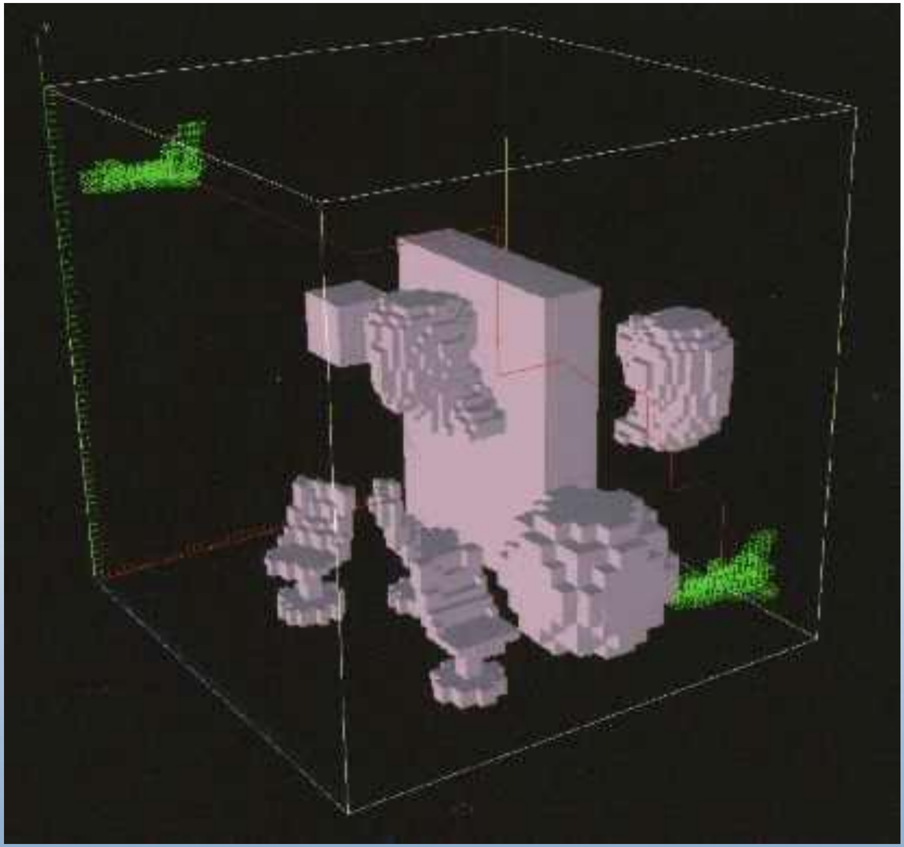
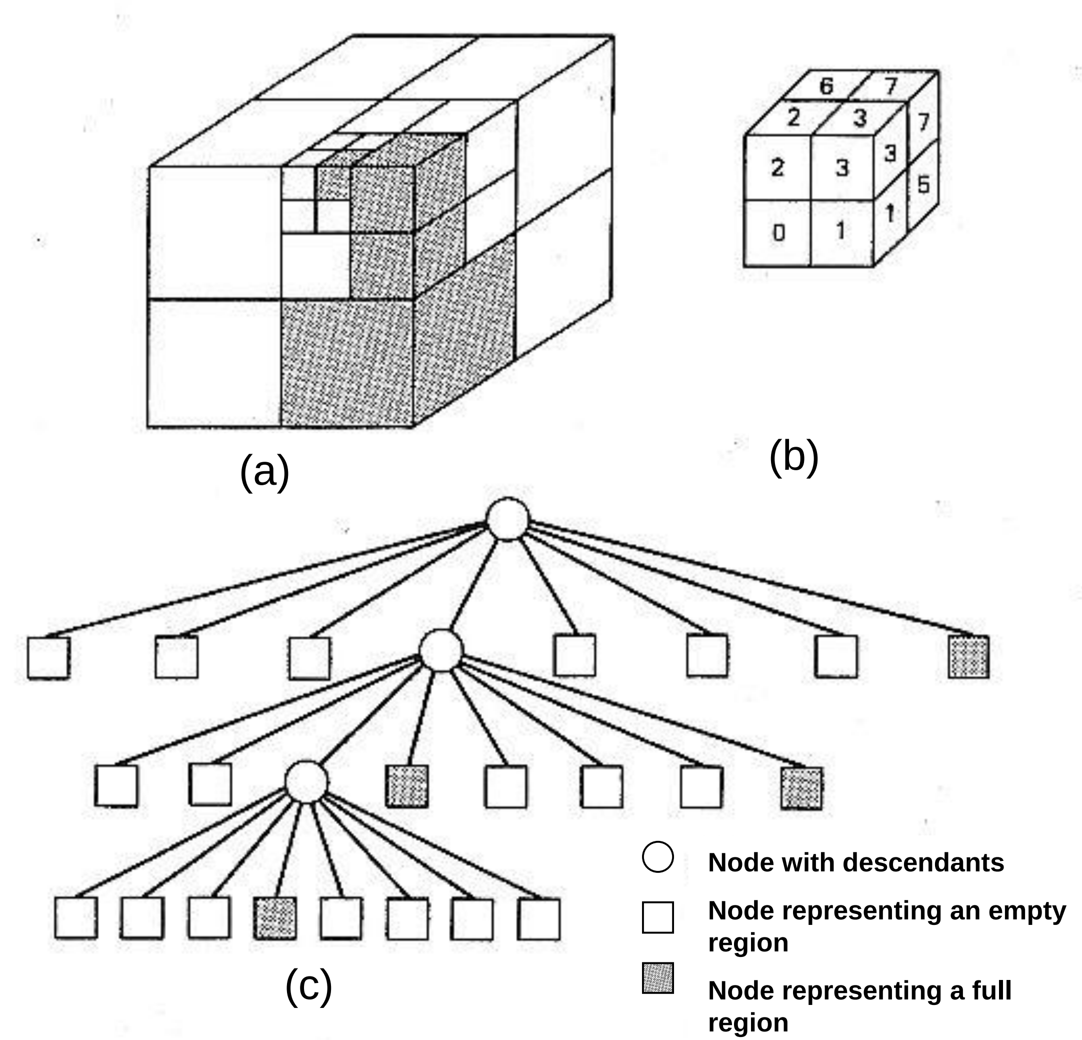


Image by Yoshifumi Kitamura

Octrees: example

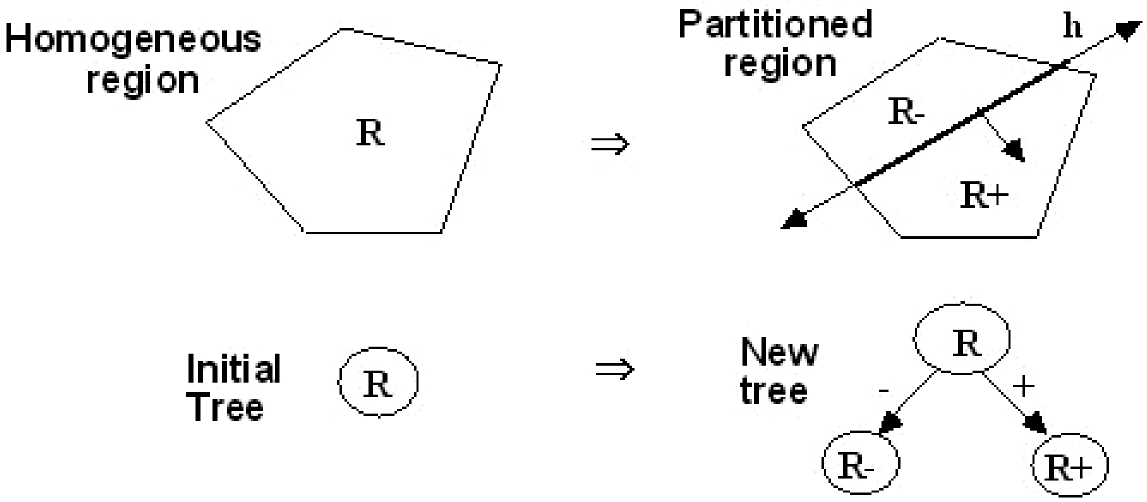


Binary space-partitioning (BSP) trees

- BSP trees recursively divide space into pairs of subspaces, each separated by a plane of arbitrary orientation and position.
- Each internal node of the BSP tree is associated with a plane and has two child pointers, one for each side of the plane.
- If the halfspace on a side of the plane is homogeneous, then its child is a leaf and represents a region either inside (“in”) or outside (“out”) the object.

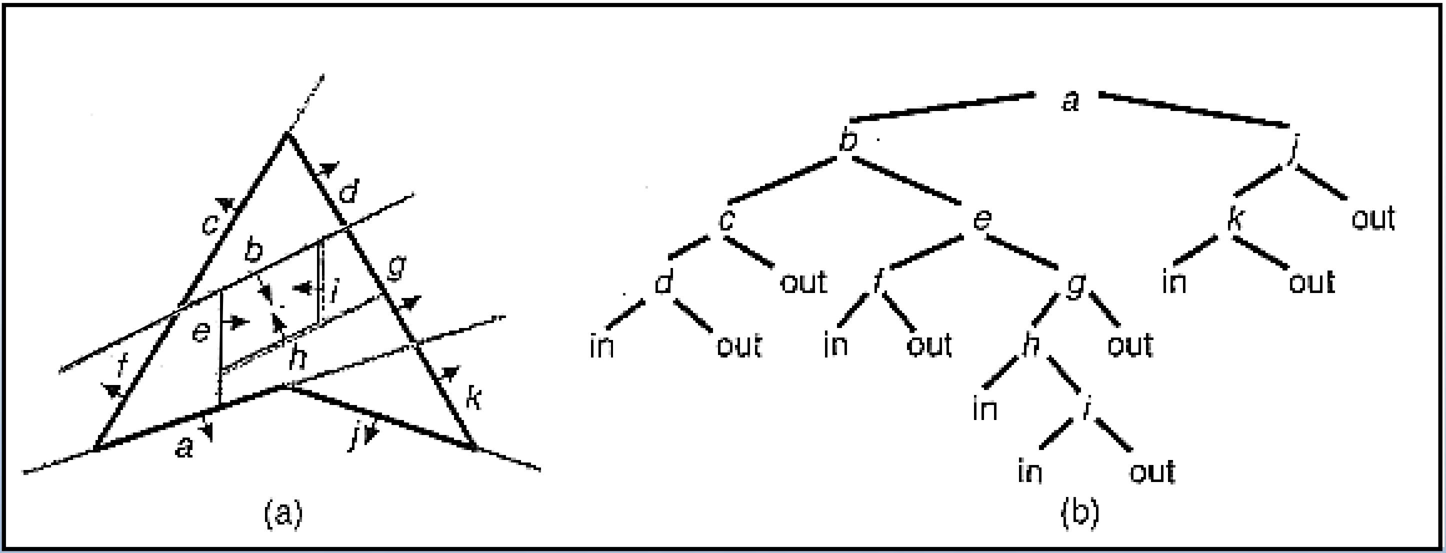
BSP trees

- Single geometric operation
  - Partition a convex region by a hyperplane
- Single combinatorial operation
  - Two child nodes added as leaf nodes



B. Naylor

A BSP tree can represent an arbitrary concave polyhedral solid with holes as a union of convex "in" regions.

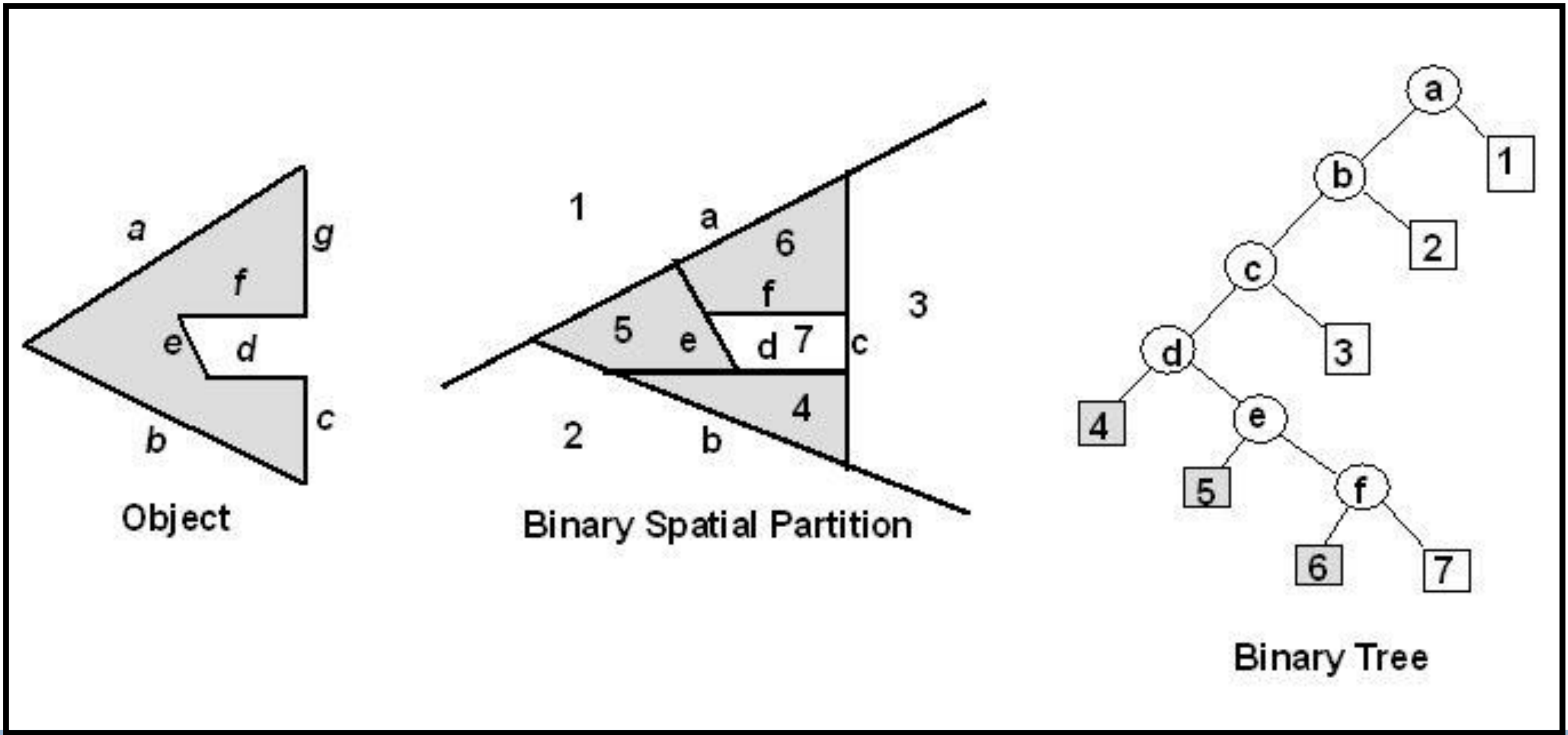


A BSP tree representation in 2D:

(a) A concave polygon bounded by black lines. Lines defining the half-spaces are dark gray, and "in" cells are light gray.

(b) BSP tree.

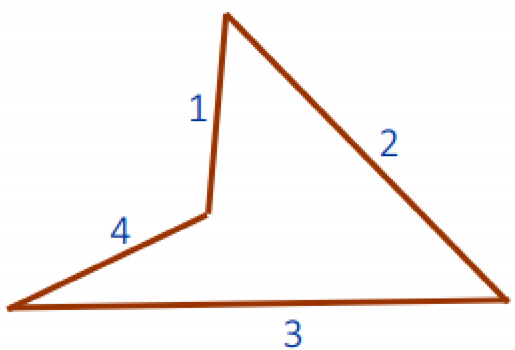
BSP trees



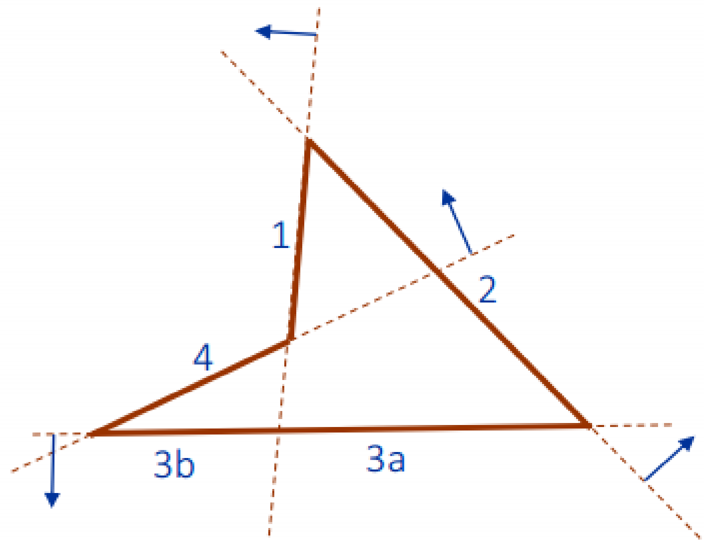
B. F. Naylor

PMC with BSP trees

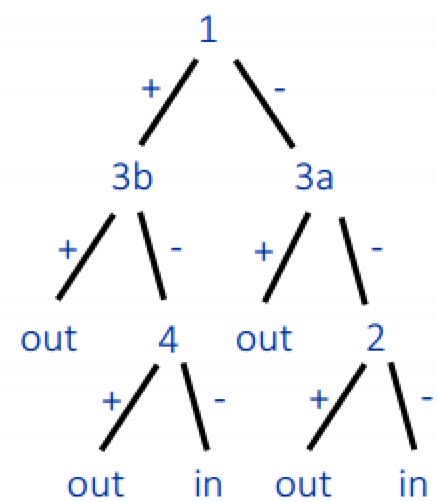
- BSP trees can be used for the inside / outside classification of closed polygons



scene



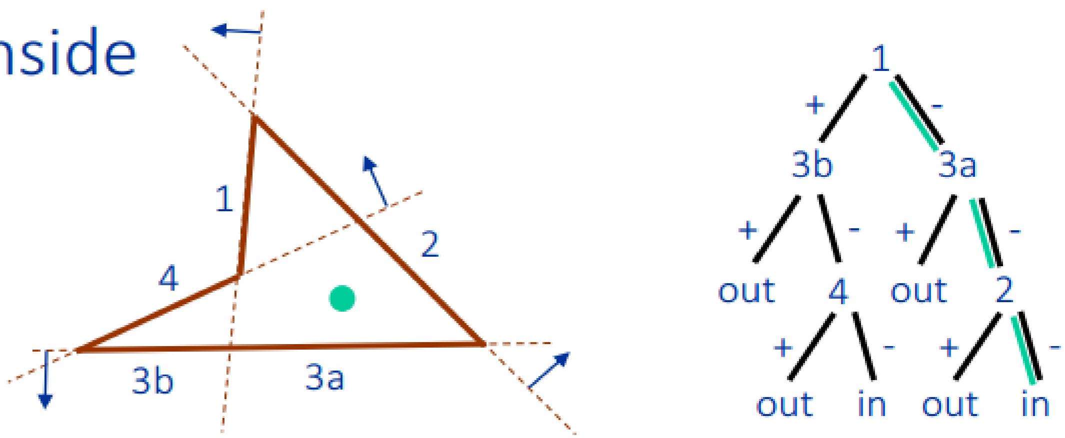
scene partitioning



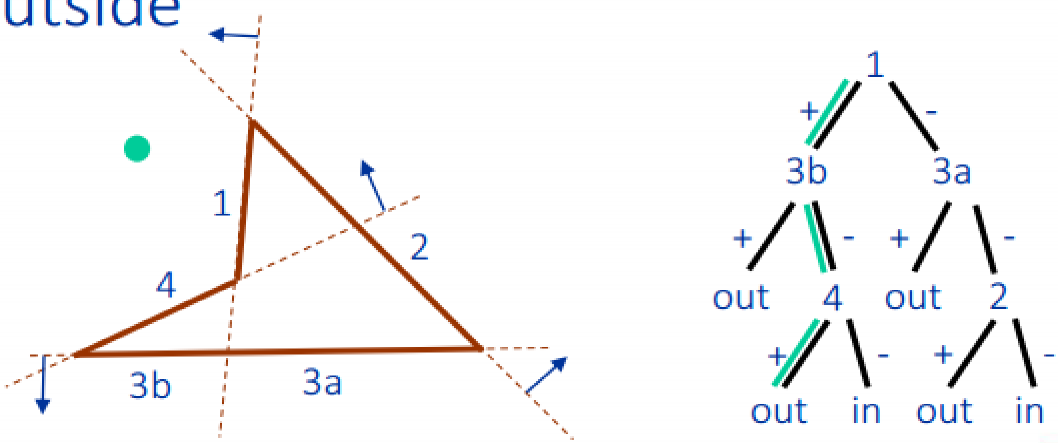
solid-leaf BSP tree

PMC with BSP trees

- query point is inside



- query point is outside



Schemes Comparison

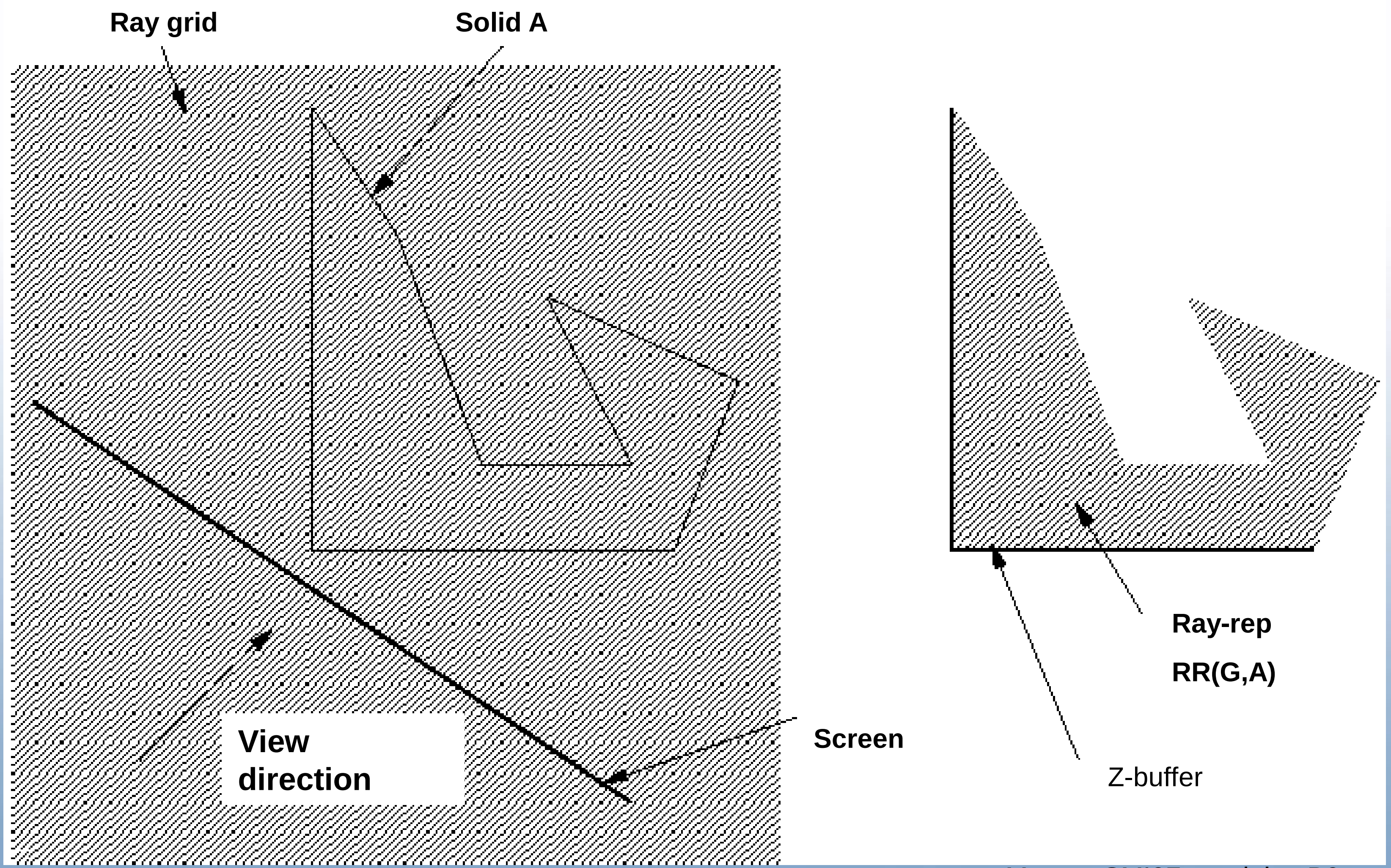
	Voxels	Octree	BSP	CSG
Accurate	No	No	Some	Some
Concise	No	No	No	Yes
Affine invariant	No	No	Yes	Yes
Easy acquisition	Some	Some	No	Some
Guaranteed validity	Yes	Yes	Yes	No
Efficient boolean operations	Yes	Yes	Yes	Yes
Efficient display	No	No	Yes	No

T. Funkhouser

Ray representation

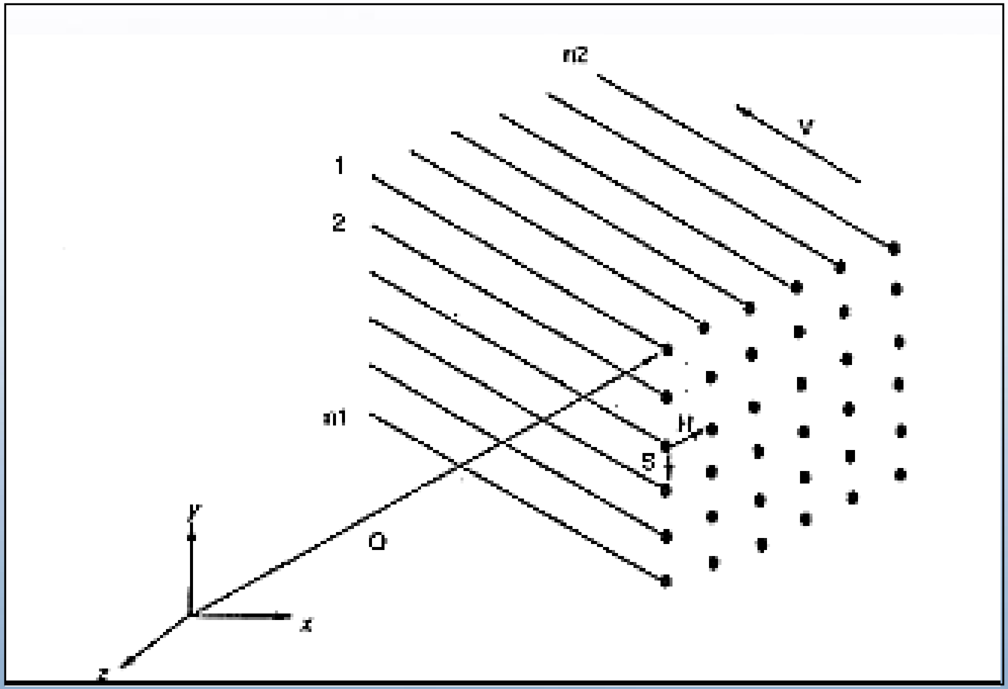
- A ray grid *G* is a rectangular array of finitely spaced parallel lines.
- The ray-rep *RR(G, A)* of a solid *A* is a set of *GinA* and *GonA* segments of a ray grid.
- A tag is a descriptive symbolic information appended to the *GinA* segments.

2D ray-rep:



Menon, SMI97 tutorial, p.B9

A ray grid in 3D:



Menon, SMI97 tutorial, p.B17

CSG to ray-rep conversion with RayCasting Engine

RayCasting Engine (RCE) is highly parallel, special-purpose computer that generates ray-reps by classifying ray grids with respect to CSG solids.

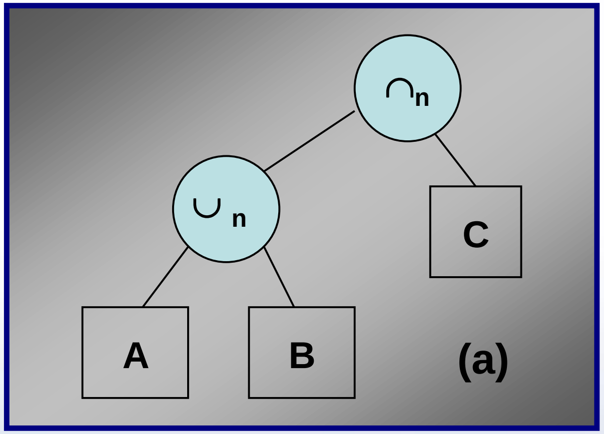
The output from the RCE is a list of interval segments. Each interval in the list represents an interval of intersection between a ray and a CSG solid. This collection of interval segments is a discrete approximation of the solid.

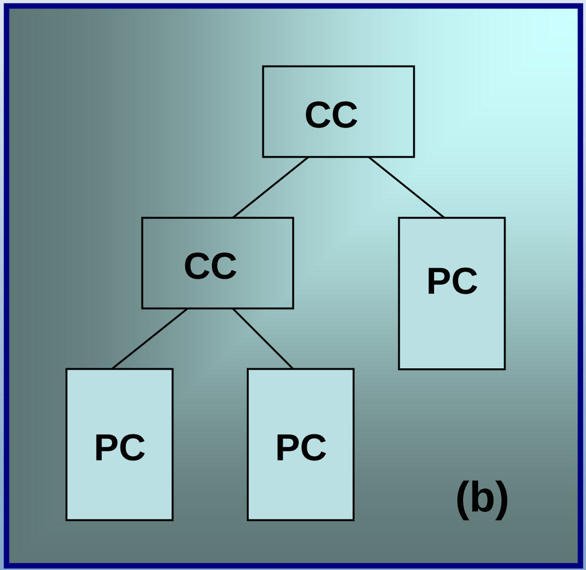
RCE functions

- Performs ray/ primitive classification in parallel with Primitive Classifiers (PCs)
- Provides a deep pipeline to perform the upward propagation of the classification result. Classification Combiners (CCs) combine “in” segment from the left and right subtrees in accordance with the set operations on the nodes of the tree.
- Combines classifications in parallel at every level of the tree

RCE basics

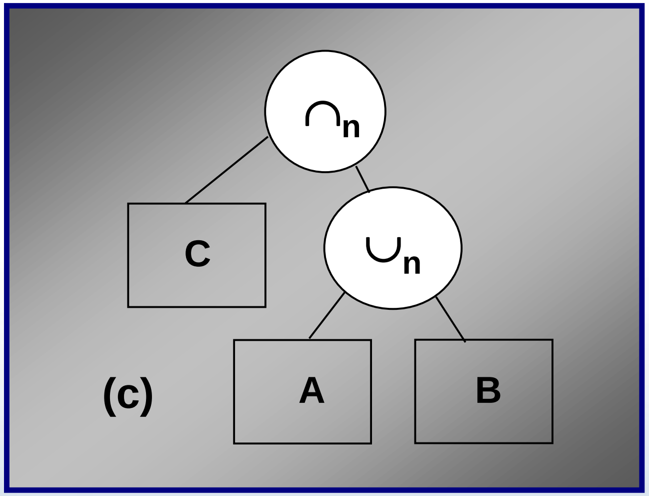
A CSG tree (a) is associated with a programmably reconfigurable tree (b) whose leaves are Primitive Classifier processors and whose nodes are Classification Combine processors.

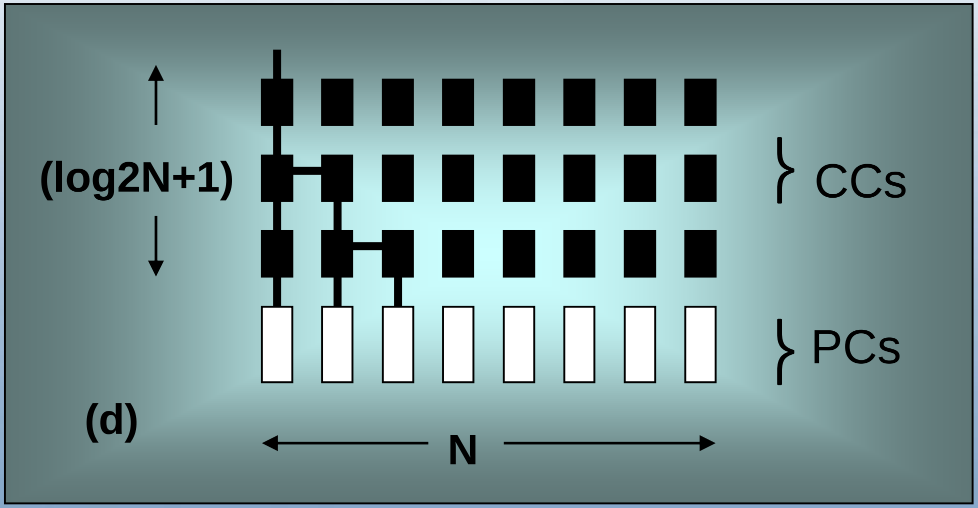




RCE basics

We can present any CSG tree by a right-heavy tree as in (c), and any N-leaf right-heavy tree can map onto an $N \times (\log_2 N+1)$ array of processors as in (d).





References

- J. Foley et al. “Computer Graphics. Principles and Practice”, Addison-Wesley, 1990, 2nd ed. 1997, 1174 p.
- Michael E. Mortenson “Geometric Modeling”, John Wiley & Sons, 1985, 2nd ed. 1997, 3d ed. 2006, 763 p.
- A. Kaufman, D. Cohen, R. Yagel “Volume graphics”, Computer, vol.26, No.7, 1993, pp. 51-64.

References

- Naylor B., Amanatides, J. and Thibault W., Merging BSP Trees Yields Polyhedral Set Operations, Computer Graphics (SIGGRAPH '90 Proceedings), 24(4), 1990, pp. 115-124.

- Bruce F. Naylor, Binary space partitioning trees as an alternative representation of polytopes, CAD 22(4), pp. 250-252, 1990.

- J. Menon, R. Marisa, J. Zagajac "More powerful solid modeling through ray representations", IEEE Computer Graphics and Applications, vol.14, No.3, 1994, pp. 22-35.