# Marching Cubes algorithm

Carlos Andújar

April 2014

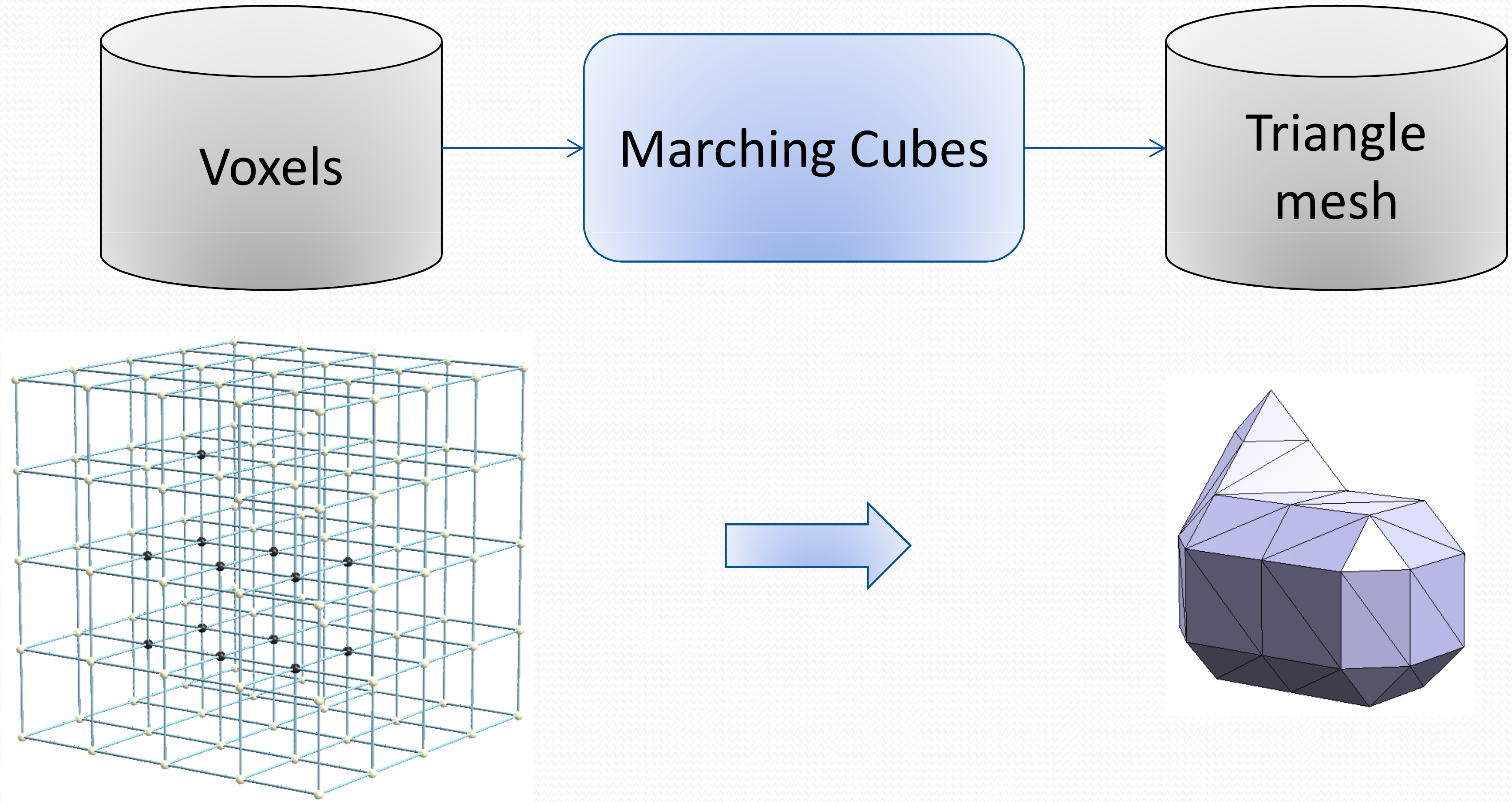
# Introduction

- Marching Cubes (MC) is a popular algorithm for isosurface extraction (creating a polygonal mesh from a voxel model)
- Original version:

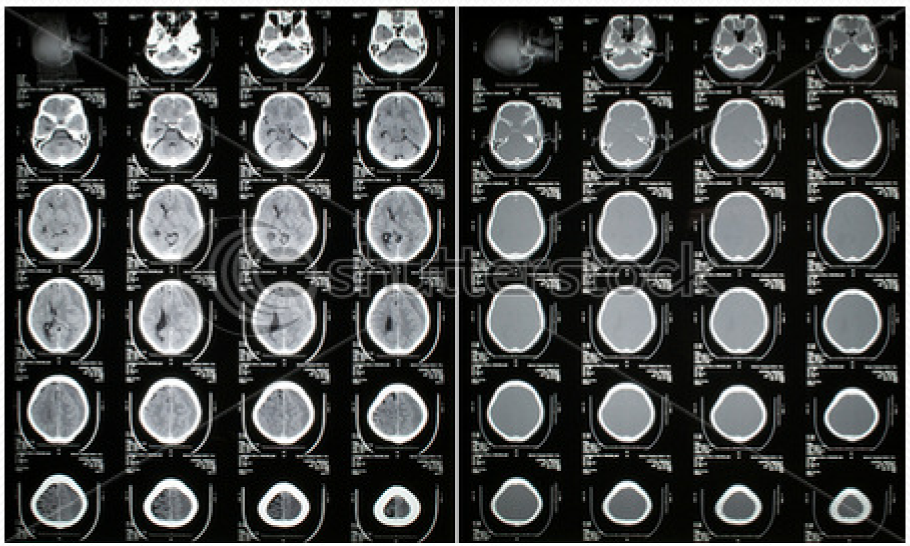
  Lorensen, W.E. and Cline, H.E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. ACM Computer Graphics, 21(4). (SIGGRAPH '87)
- Improved version (ensuring closed meshes):

  Claudio Montani , Riccardo Scateni and Roberto Scopigno . A modified look-up table for implicit disambiguation of Marching Cubes. The Visual Computer, 10(6), 353-355

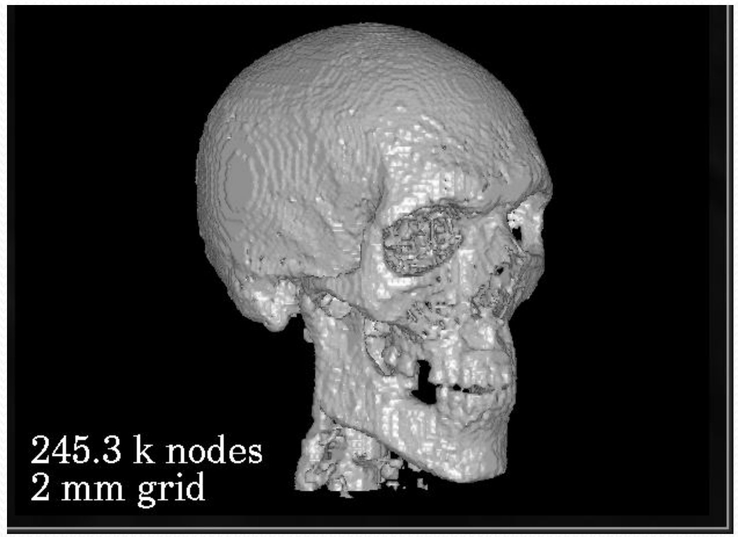
# Introduction



# Introduction

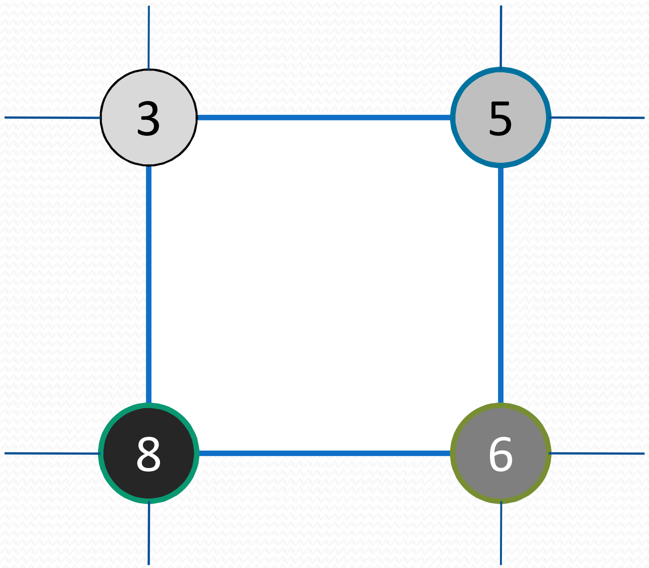





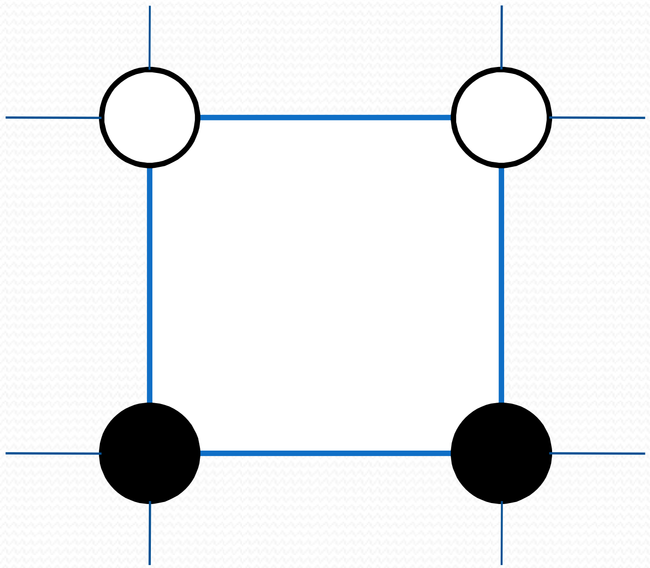


# Input data

- The input of the MC algorith is a **voxelization** representing a **scalar field** v=f(x,y,z)
- The input scalar field might be binary (or not):



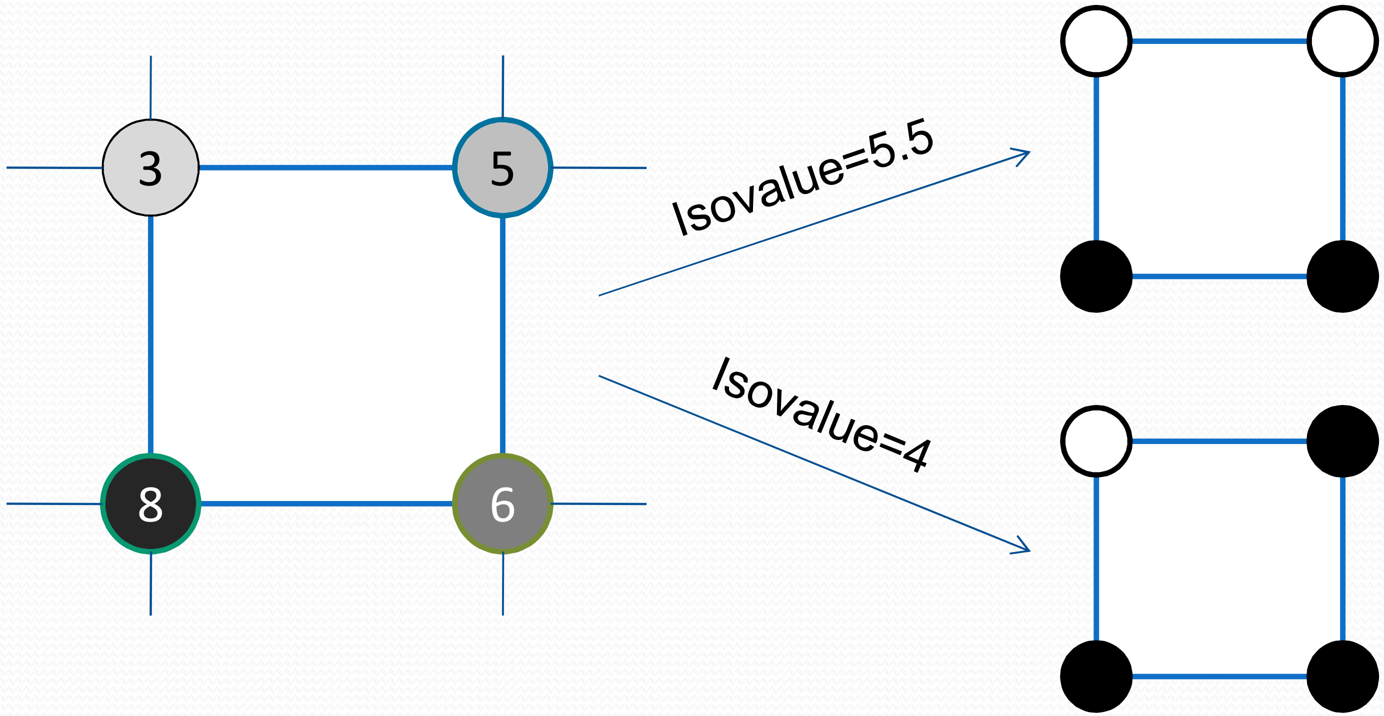

Values in $\mathbb{R}$



Binary values

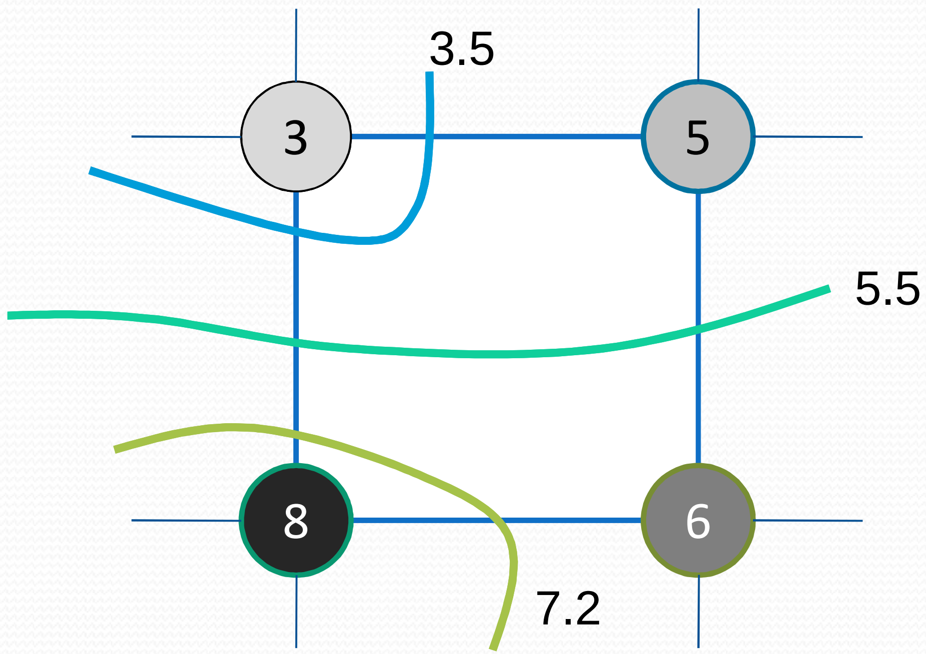
# Input data

- If the input data is non-binary, MC requires an additional parameter (*threshold value* or *isovalue*) to classify samples as inside/outside the surface.



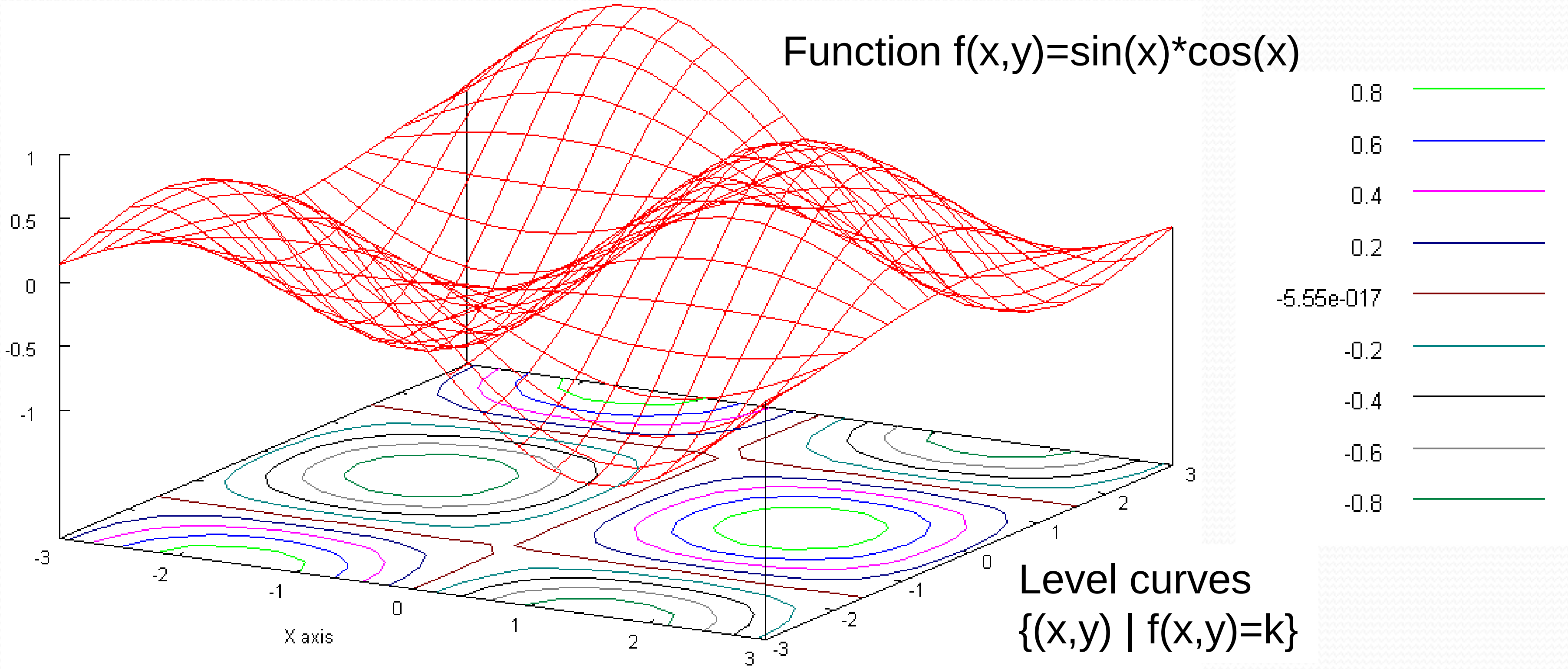
# Output surface

- If the input model is binary, we would like a surface *separating interior from exterior points*.
- If the input model is not binary, we would like the *isosurface* joining all points with the choosen *isovalue*.



# Level curves and level surfaces

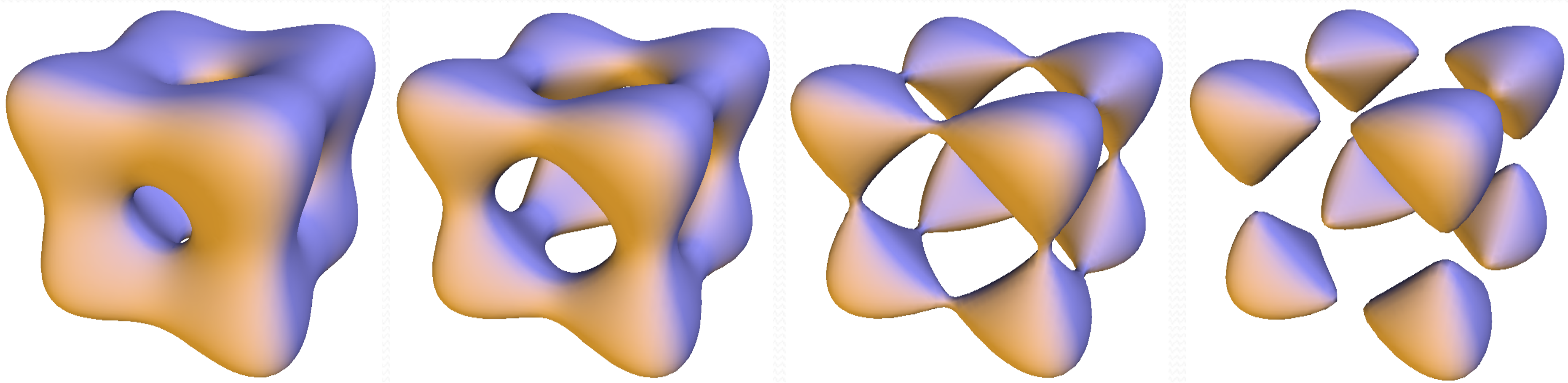
- A mapping $\mathbb{R}^2 \rightarrow \mathbb{R}$ defines **level curves**



# Level curves and level surfaces

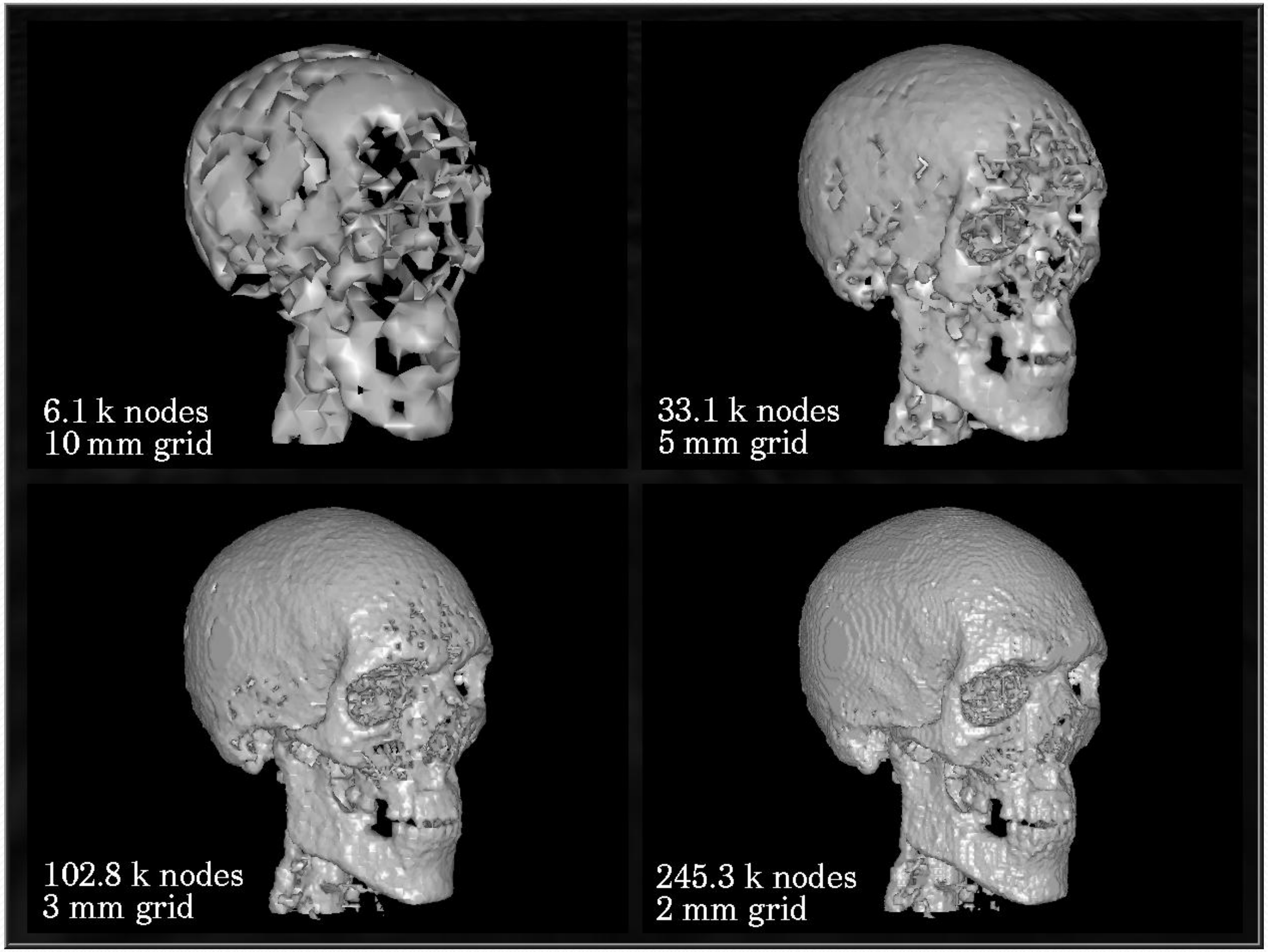
- A mapping $\mathbb{R}^3 \rightarrow \mathbb{R}$ defines **level surfaces**:

$$\{ (x,y,z) \mid f(x,y,z)=k \}$$



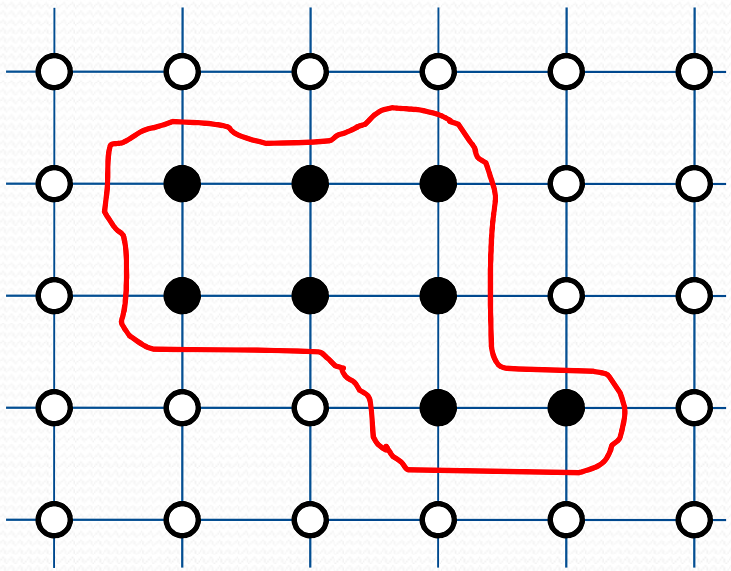
Level surfaces of $f(x,y,z)= (3x)^4 + (3y)^4 + (3z)^4 - 45x^2 - 45y^2 - 45z^2$

# Varying the isovalue



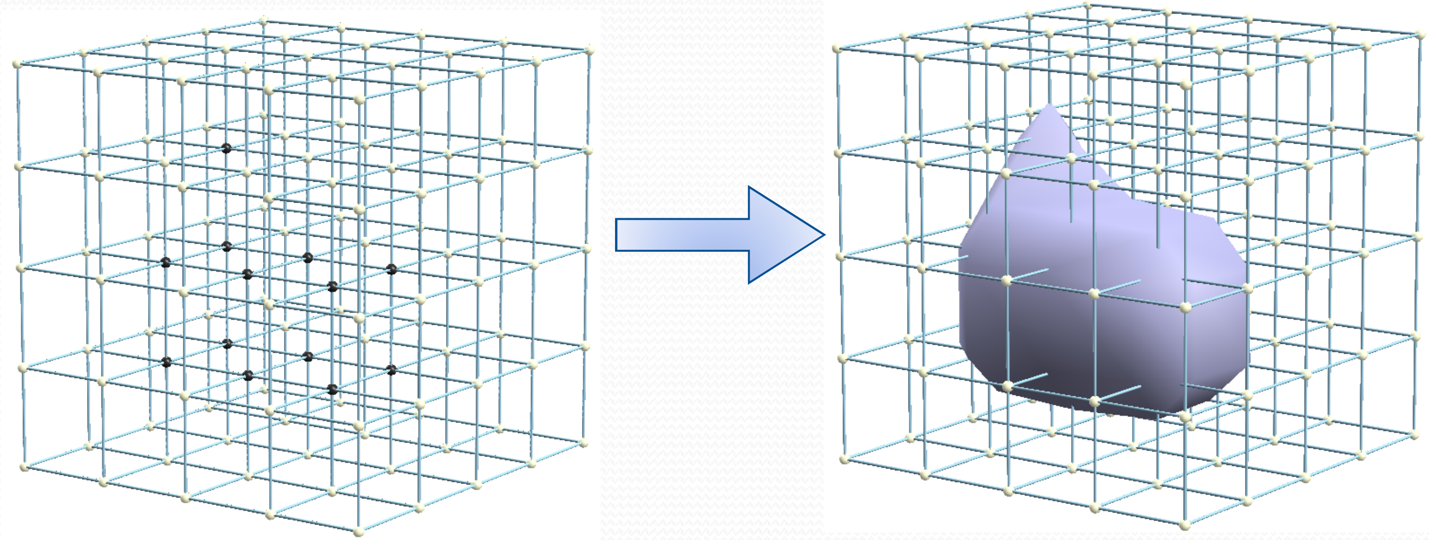
# Output surface

- The output surface always has to fulfill these conditions:
  - It must separate interior points from interior points
  - Thus it must be orientable and closed.



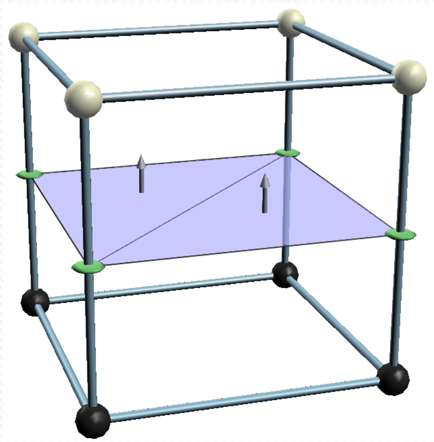
# Output surface

- The output surface always has to fulfill these conditions:
  - It must separate interior points from interior points
  - Thus it must be orientable and closed.



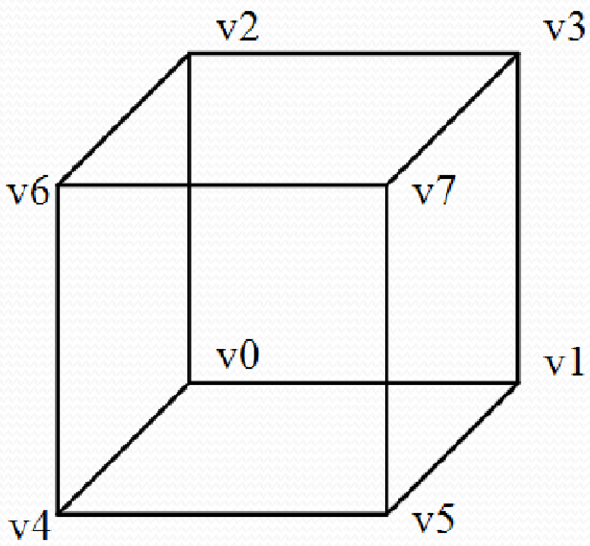
# Basic idea

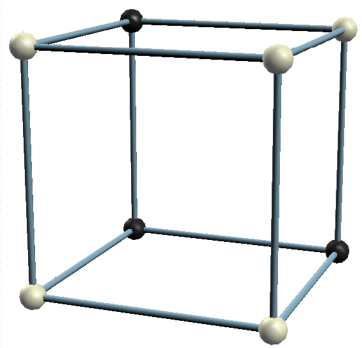
- The basic idea in MC is to traverse ("march") all the cubes formed by 2x2x2 neighboring samples.
- For each cube, MC generats a set of triangles corresponding to the output isosurface inside the cube (all triangles generated by MC belong to a unique cube).



# Configurations & cases

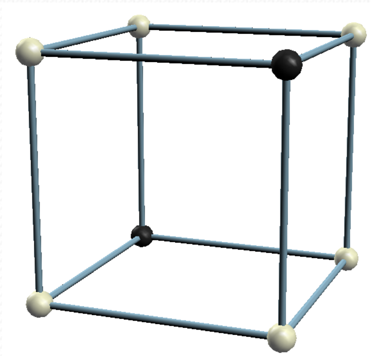
- Each cube has 8 B/W vertices → $2^8 = 256$ configs
- If we label the vertices, each configuration can be represented with one byte.



???

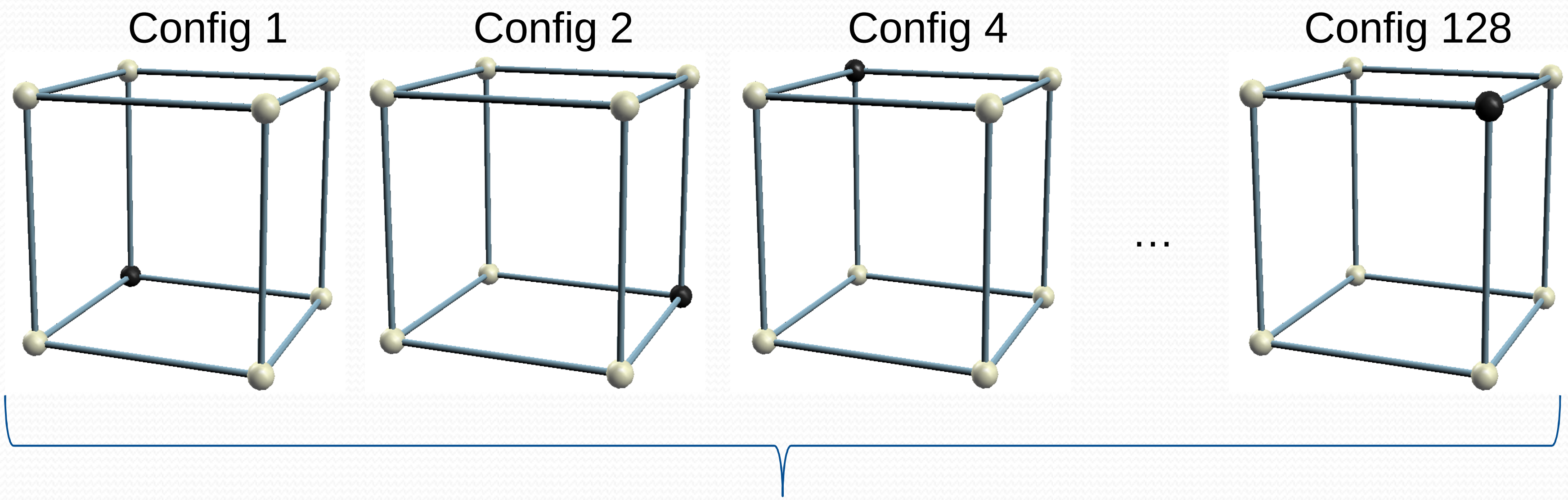
Config 7 = 0000 0111



Config 129 = 1000 0001

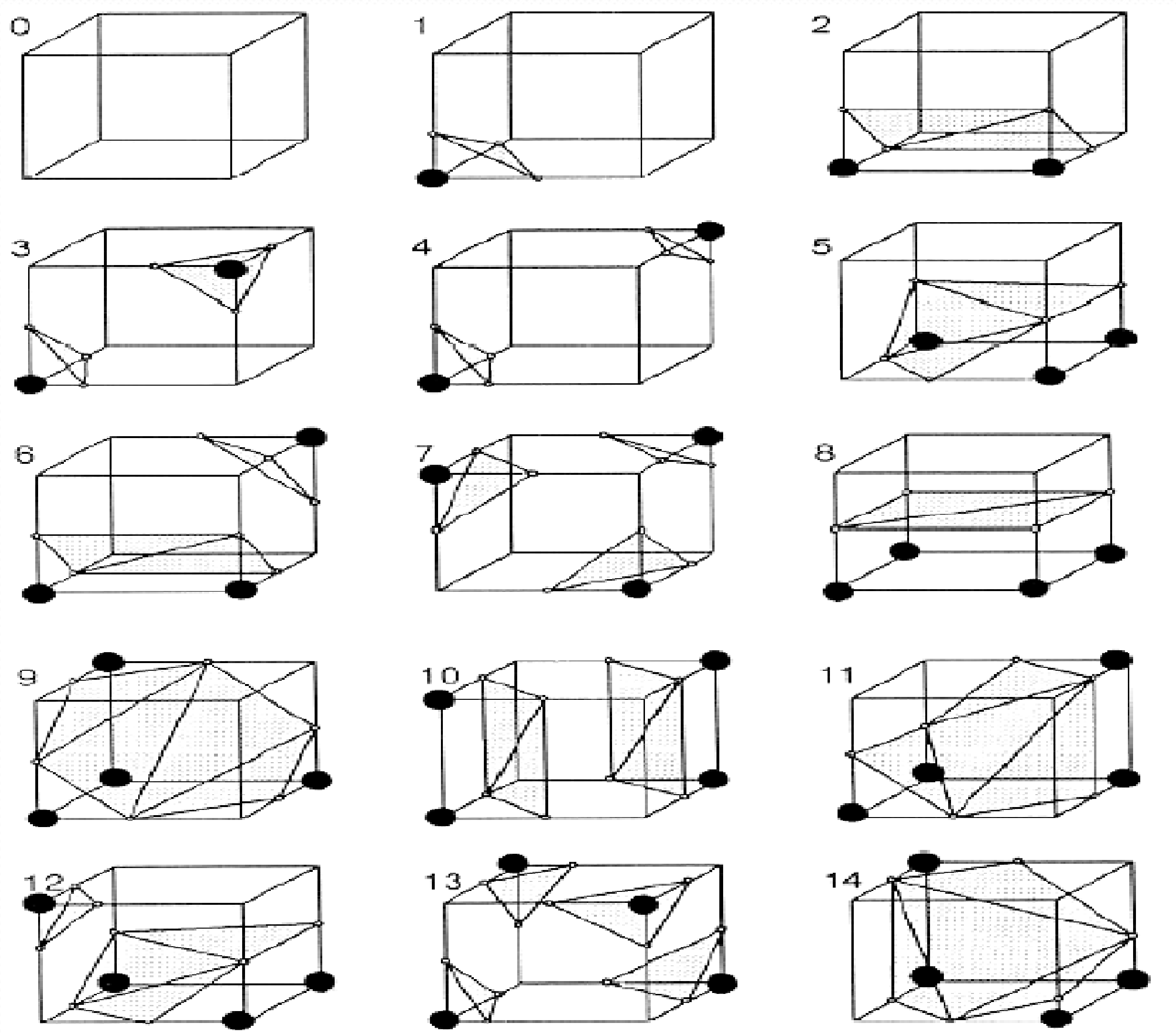
# Configurations & cases

- Many *configurations* are symmetric and can be grouped.
- Grouping symmetric configurations results in *cases*.




These 8+8 configs are grouped into a single case.

# Cases in the original MC (14+1)

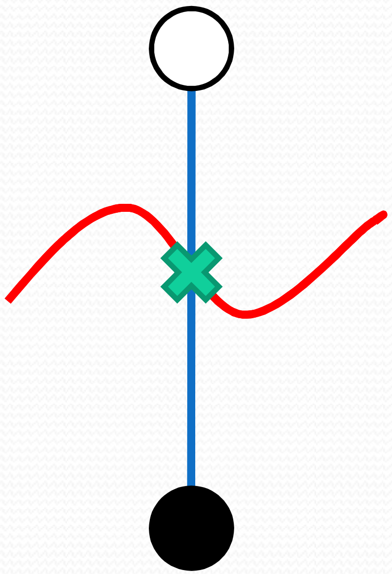


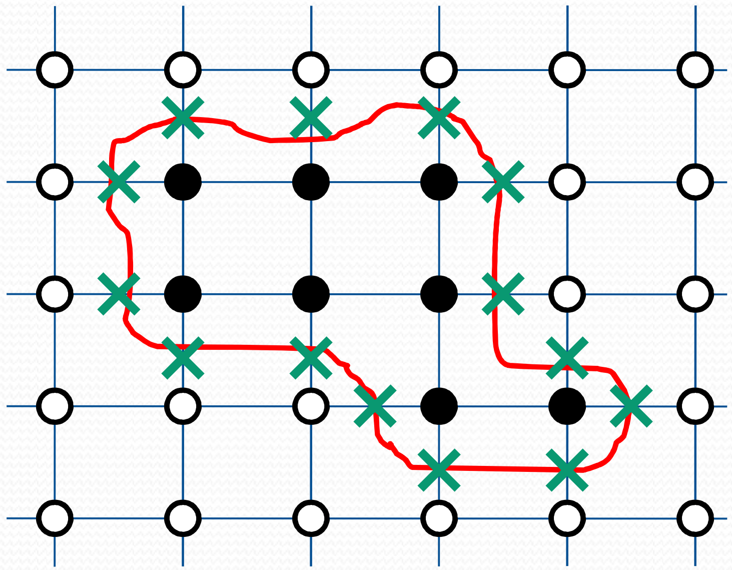
# Marching Cubes algorithm

- For each cube being processed, we must generate:
  - Geometry: vertices of the isosurface
  - Topology: triangles connecting these vertices.

# Creating vertices

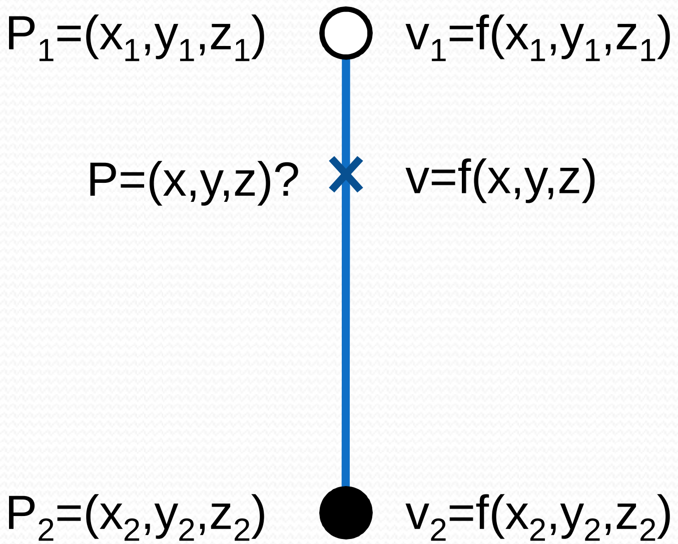
- Assuming that the field is continuous, then edges with a sign change must be intersected by the isosurface:





# Creating vertices

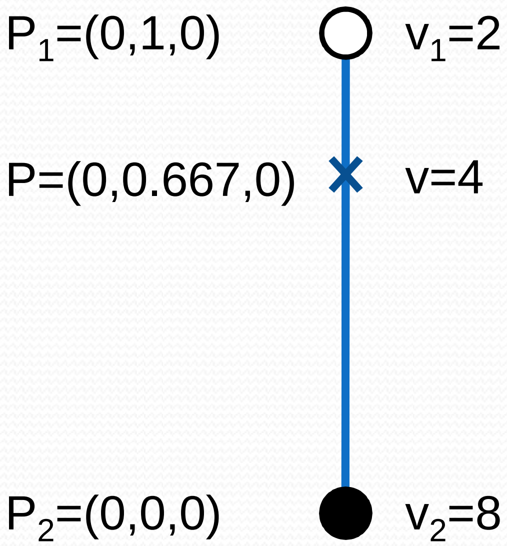
- Marching Cubes creates a vertex for each grid edge with a sign change.
- The exact position of the vertex along the edge is computed through **linear interpolation**:




$$P = P_1 \frac{|v - v_2|}{|v_2 - v_1|} + P_2 \frac{|v - v_1|}{|v_2 - v_1|}$$

# Creating vertices

- Marching Cubes creates a vertex for each grid edge witha sign change.
- The exact position of the vertex along the edge is computed through **linear interpolation**:

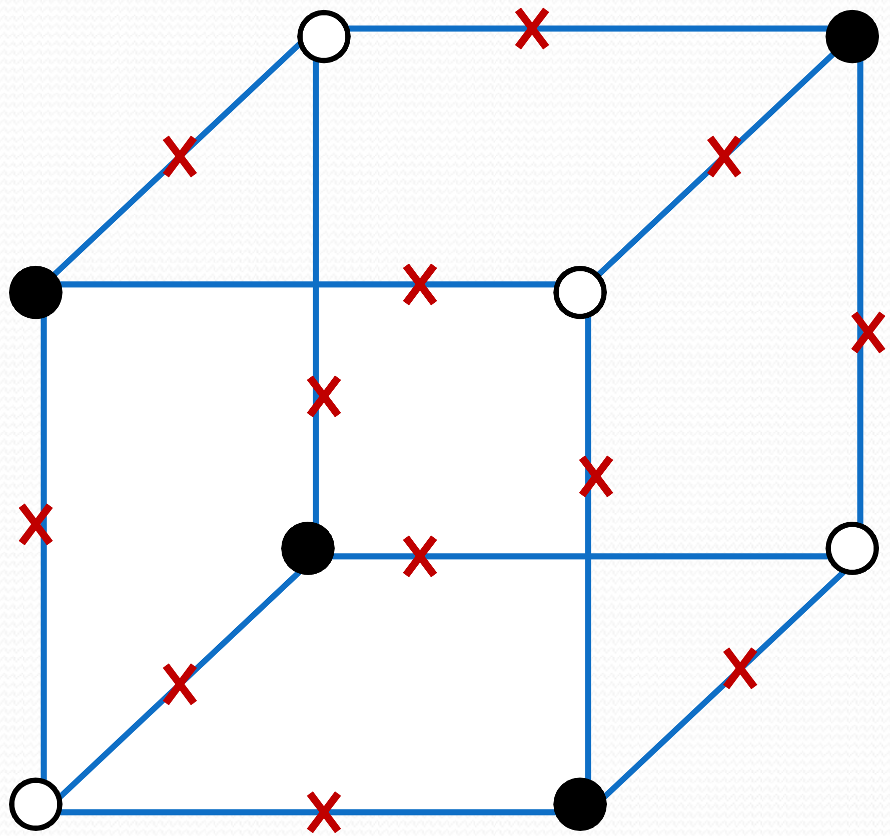



$$P = (0,1,0)\frac{|4-8|}{|8-2|} + (0,0,0)\frac{|4-2|}{|8-2|}$$

$$P = (0,\frac{2}{3},0)$$

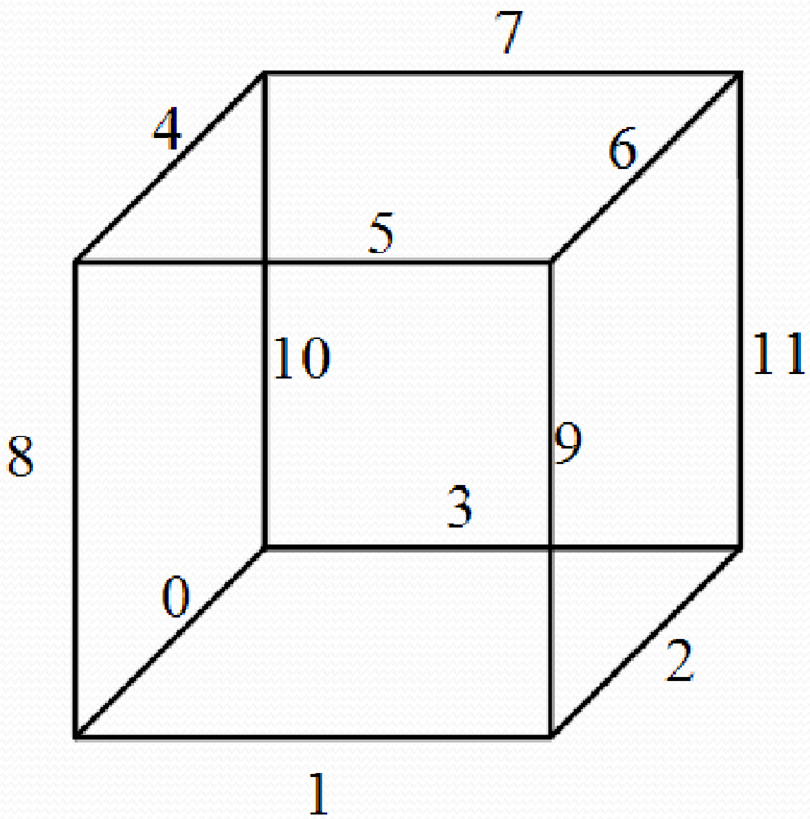
# Creating vertices

- Vertices can be created in any of the 12 grid edges.



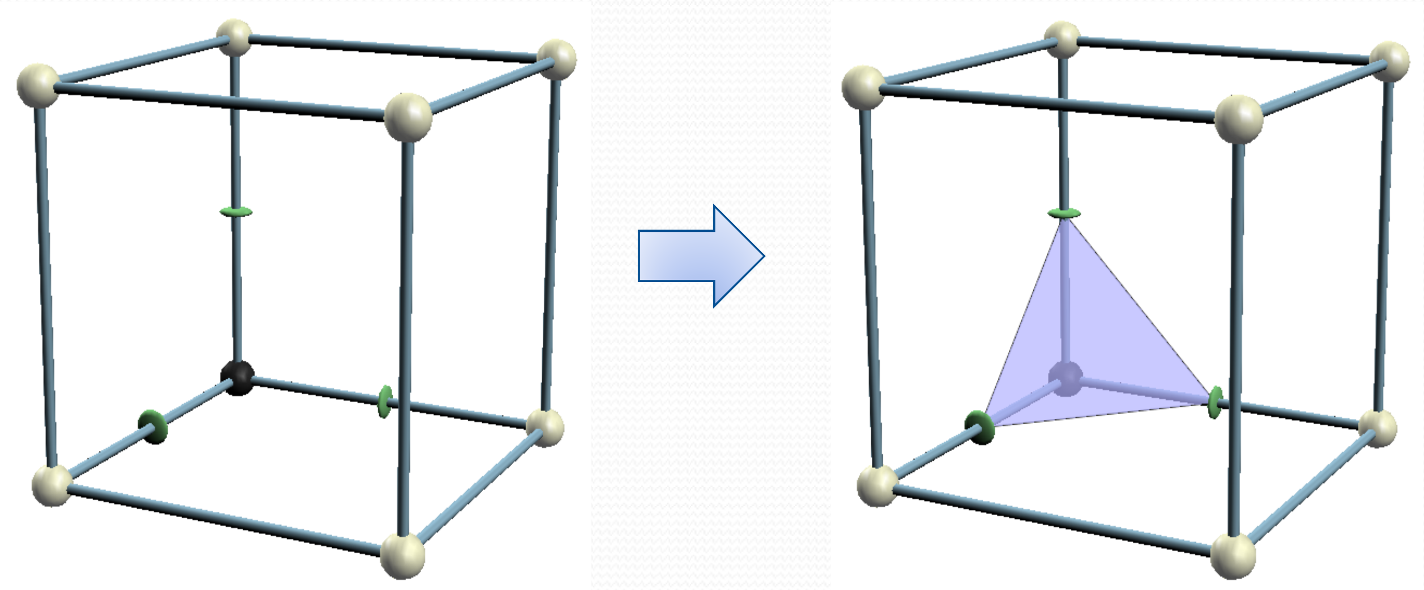
# Creating vertices

- We will used this edge numbering (≠ Lorensen,Cline)



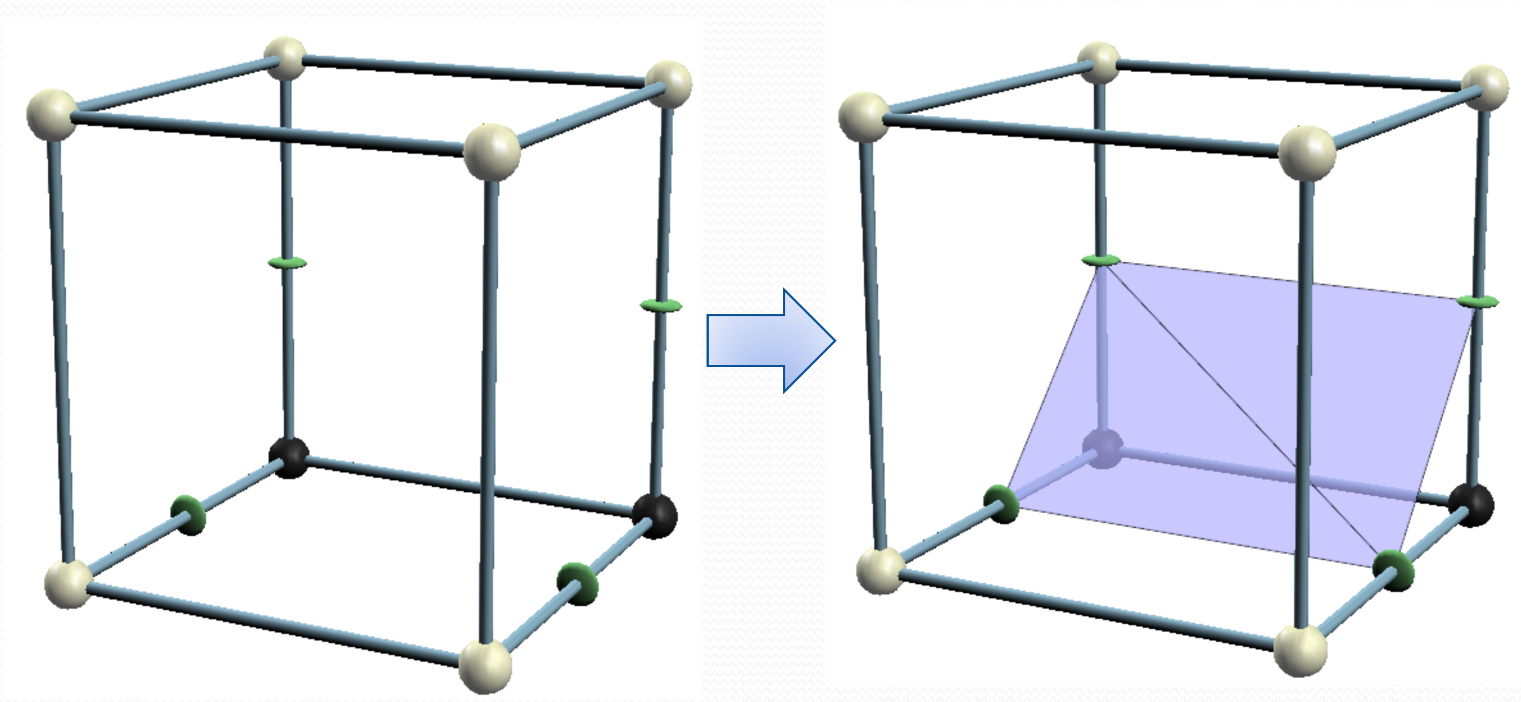
# Creating triangles

- Some cases are trivial...



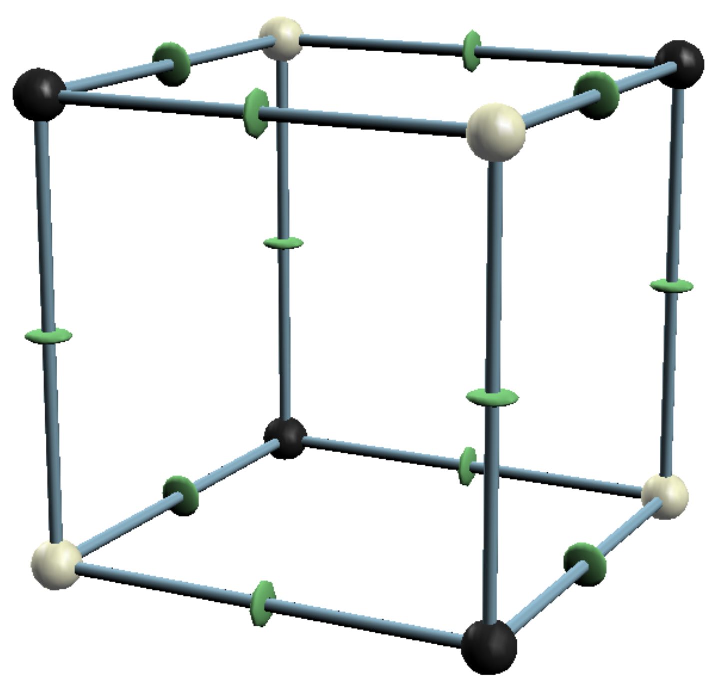
# Creating triangles

- Some cases are trivial...



# Creating triangles

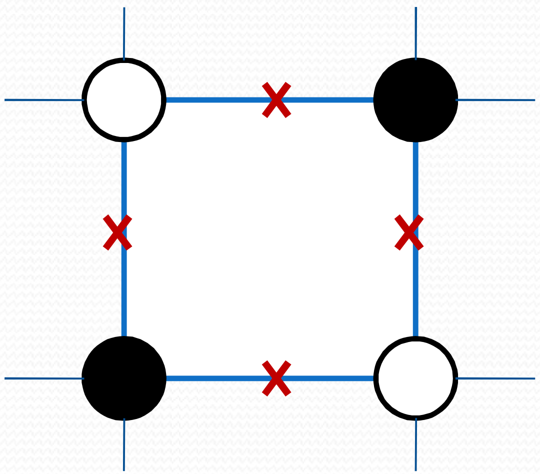
- Others are not...

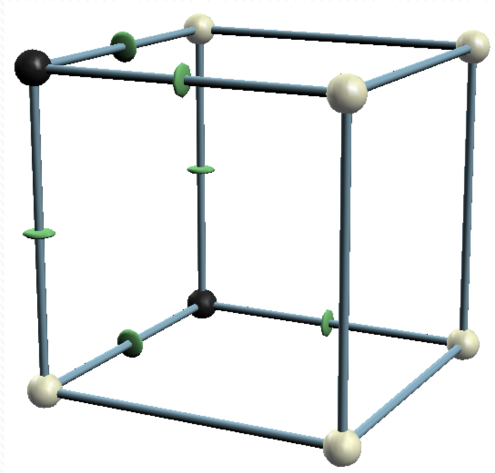


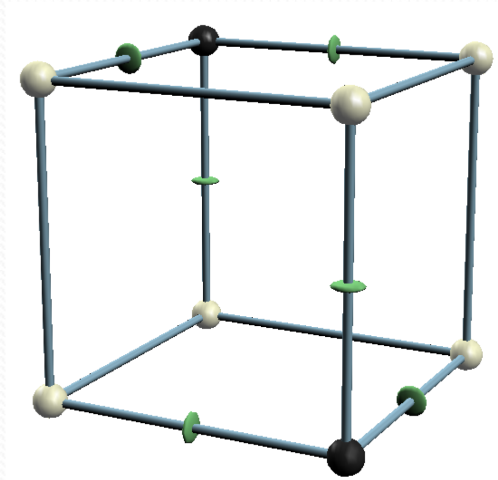
?

# Ambiguity

- Ambiguous cases:
  - Having an ambiguous face (2 white vertices and 2 black vertices in a diagonal ≈ all edges with a sign change)
  - With two white vertices (or two black vertices) in any of the cube diagonals

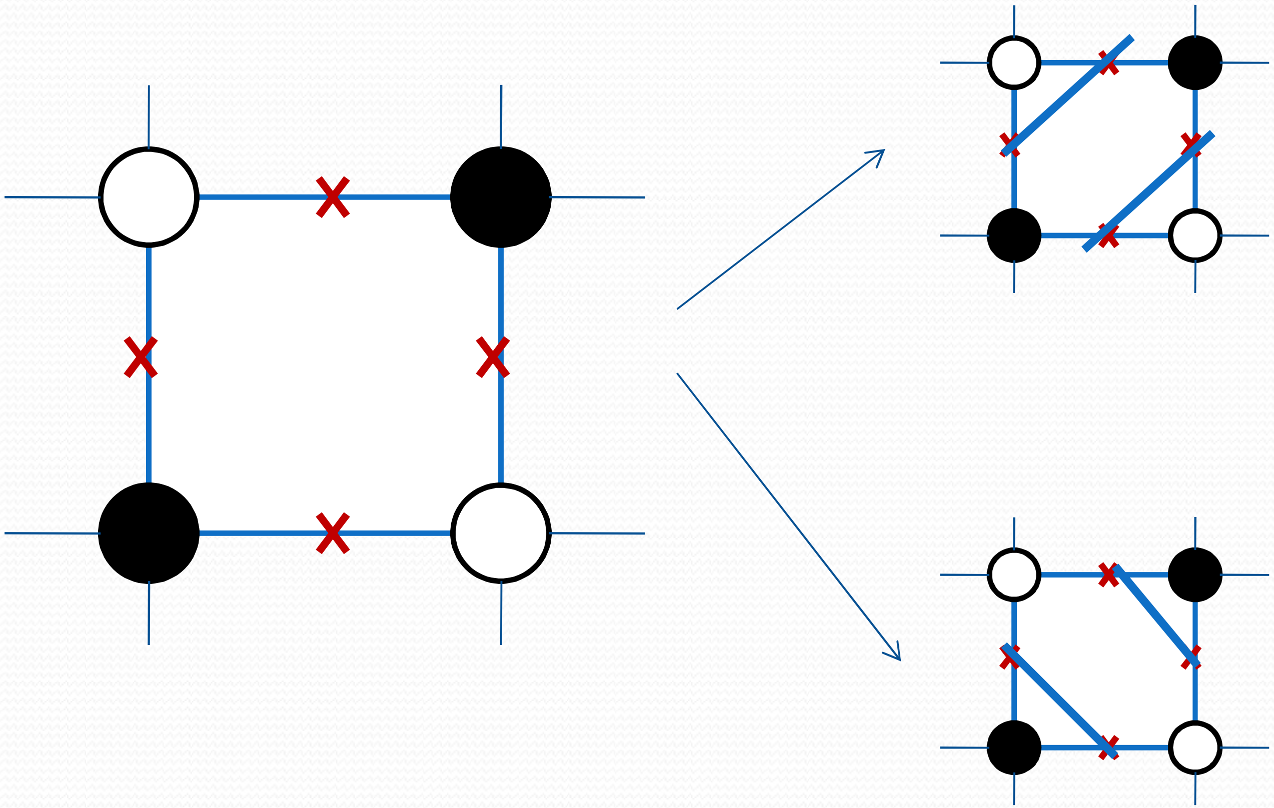






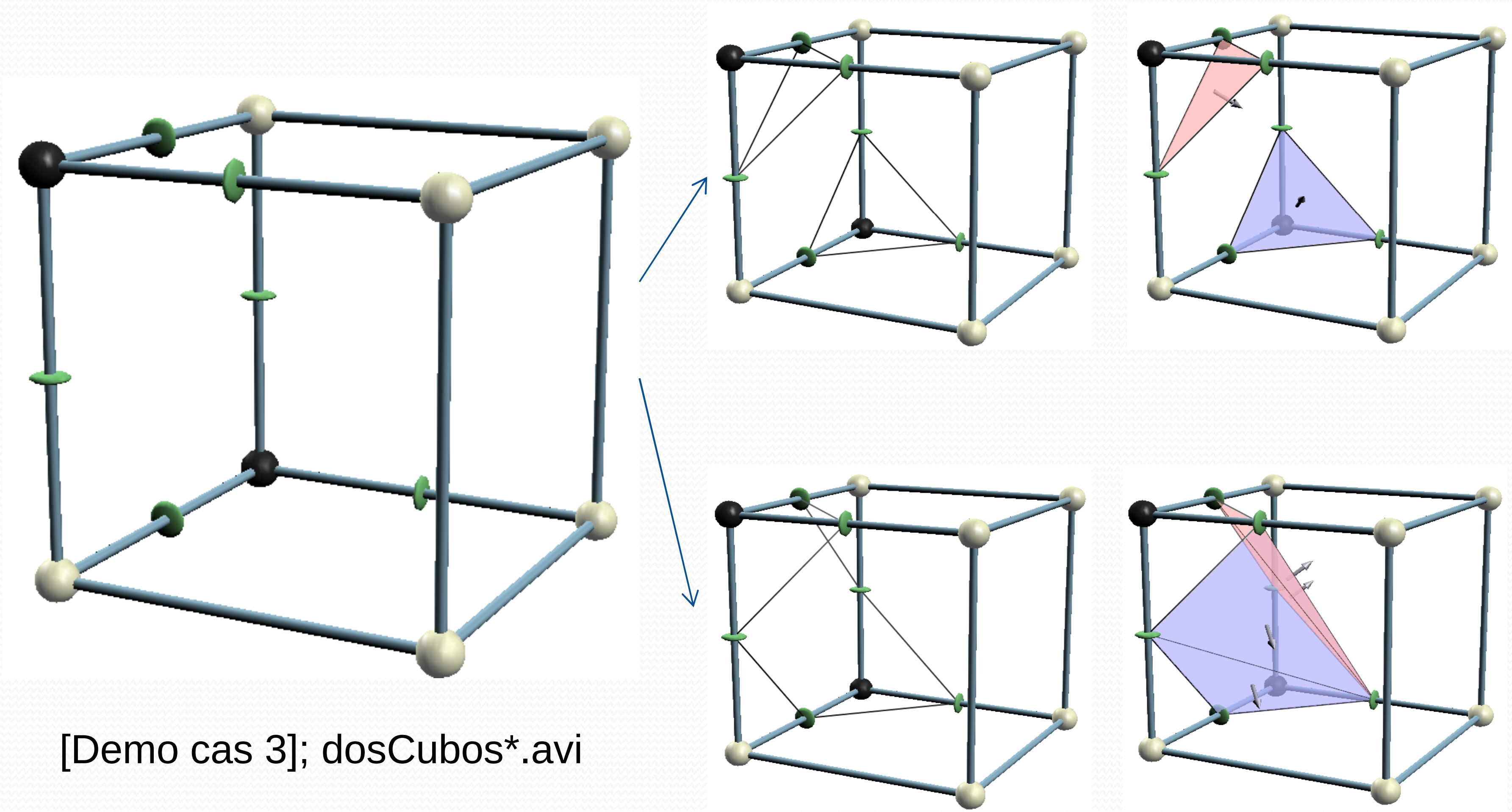
# Ambiguous faces

- Support two types of reconstruction:



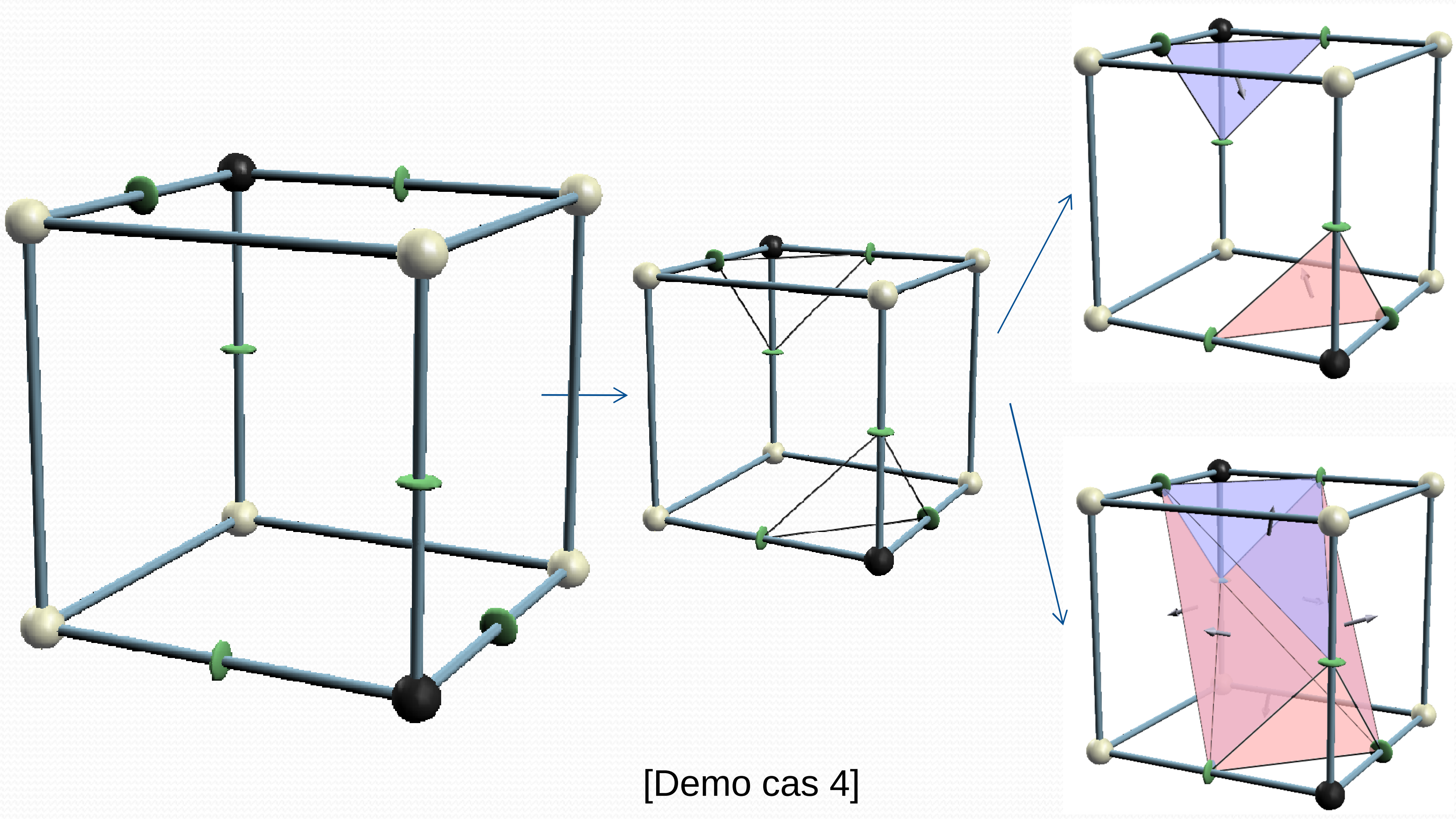
# Ambiguous faces

- Sample cube with an ambiguous face



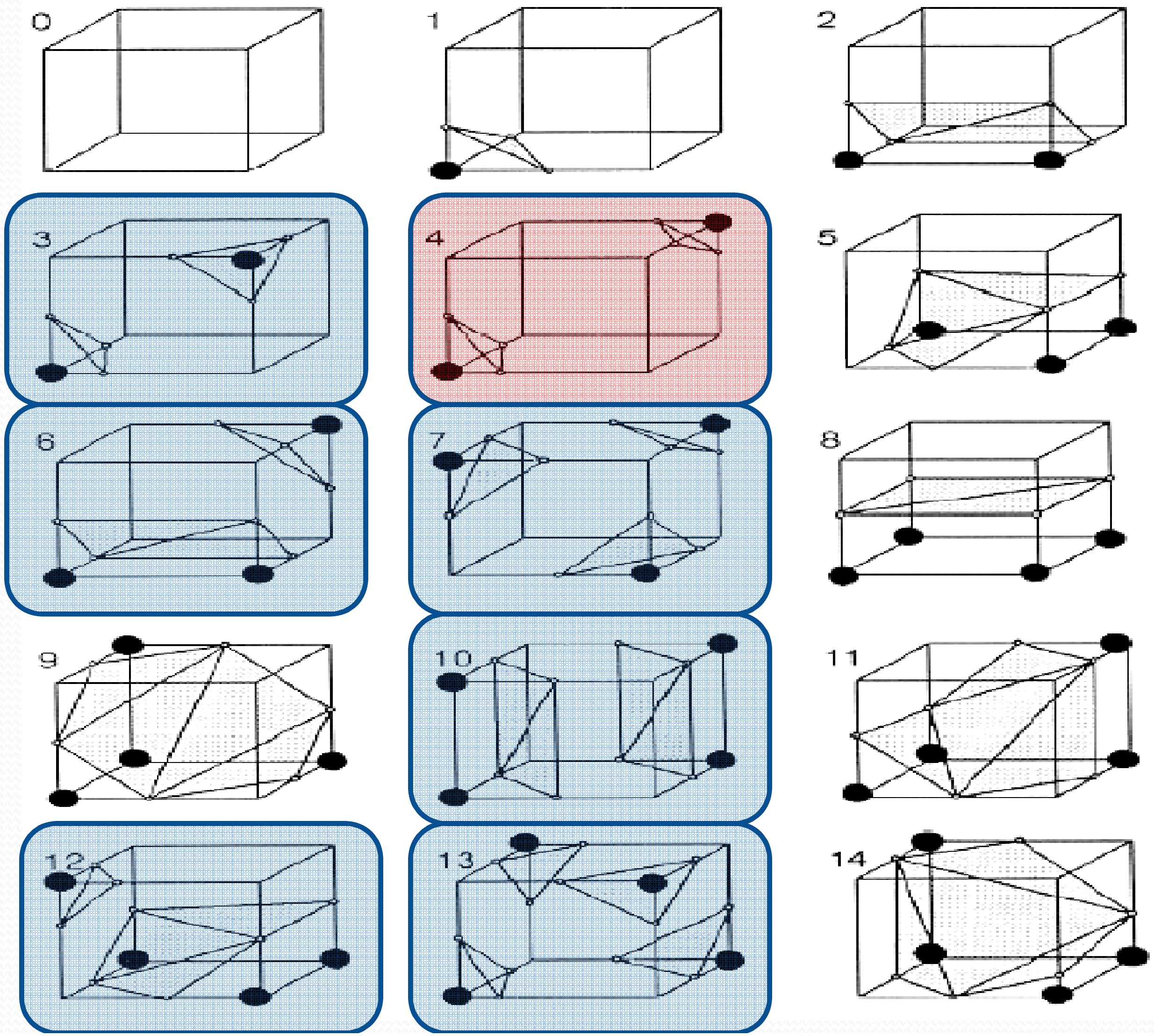
[Demo cas 3]; dosCubos*.avi

# Ambiguous cubes



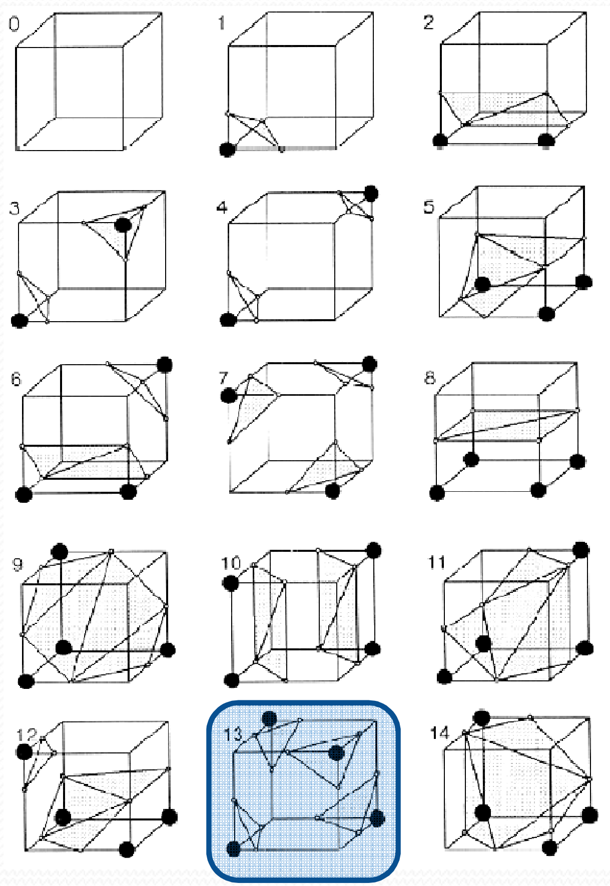
[Demo cas 4]

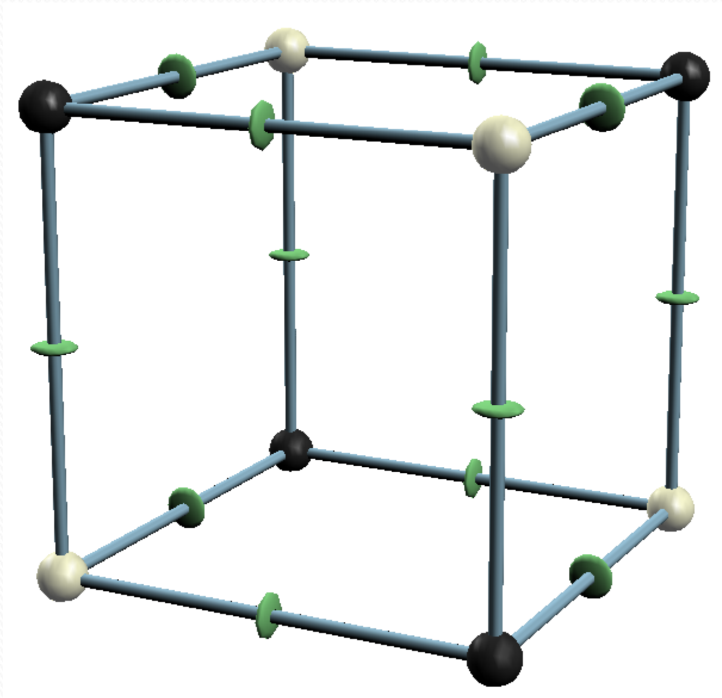
# Ambiguous cases



# Ambiguous cases

- Possible reconstructions for case 13 [demo]



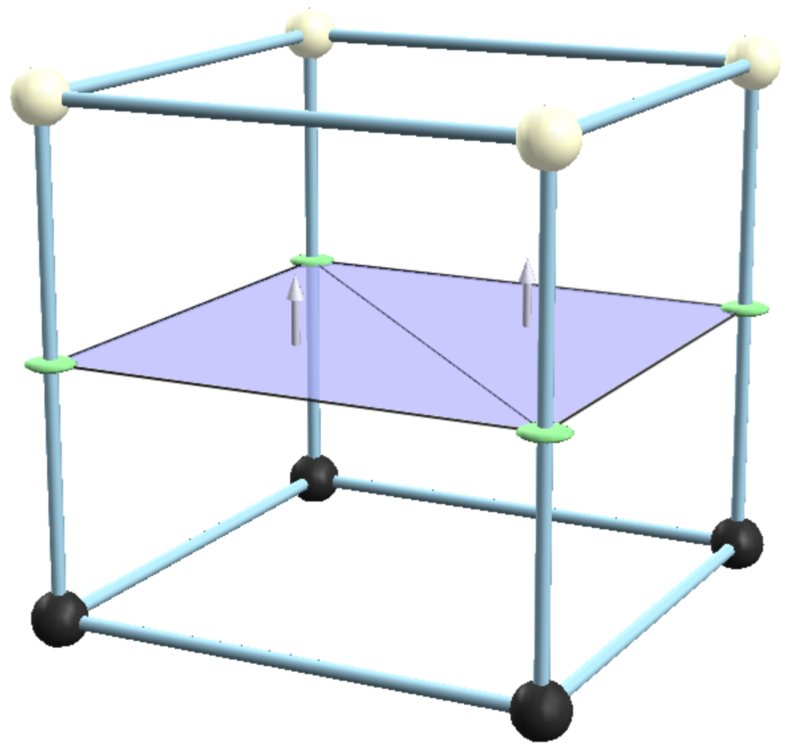
# Creating triangles

- Marching Cubes uses a LUT (look-up table) with 256 entries (one per configuration), which indicates how to build the triangles inside the cube:
  - Number of triangles
  - For each triangle:
    - Indices (a,b,c) of the vertices of the triangle. Each index is a value 0..11 indicating the edge of the cube containing the vertex.

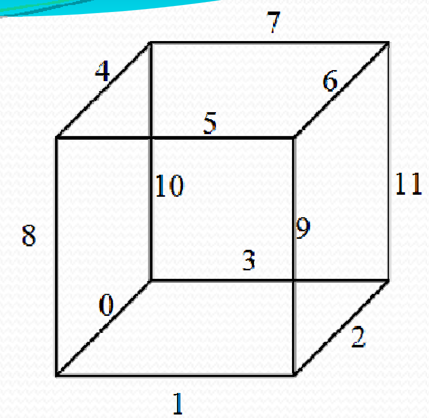
| | |
|---|---|
| 0 | Triangles for config 0 |
| 1 | Triangles for config 1 |
| 2 | Triangles for config 2 |
| ... | |
| 255 | Triangles for config 255 |

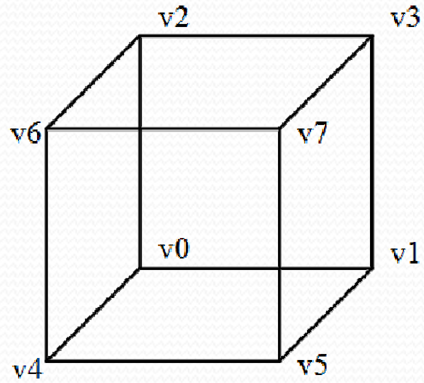
# Creating triangles

- Example:



0011 0011 → Config 51

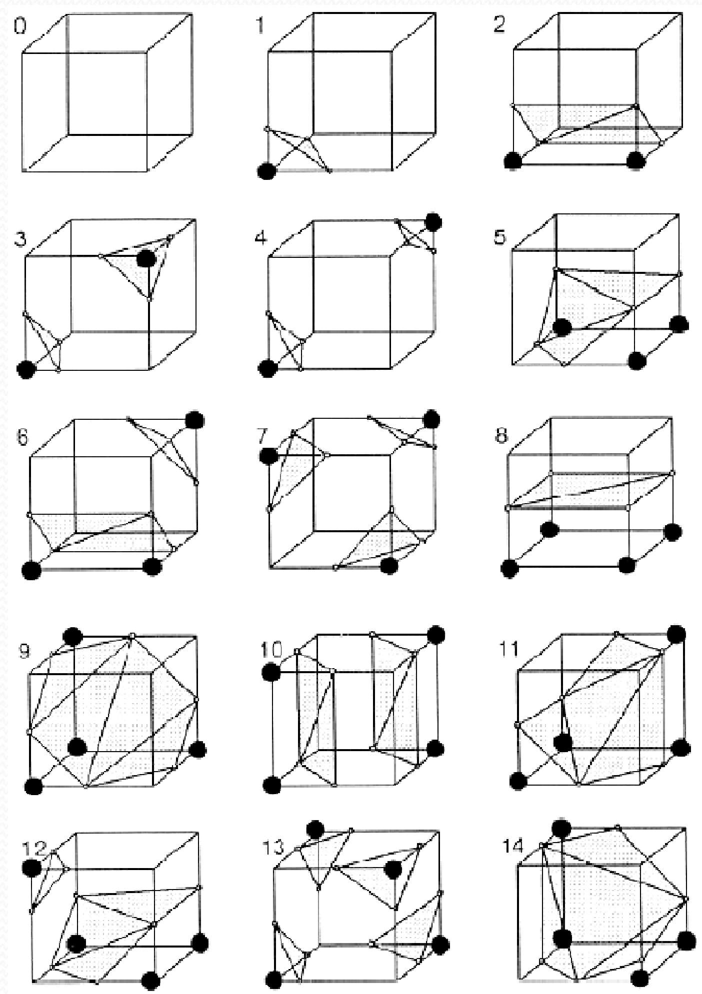


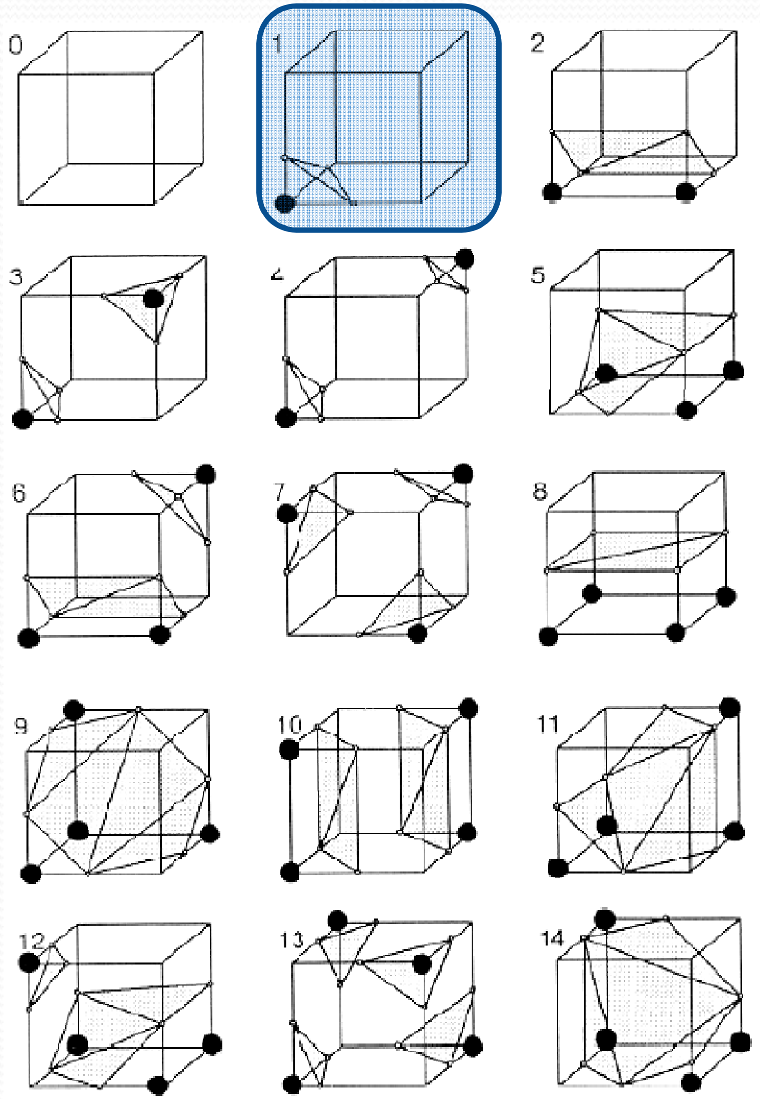

| | |
|---|---|
| 0 | Triangles config 0 |
| 1 | Triangles config 1 |
| … | |
| **51** | **{ {8, 9, 10}, {9, 11, 10} }** |
| … | |

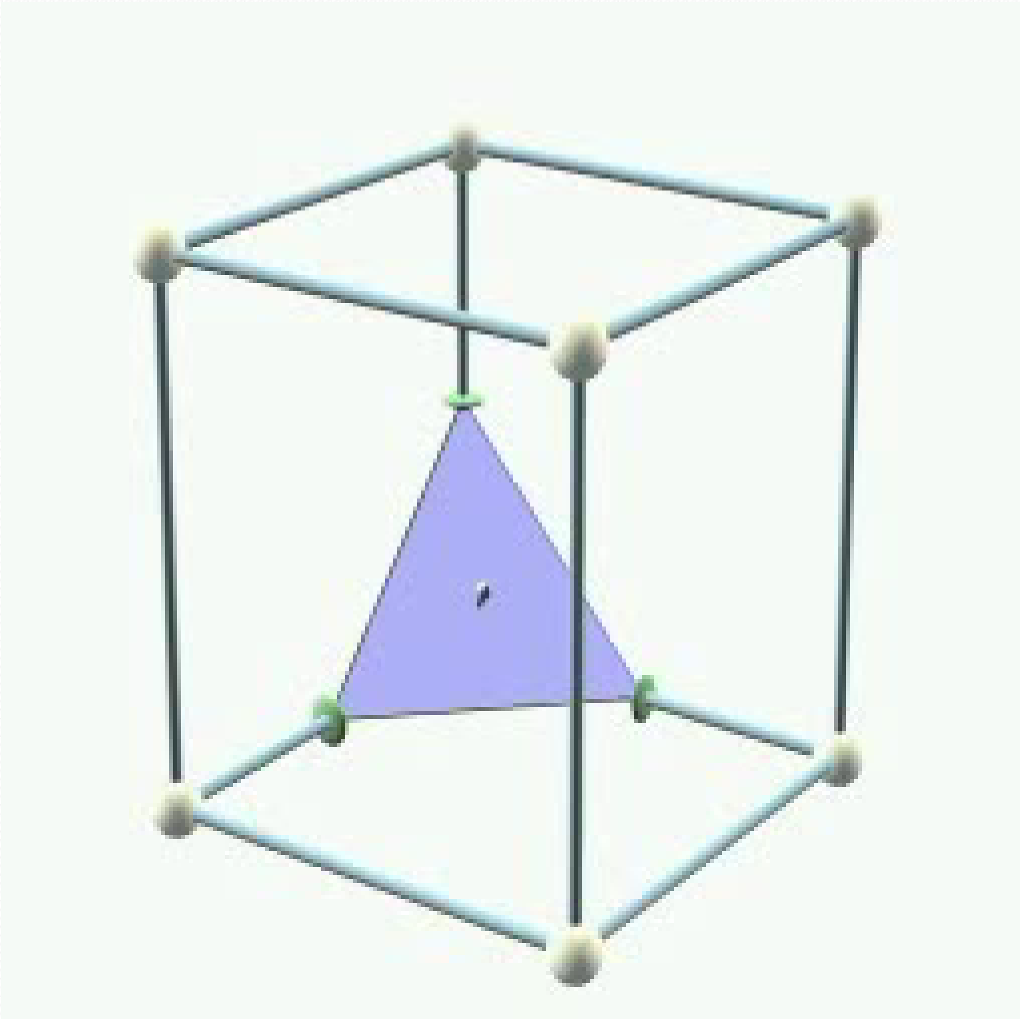
# LUT

- Case analysis (14+1 cases)
- For each ambiguous case, the authors chose the simplest reconstruction.
- Each case → 1 - 4 triangles

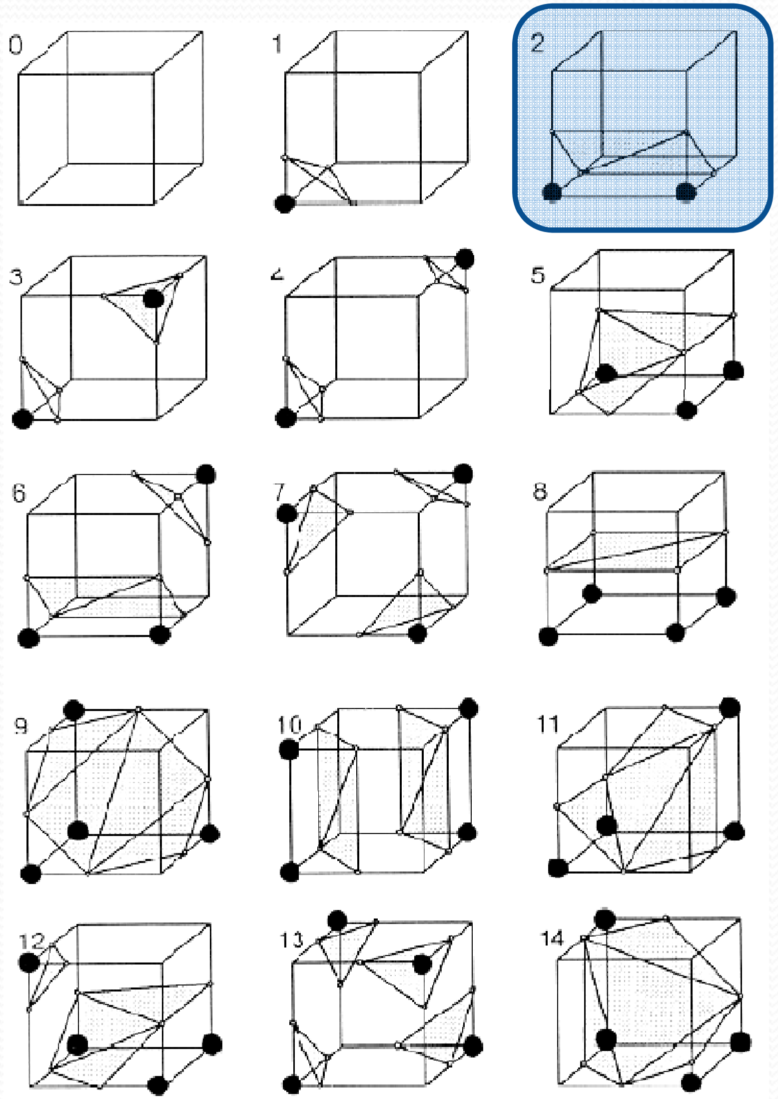


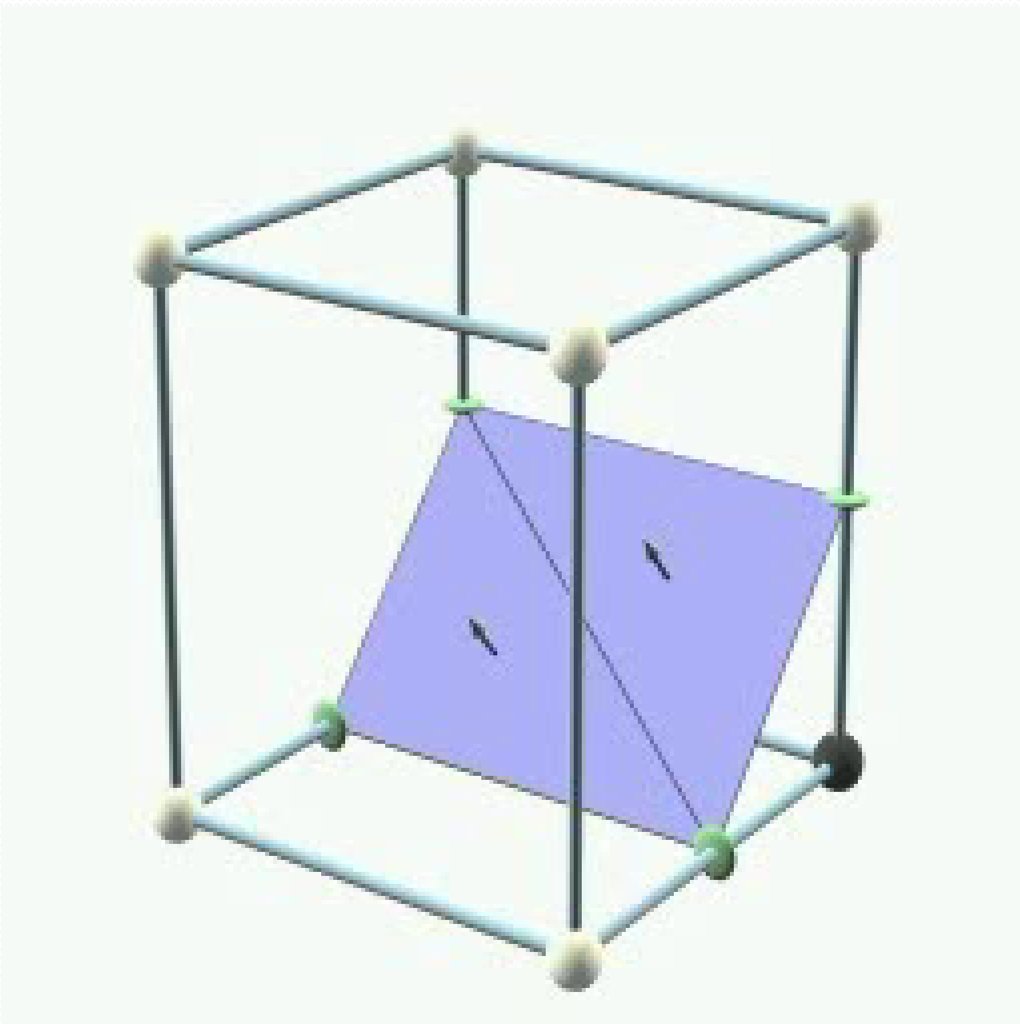
# Reconstruction case 1



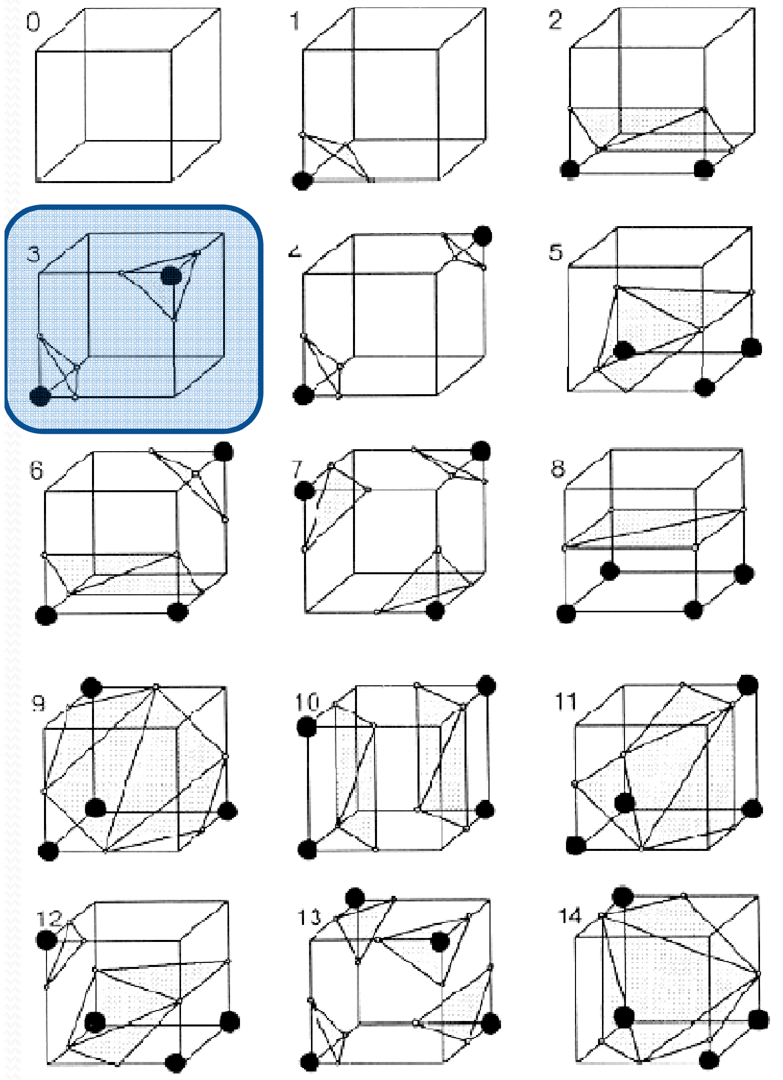



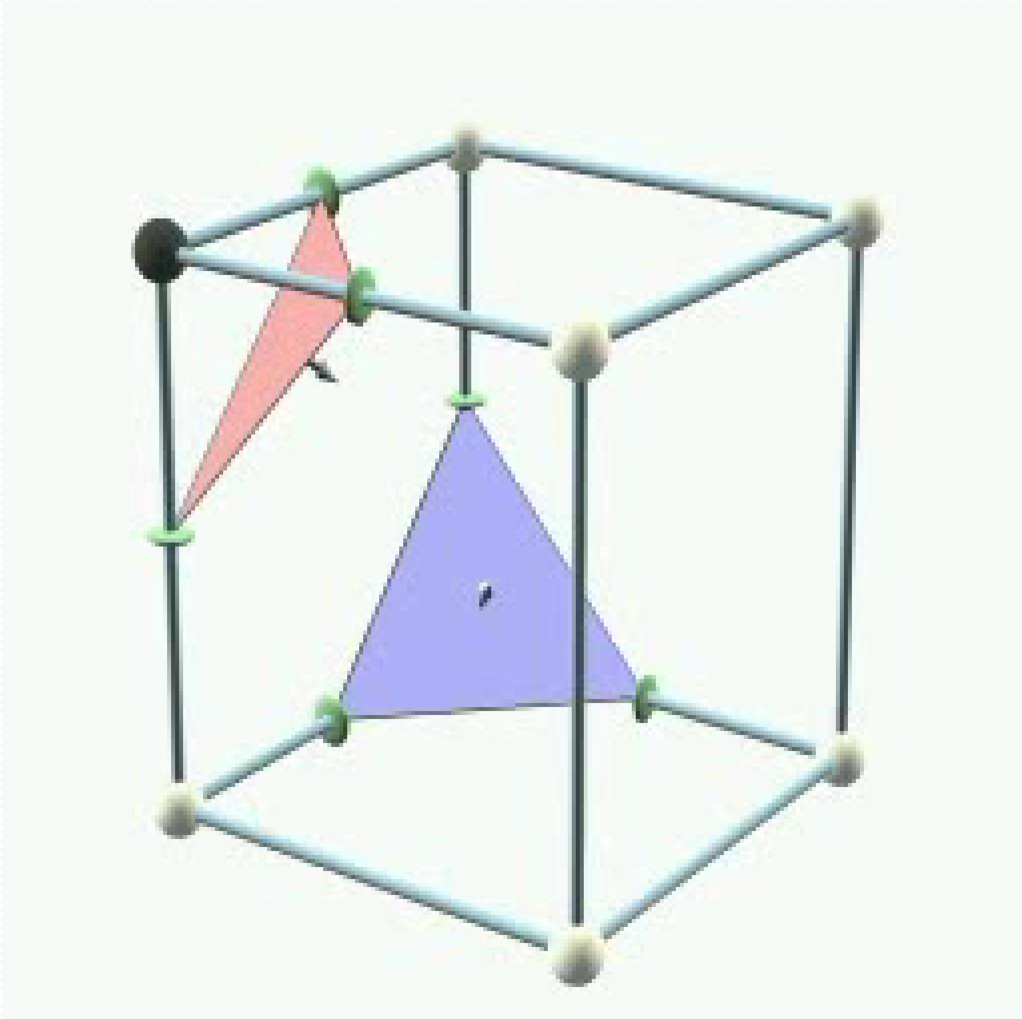
# Reconstruction case 2



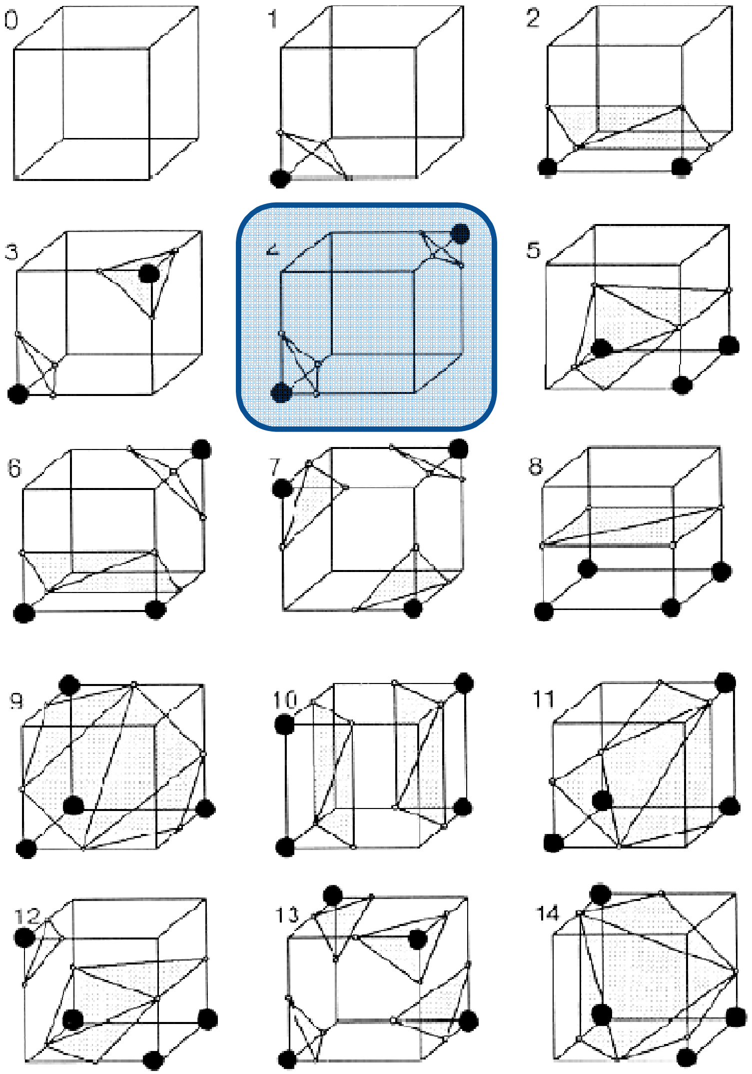



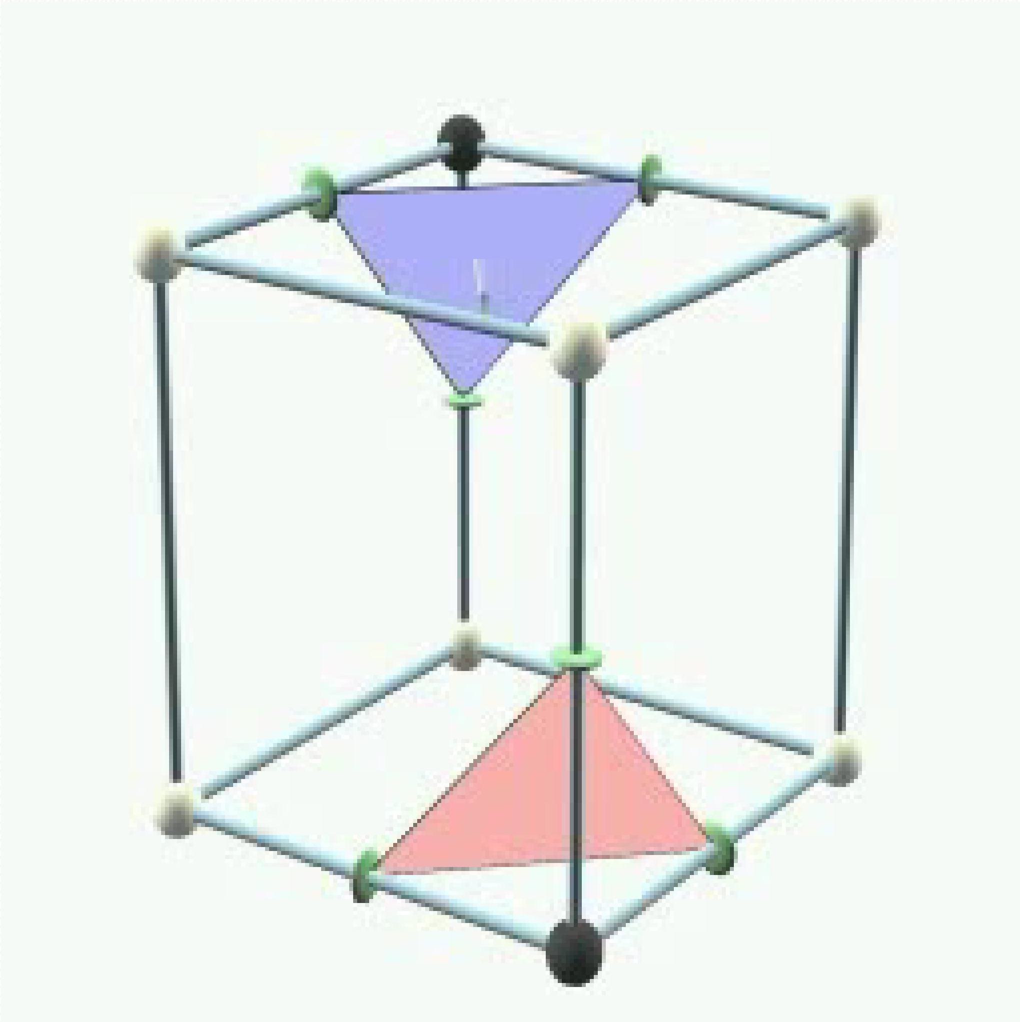
# Reconstruction case 3



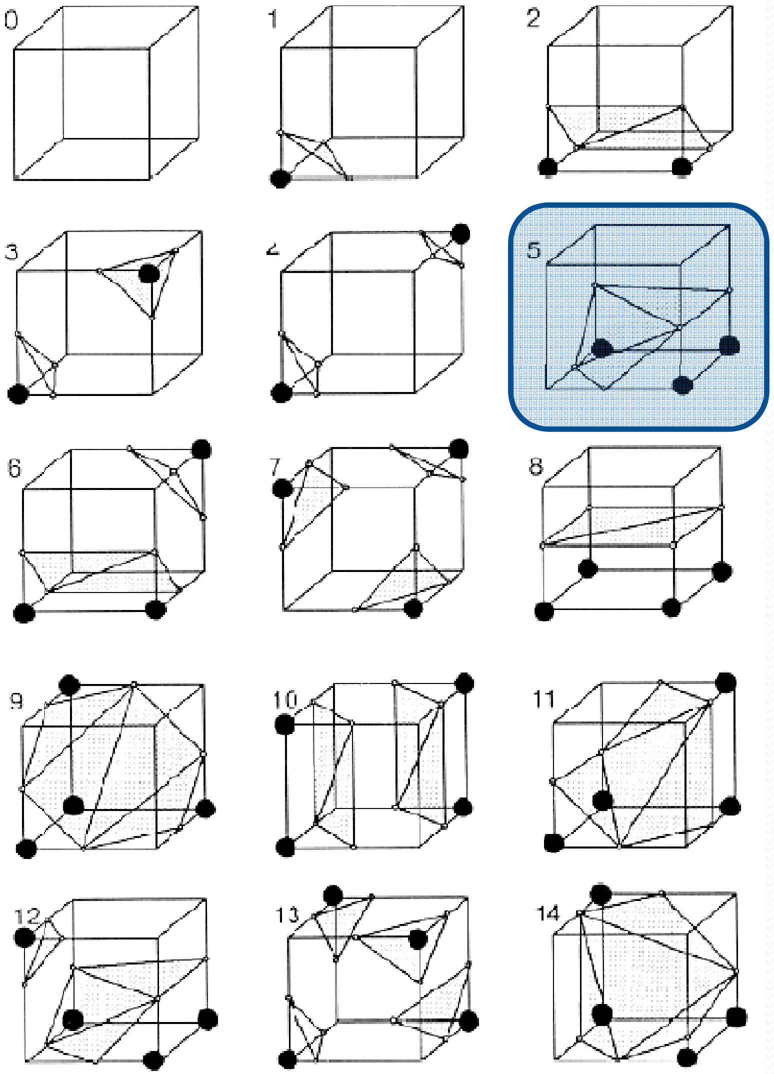



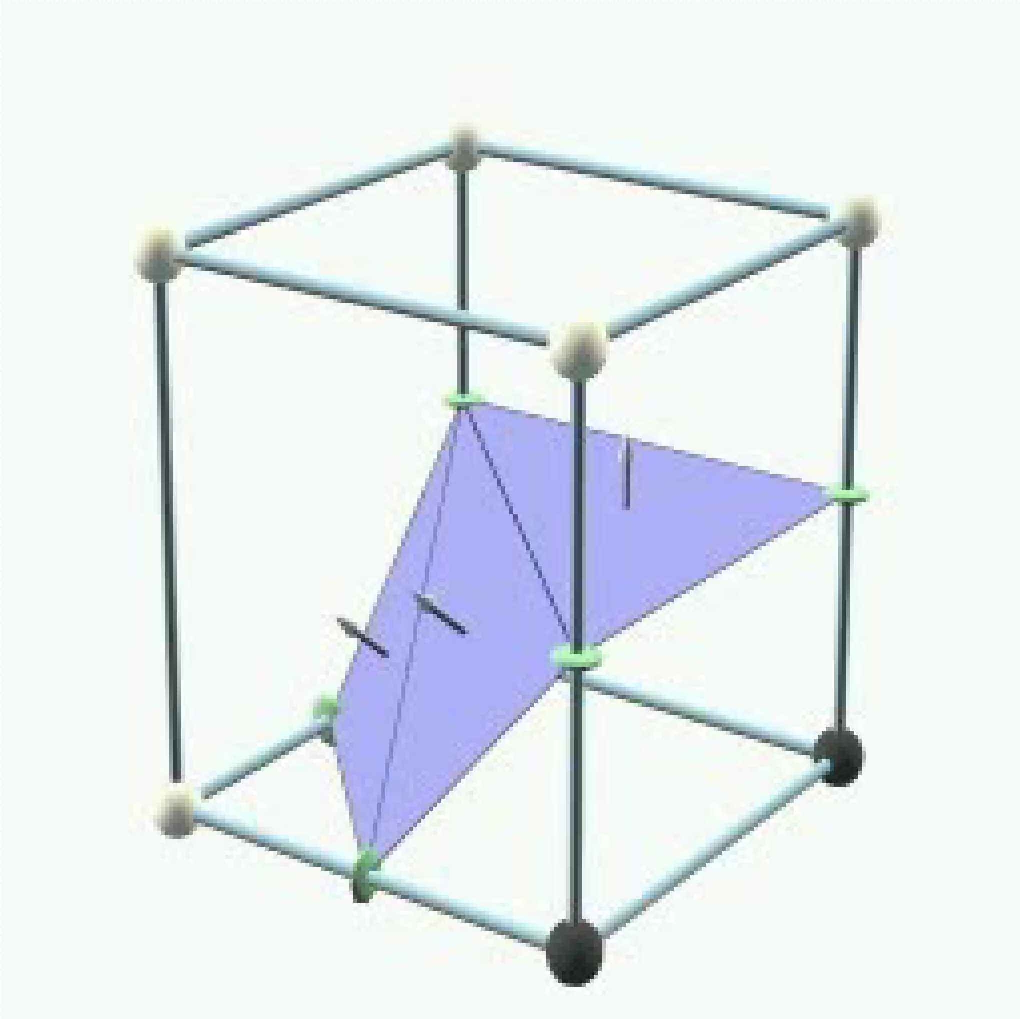
# Reconstruction case 4



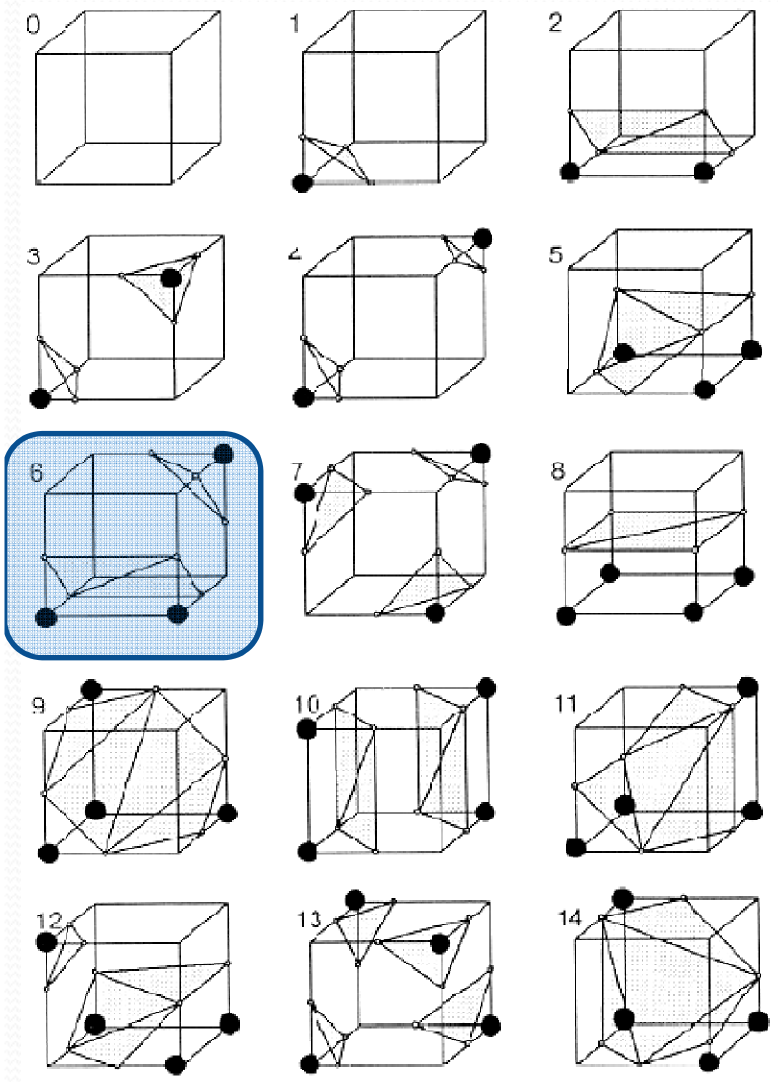



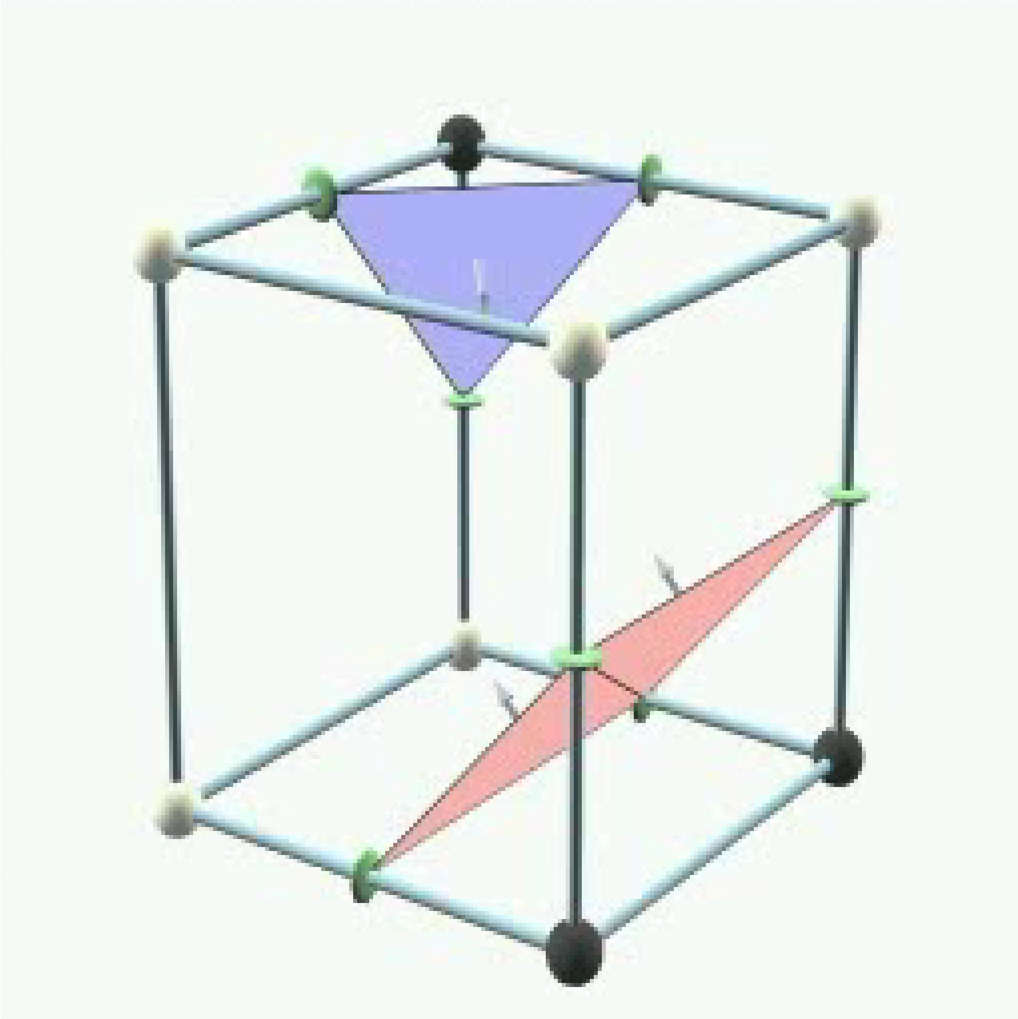
# Reconstruction case 5



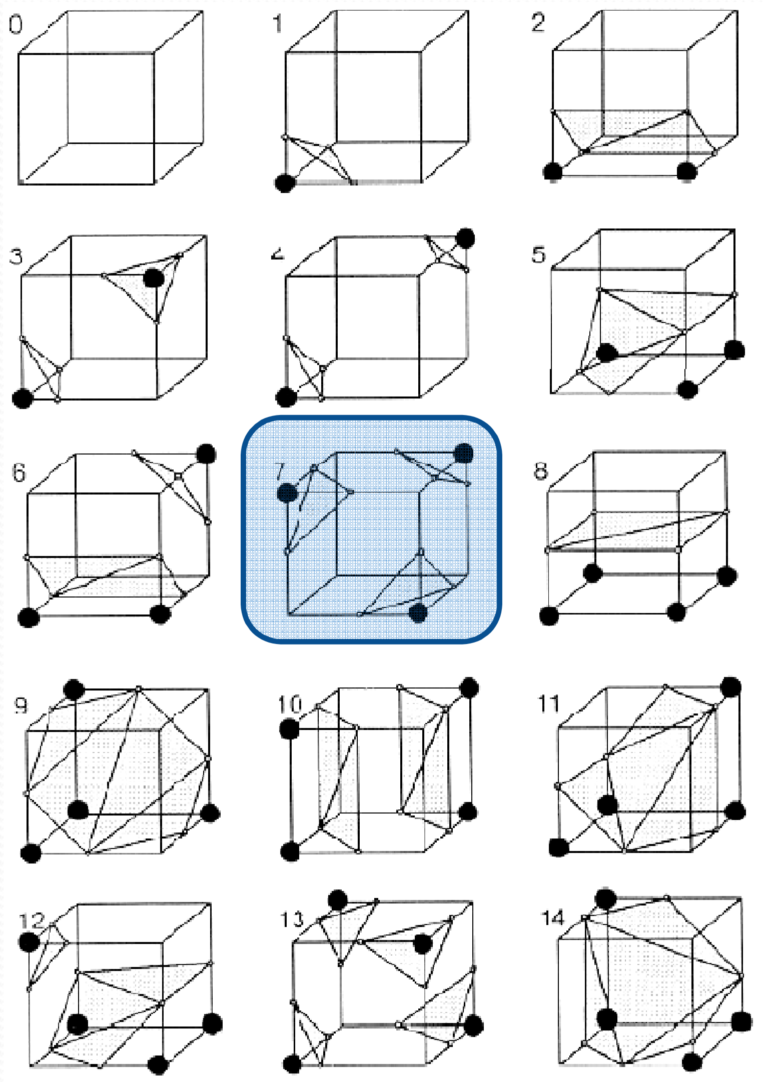



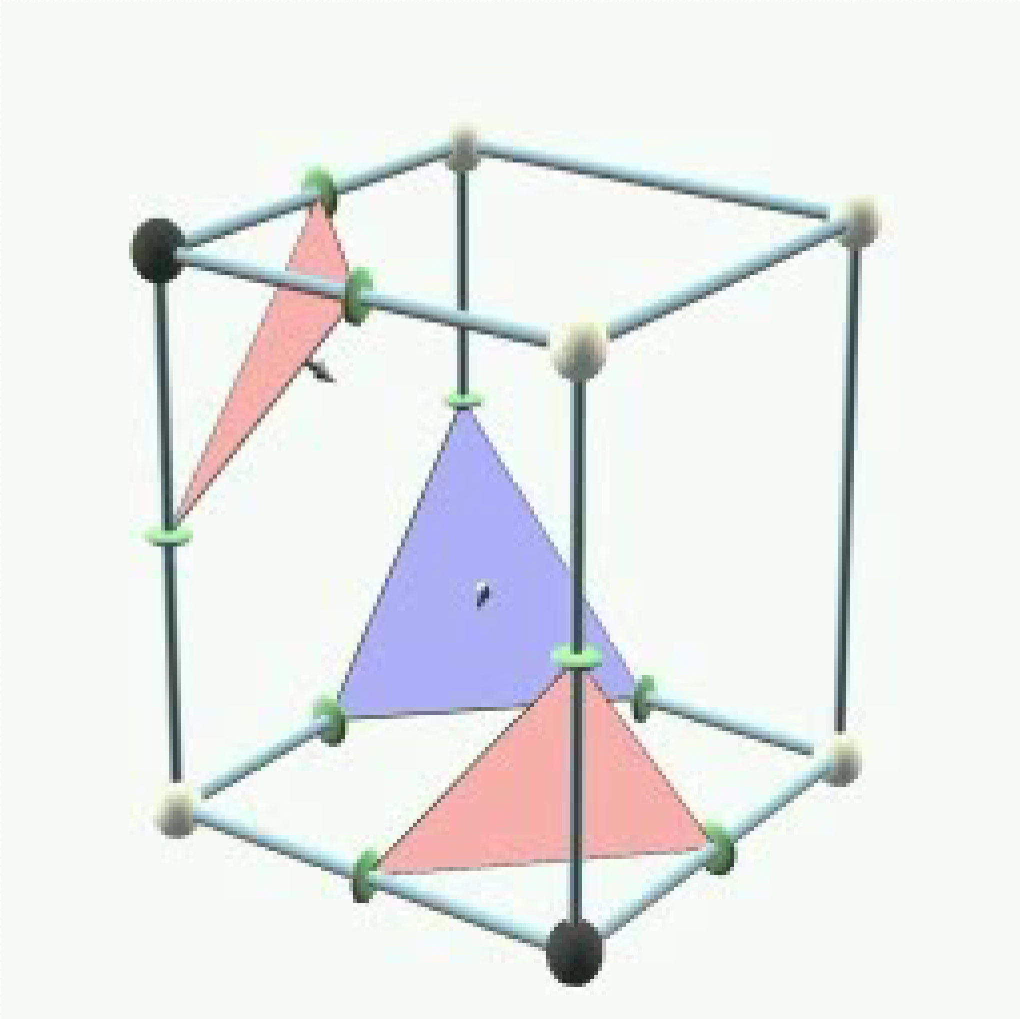
# Reconstruction case 6



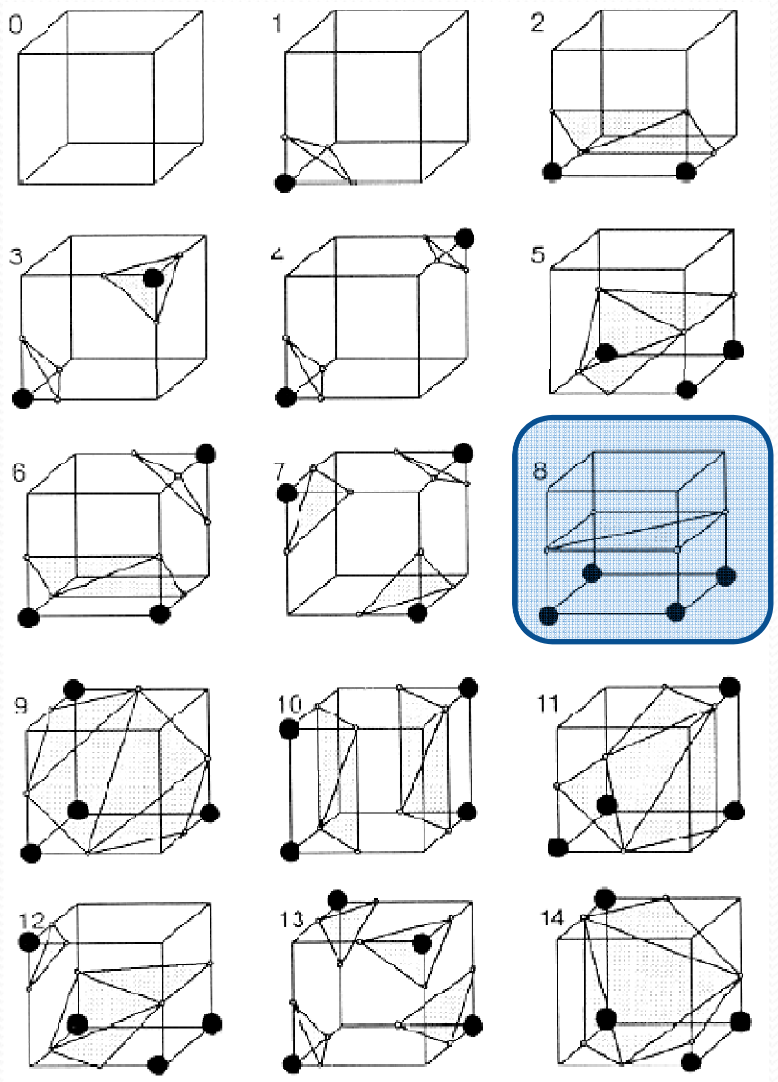



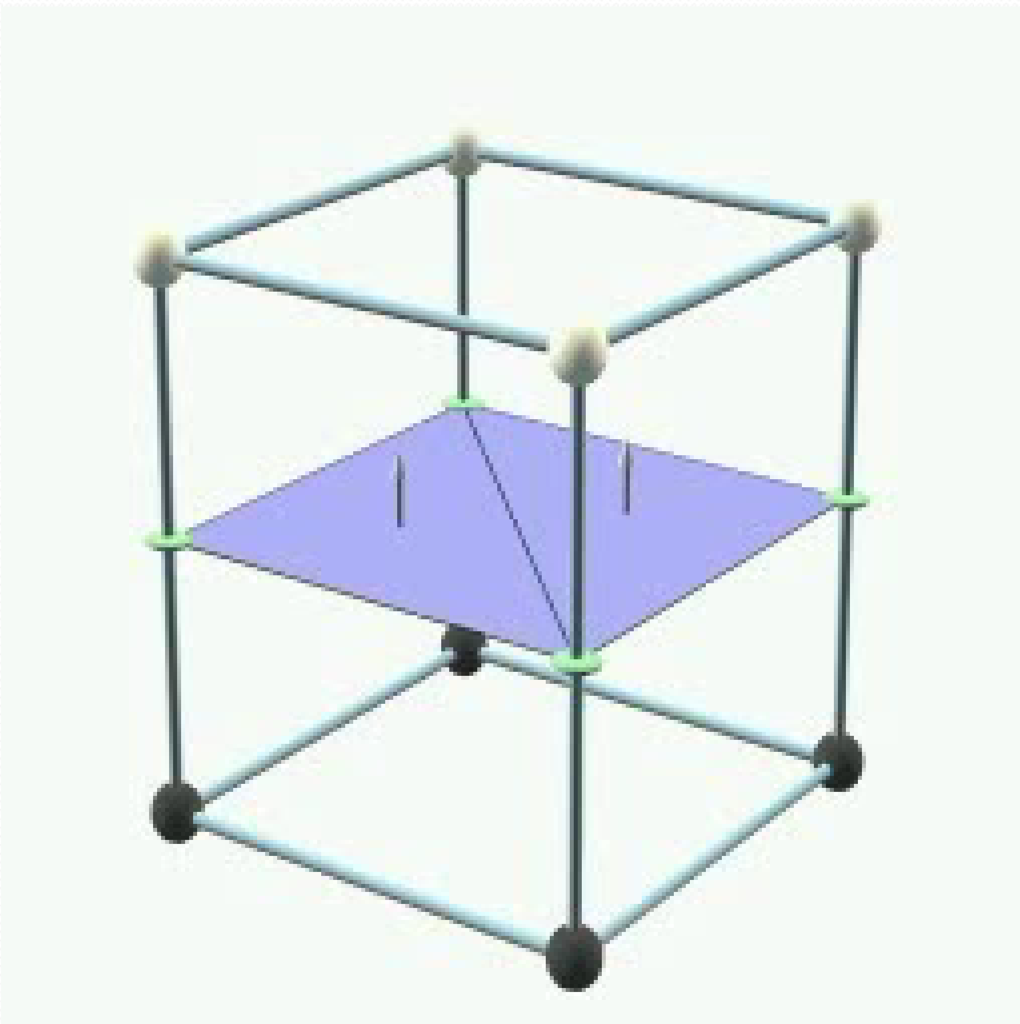
# Reconstruction case 7



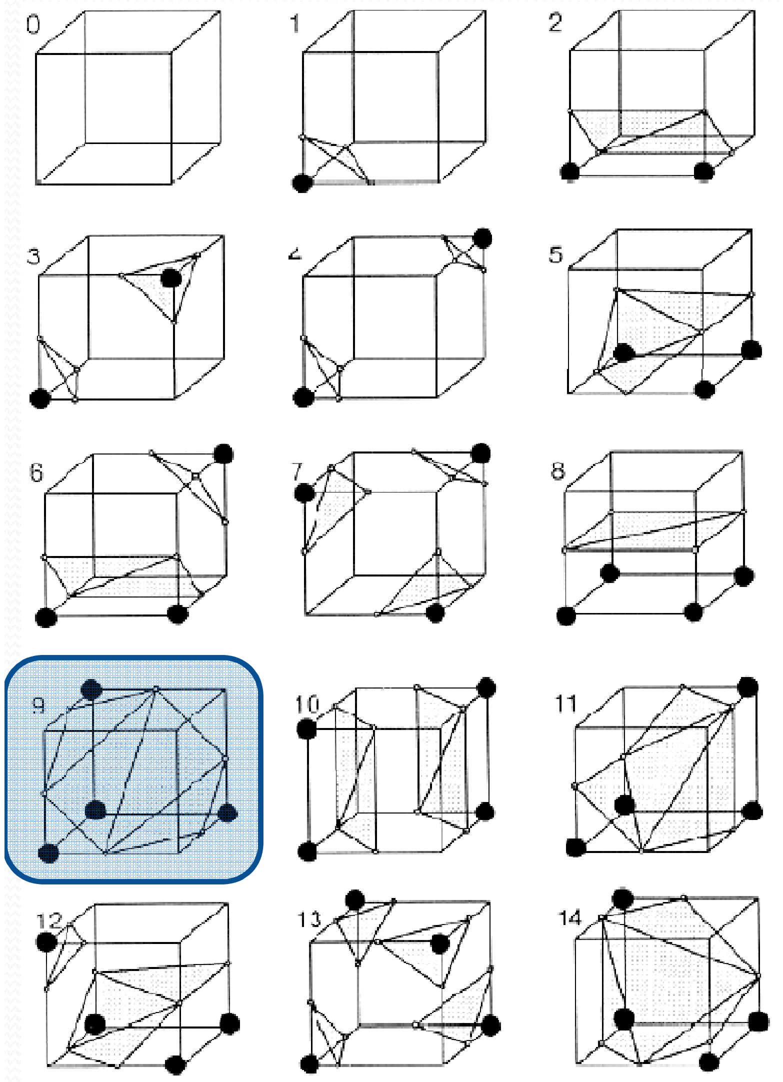



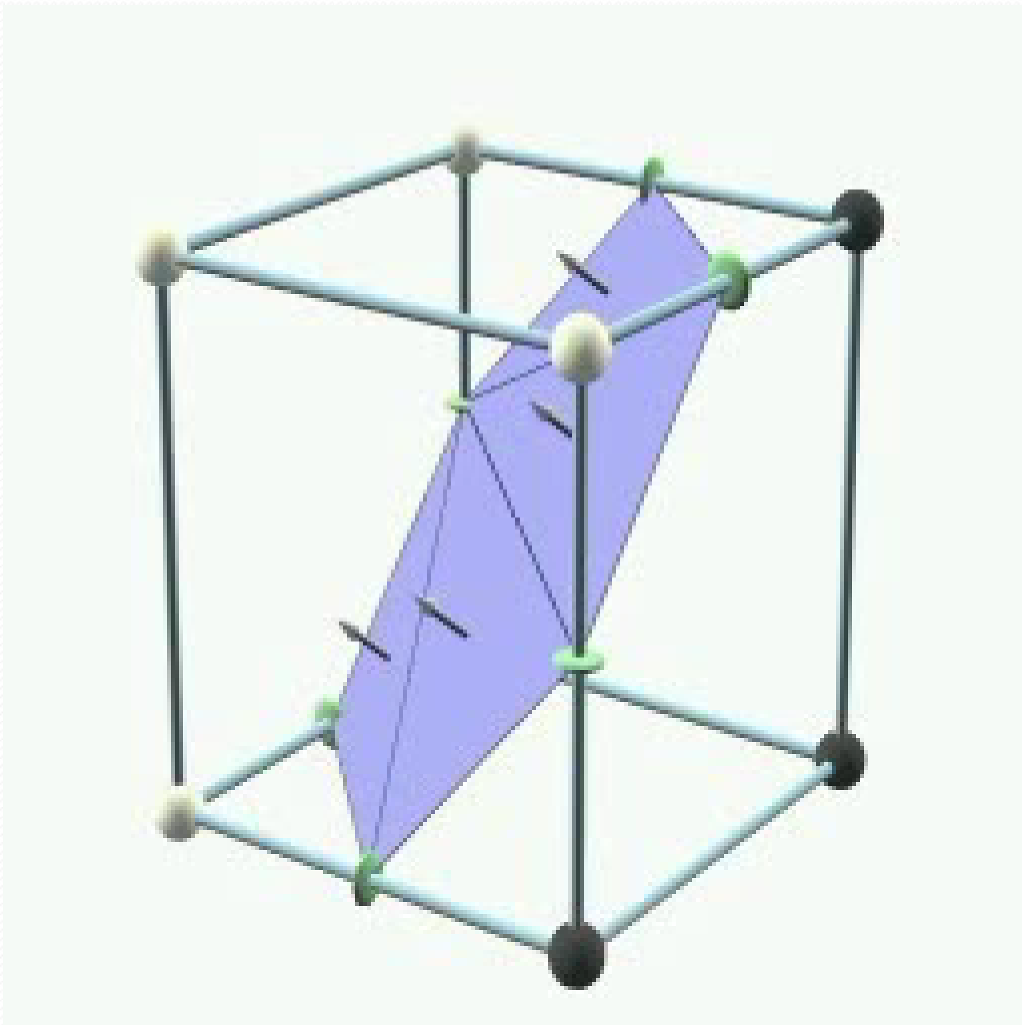
# Reconstruction case 8



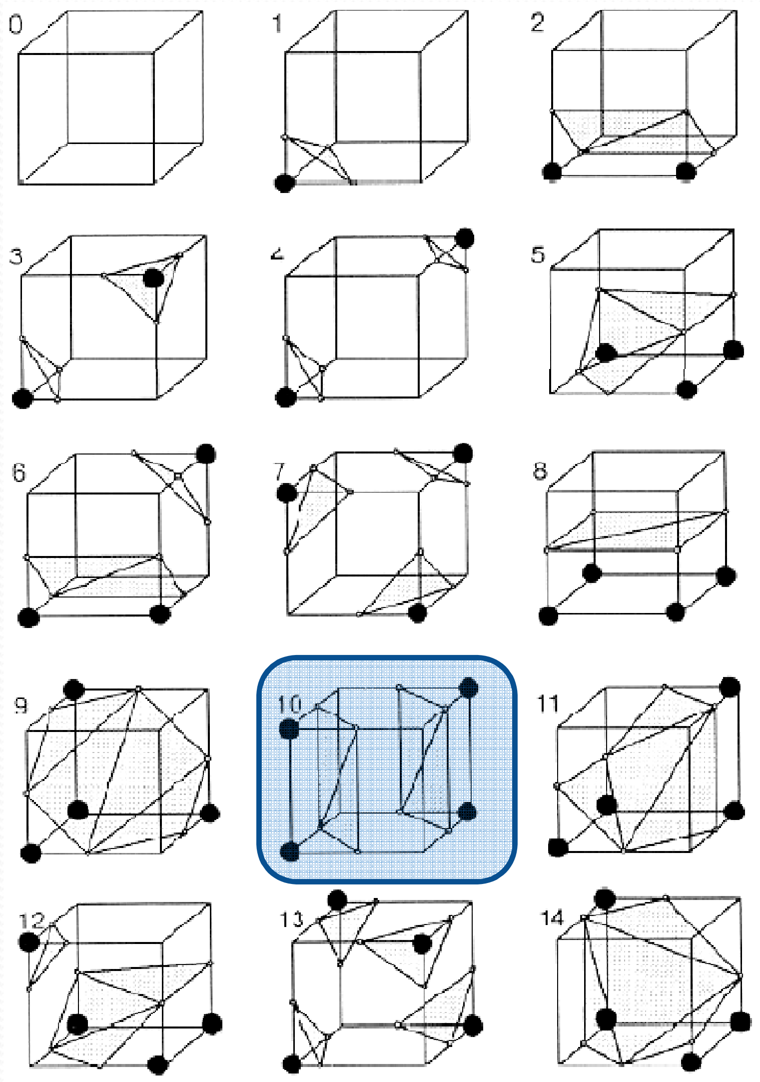



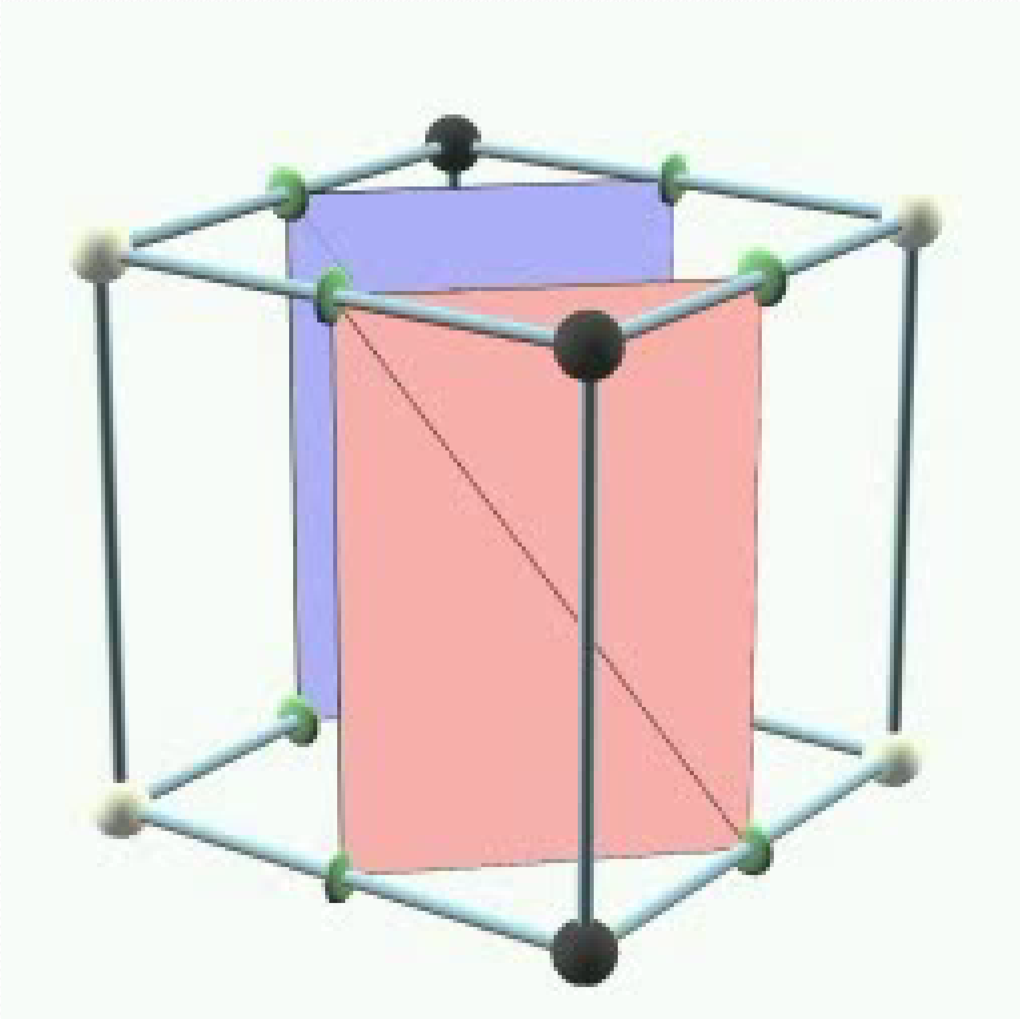
# Reconstruction case 9



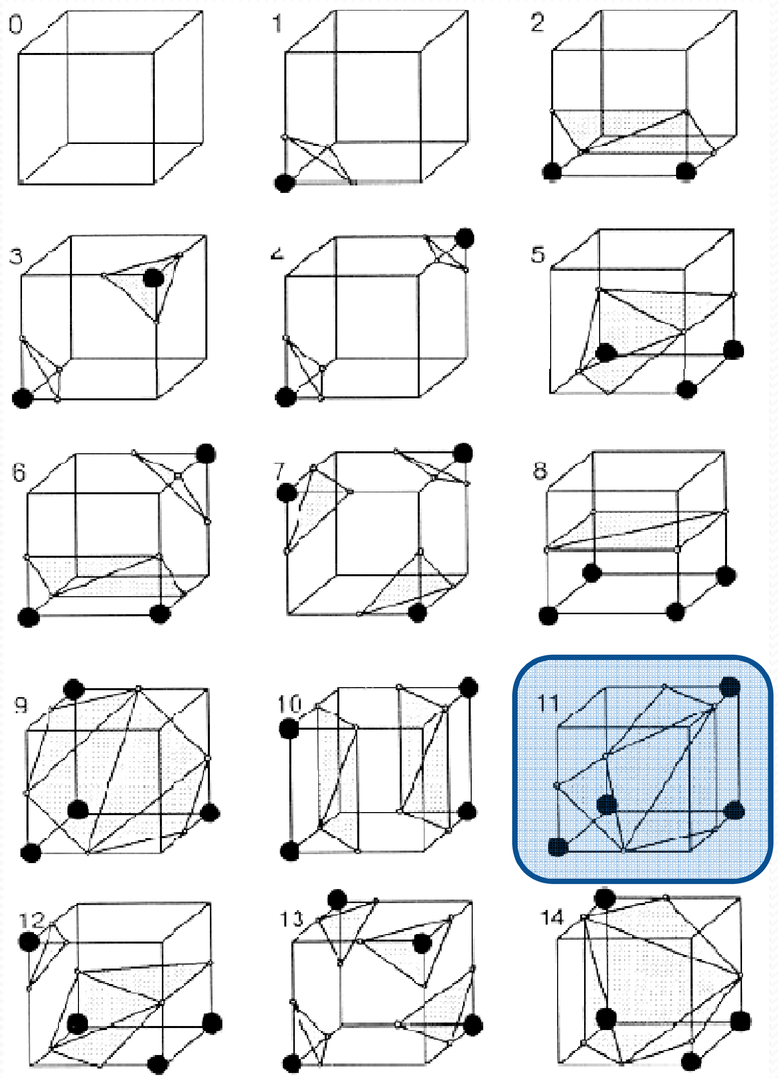



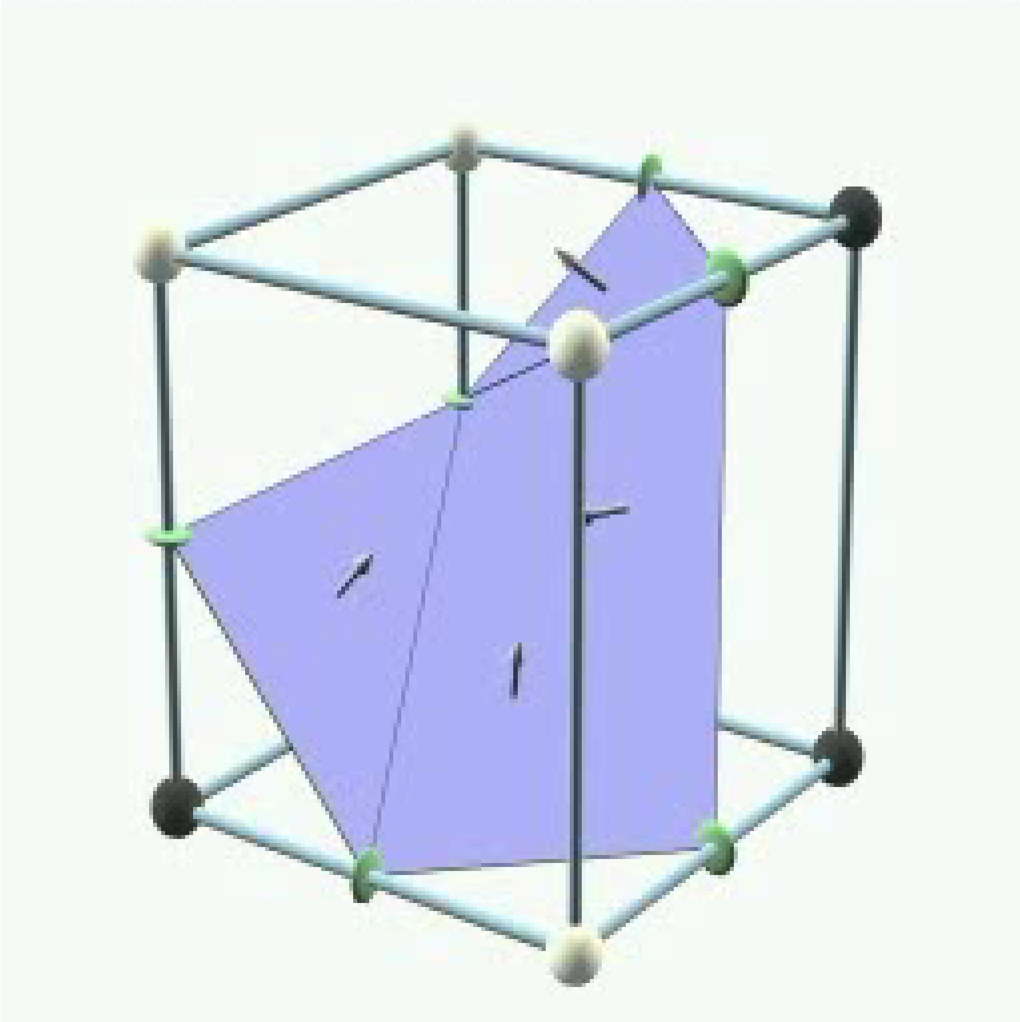
# Reconstruction case 10



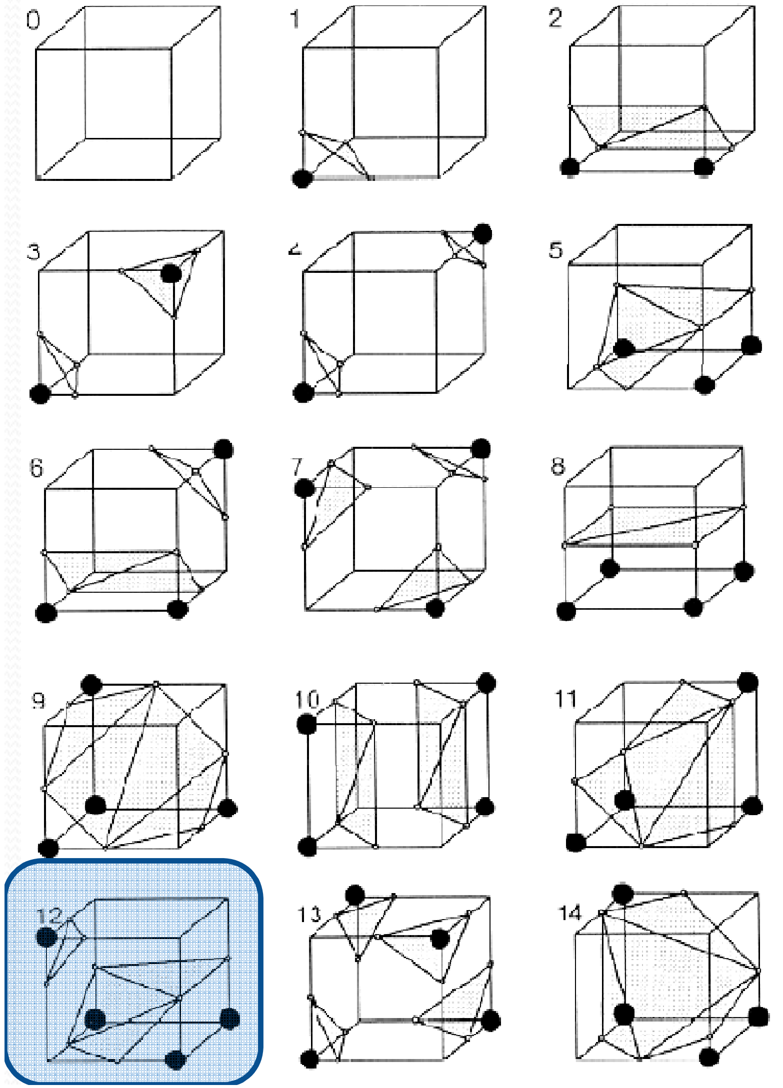



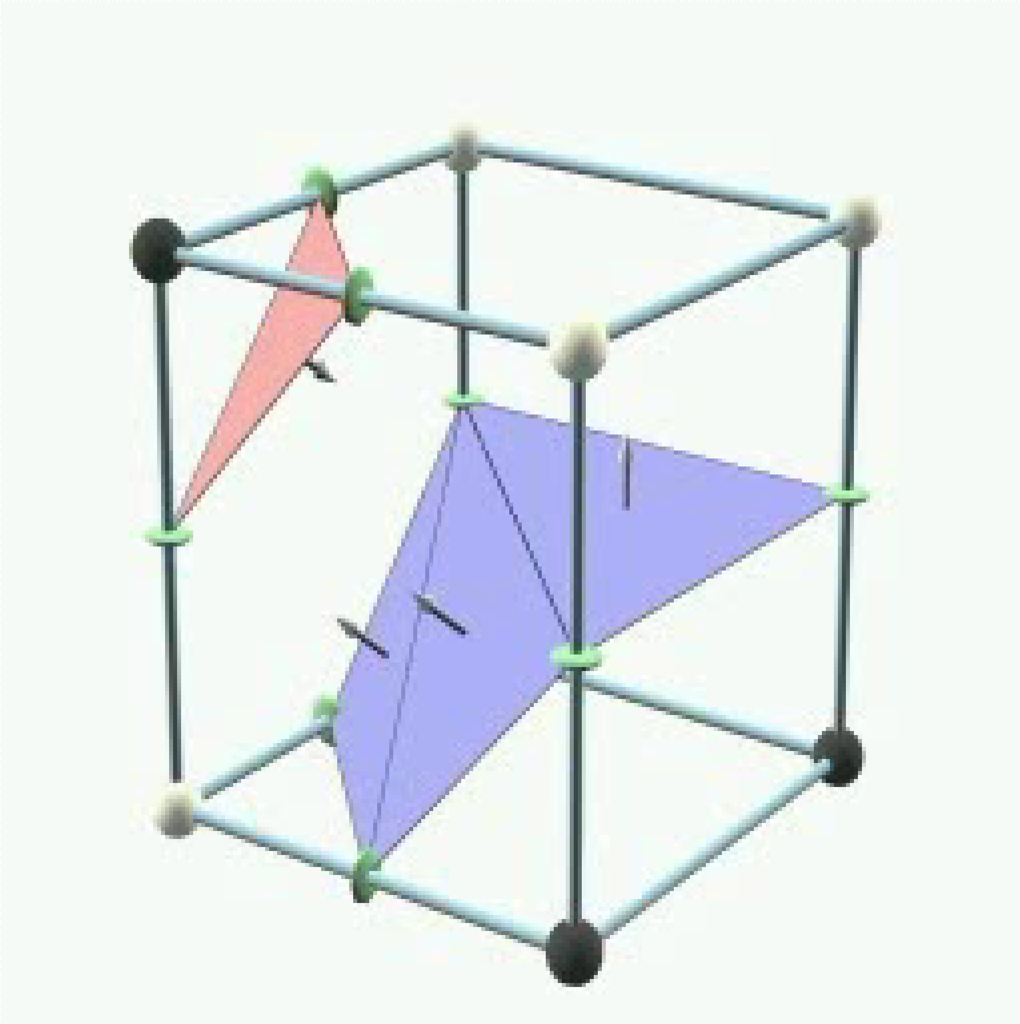
# Reconstruction case 11



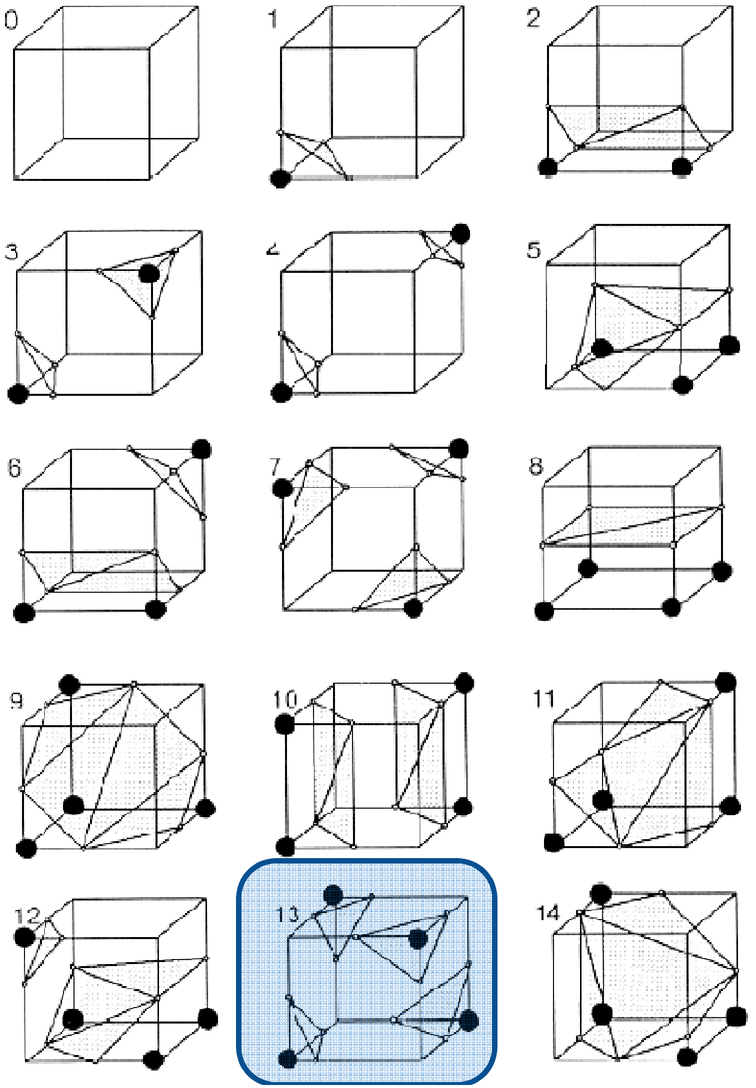



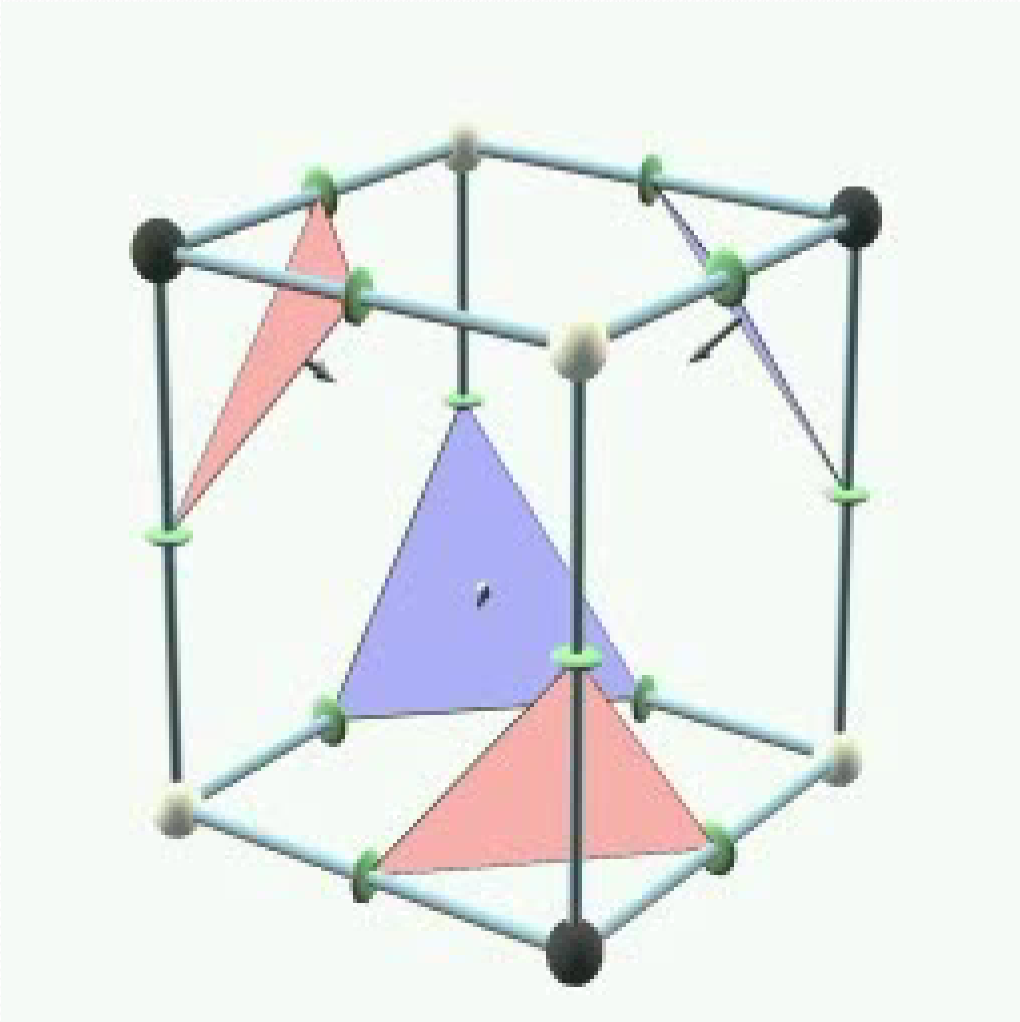
# Reconstruction case 12



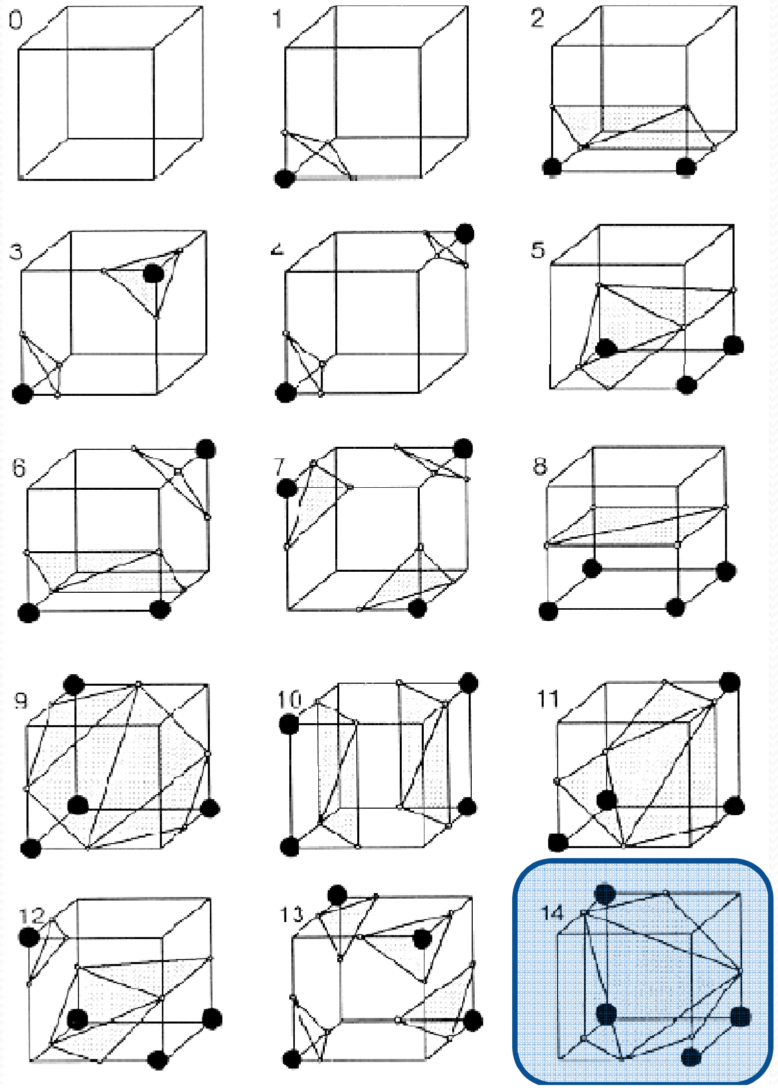



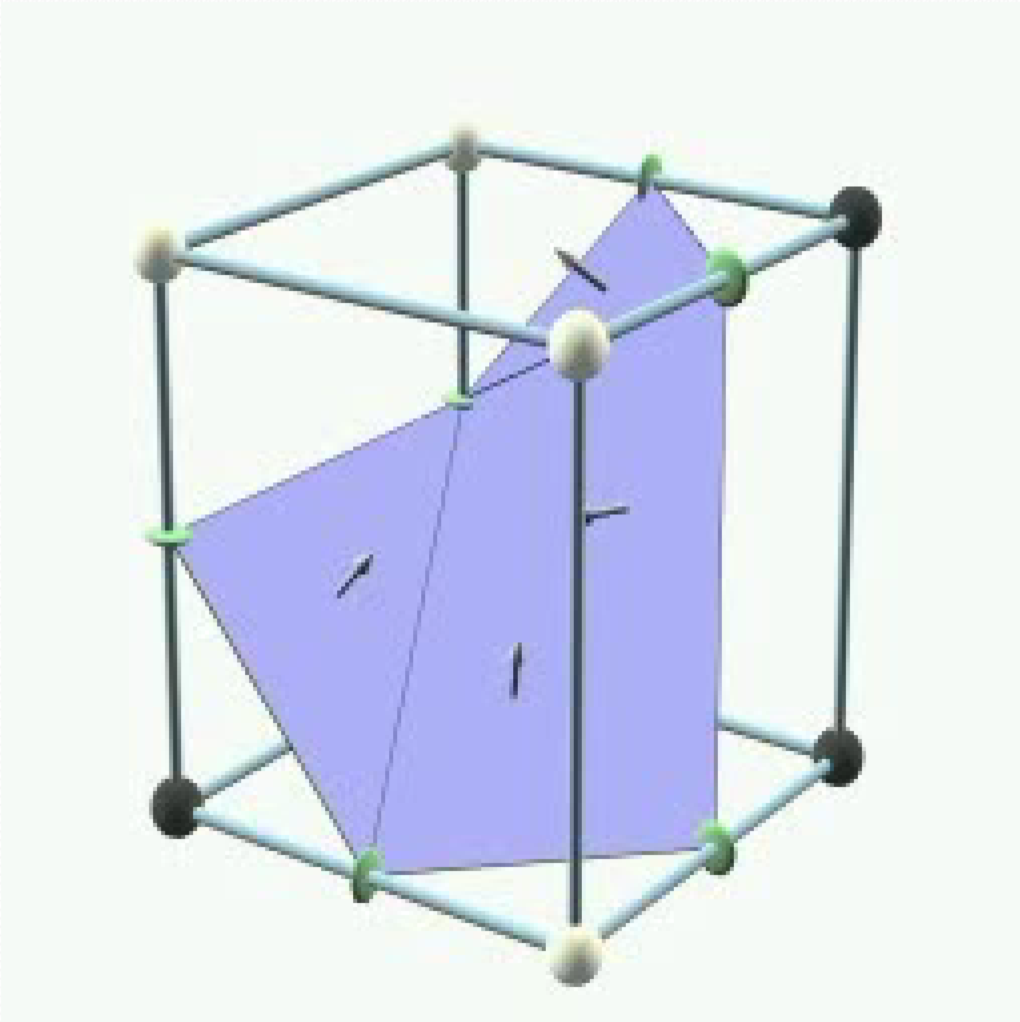
# Reconstruction case 13



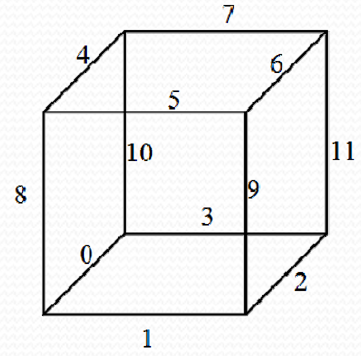



# Reconstruction case 14



# Marching Cubes




For each sample (i,j,k)

config := get configuration of the cube (i,j,k) → (i+1, j+1, k+1)

recons := LUT[config] // query precomputed LUT

// recons is e.g. {{a,b,c}, {d,e,f}...}

create vertices through linear interpolation

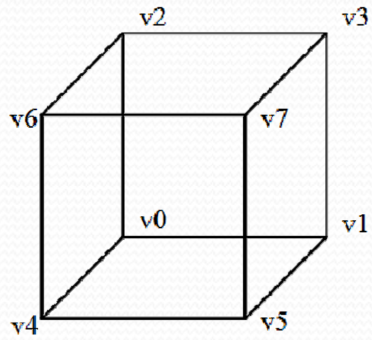
// creat a vertex for each edge in (a, b, c, d...)

create the triangles as indicated by the LUT

// create the triangles (a,b,c), (d,e,f),...

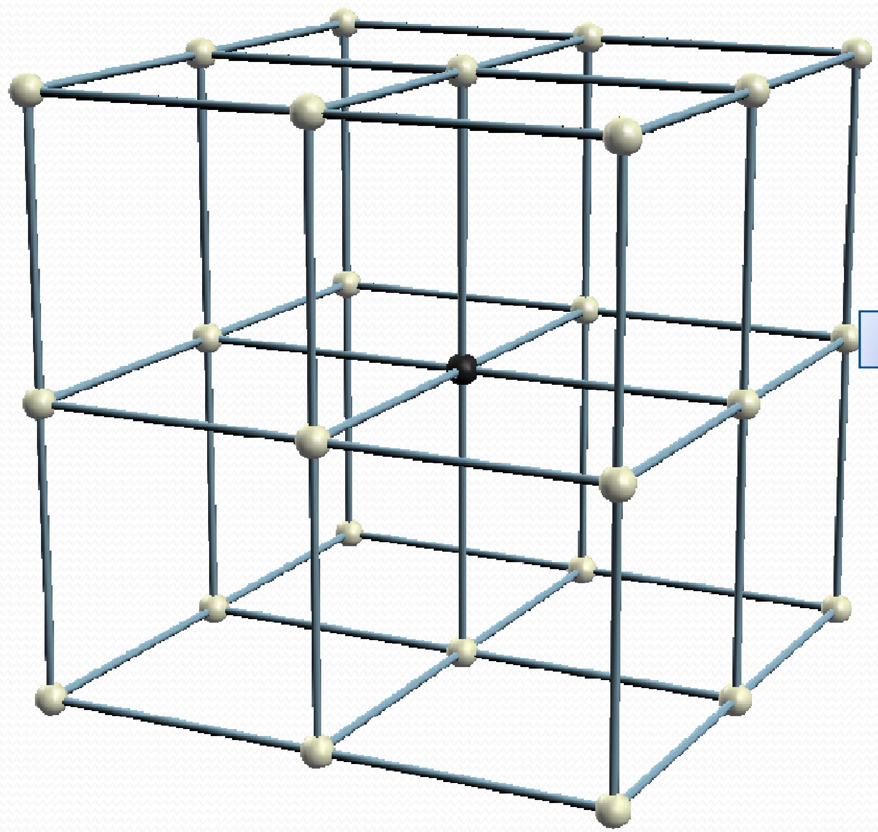
// We should replace the indices to edges (0..11)
// to indices of the mesh vertices

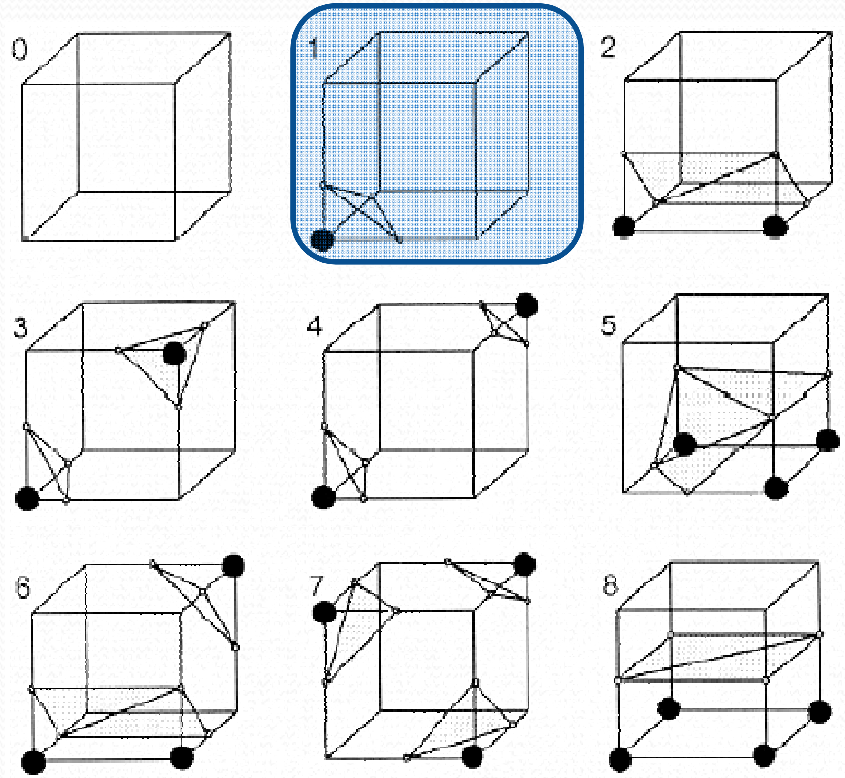
fper



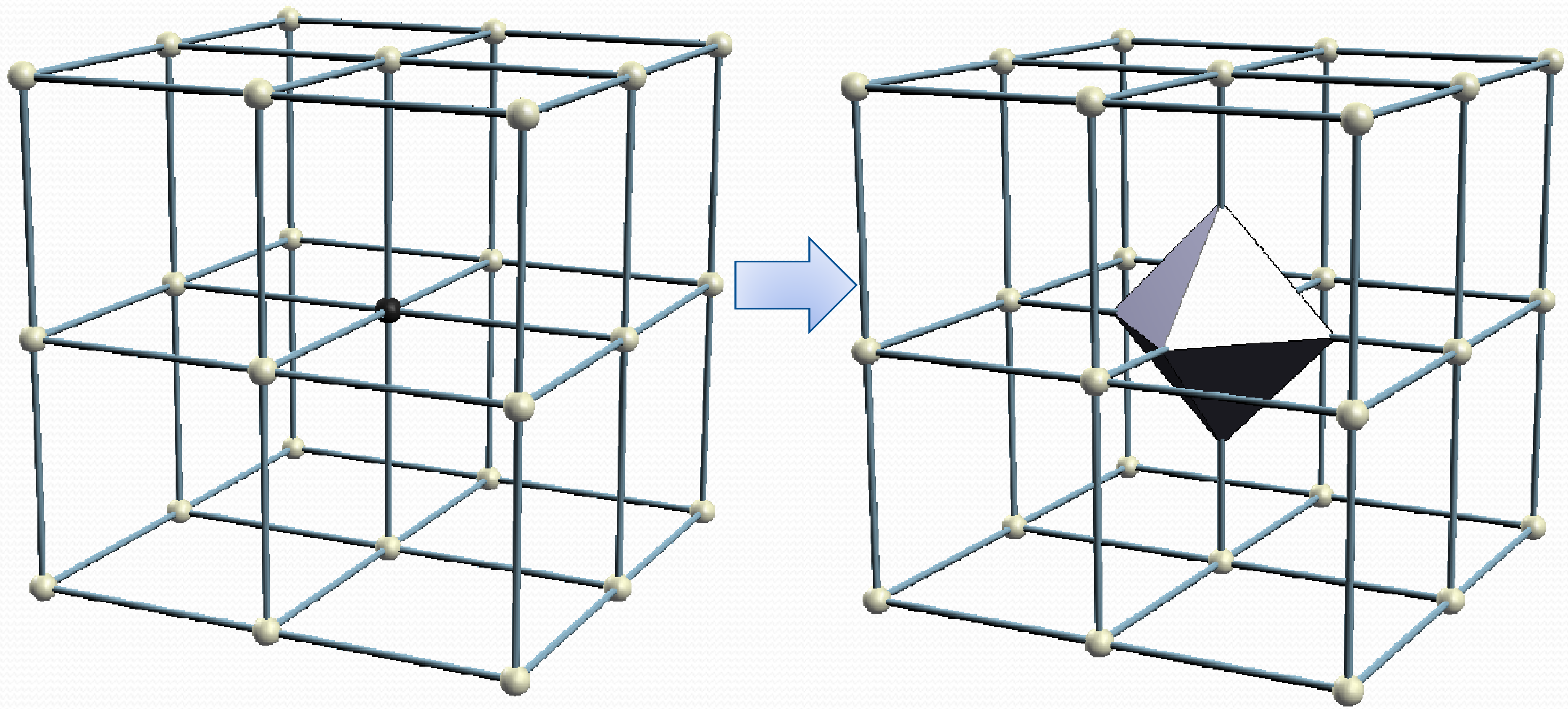

| | |
|---|---|
| 0 | Triangles config 0 |
| 1 | Triangles config 1 |
| ... | |
| **51** | **{ {8, 9, 10}, {9, 11, 10} }** |
| ... | |

# Reconstruction examples

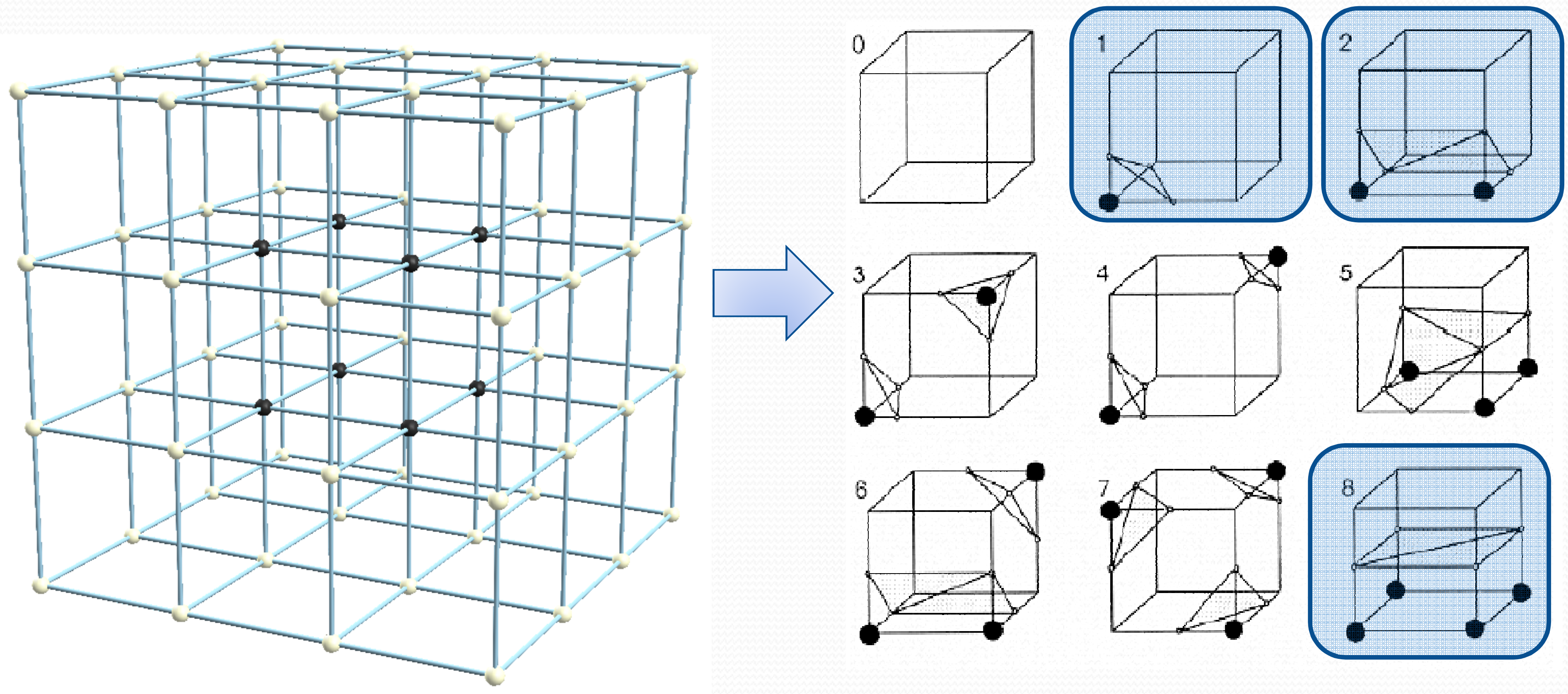




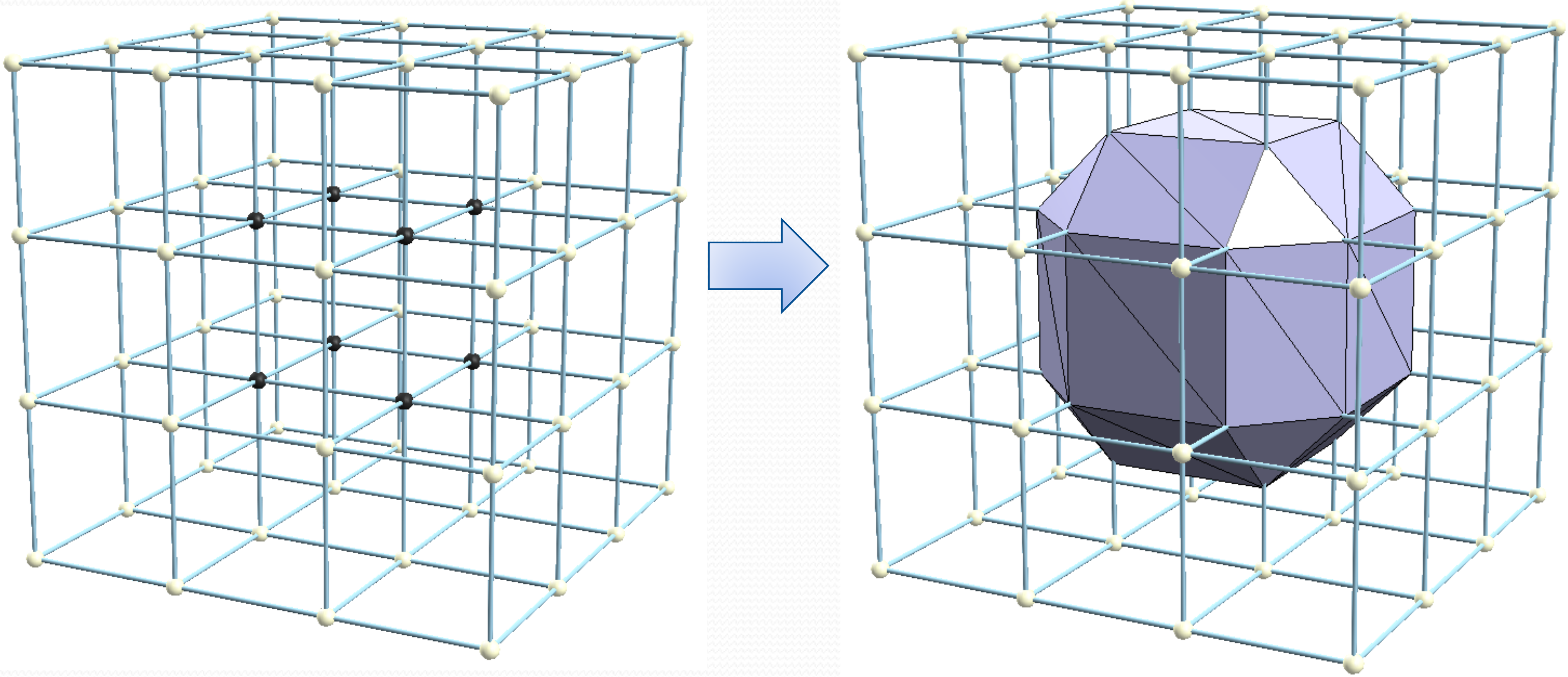
# Reconstruction examples



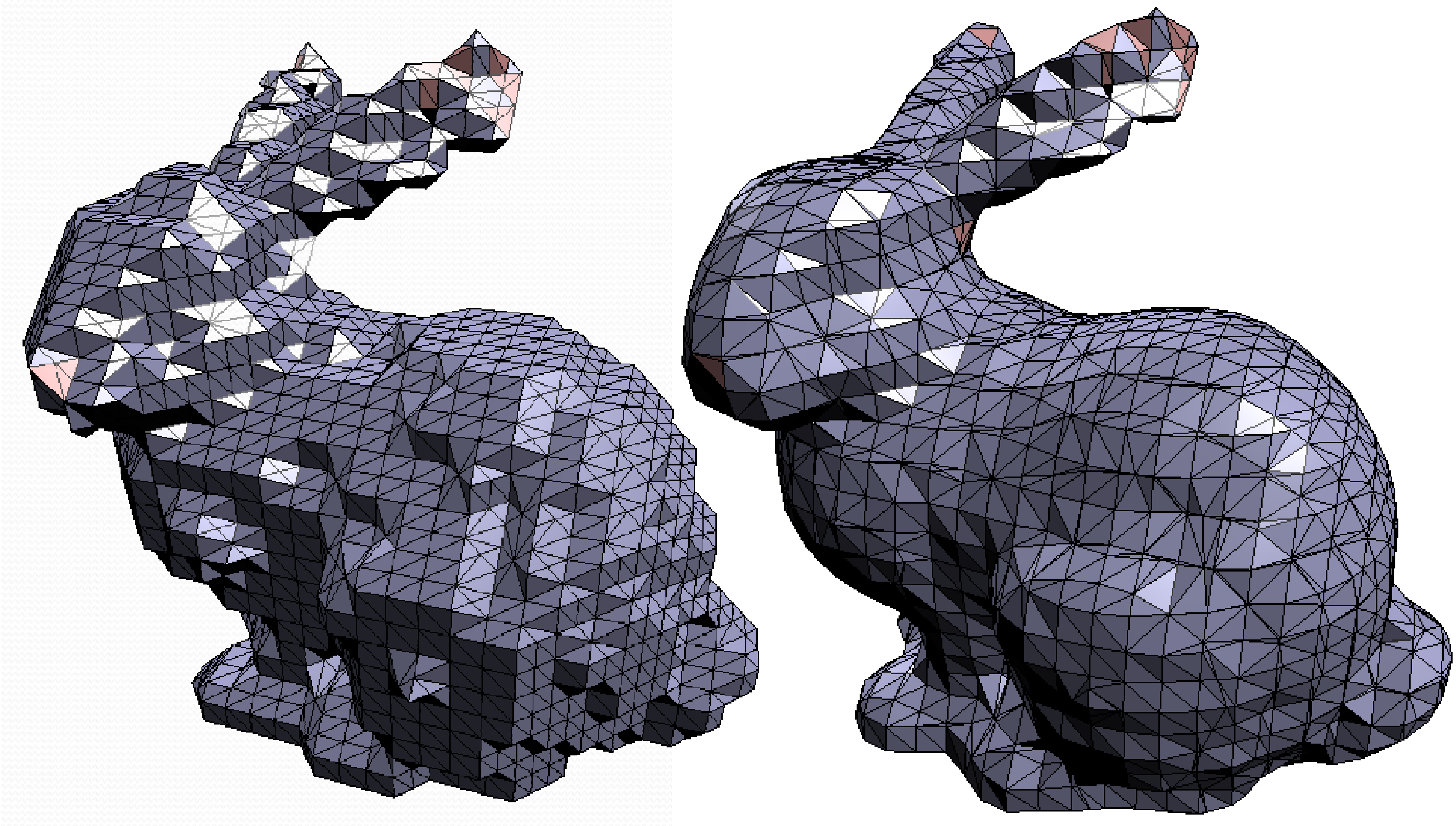
# Reconstruction examples



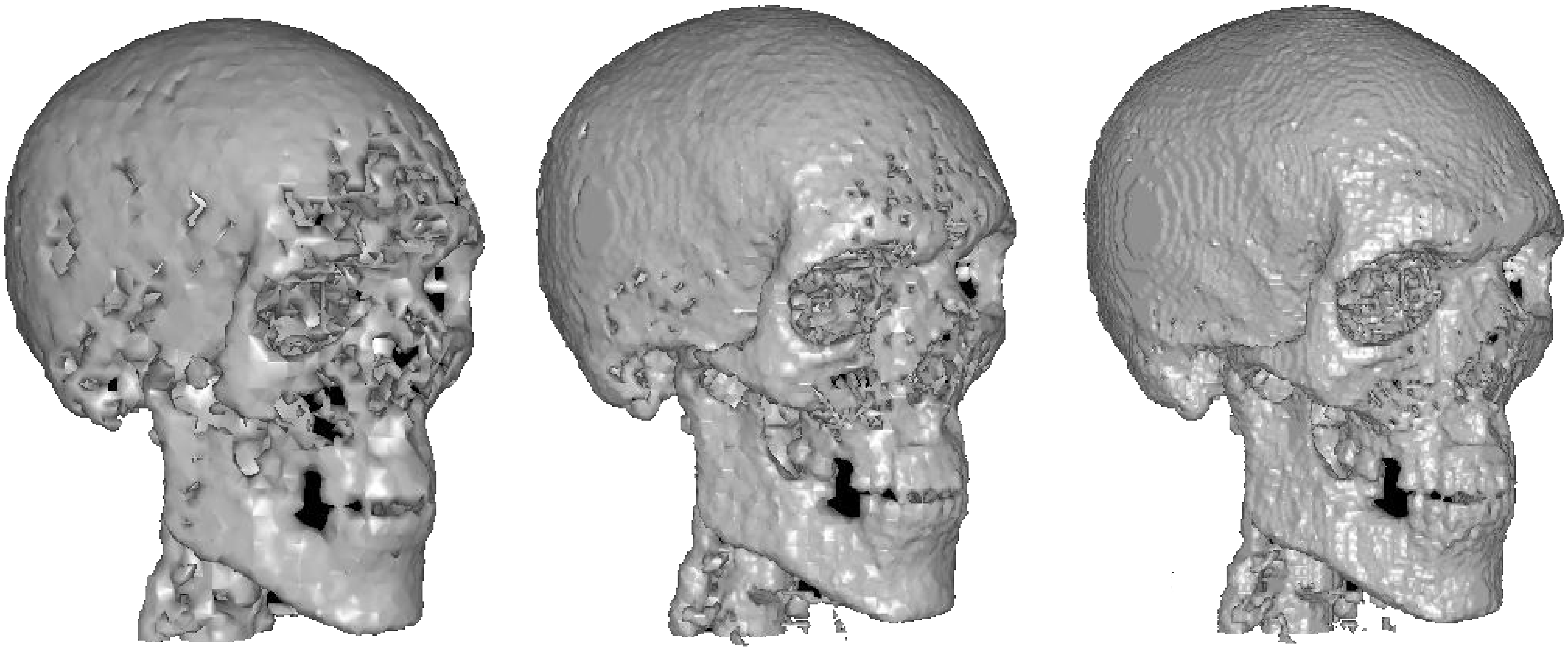
# Reconstruction examples



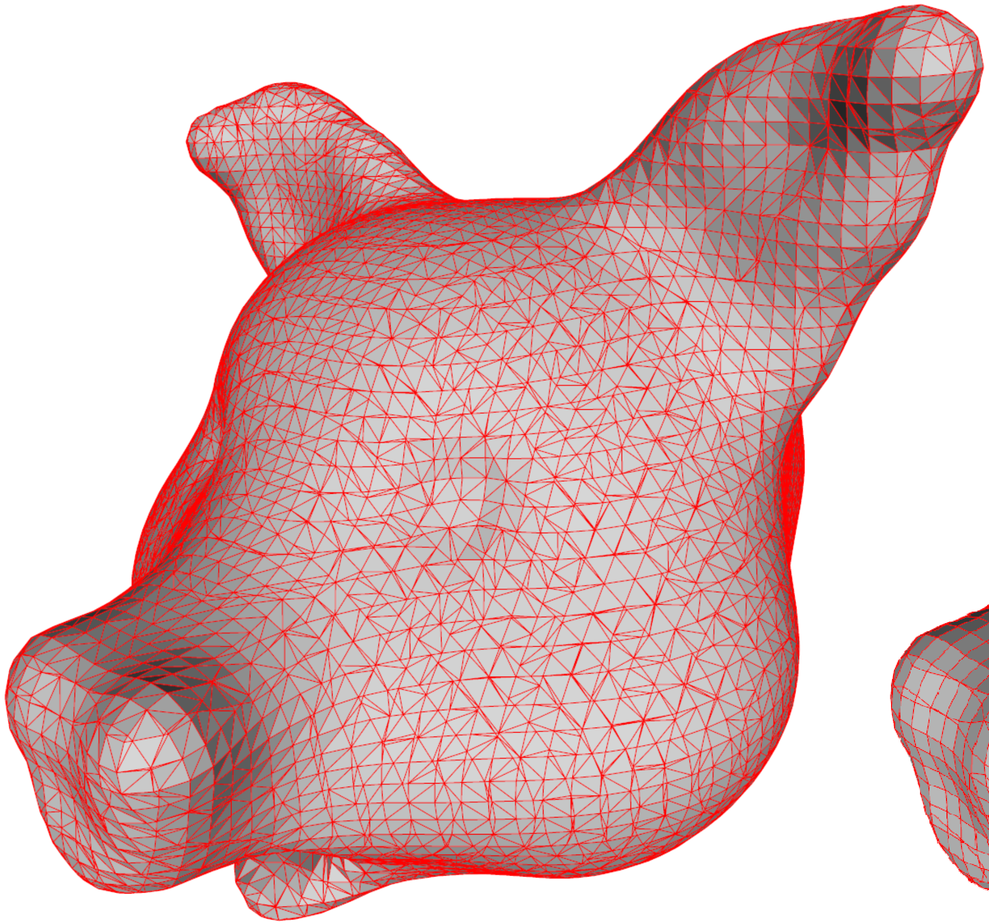
# Reconstruction examples

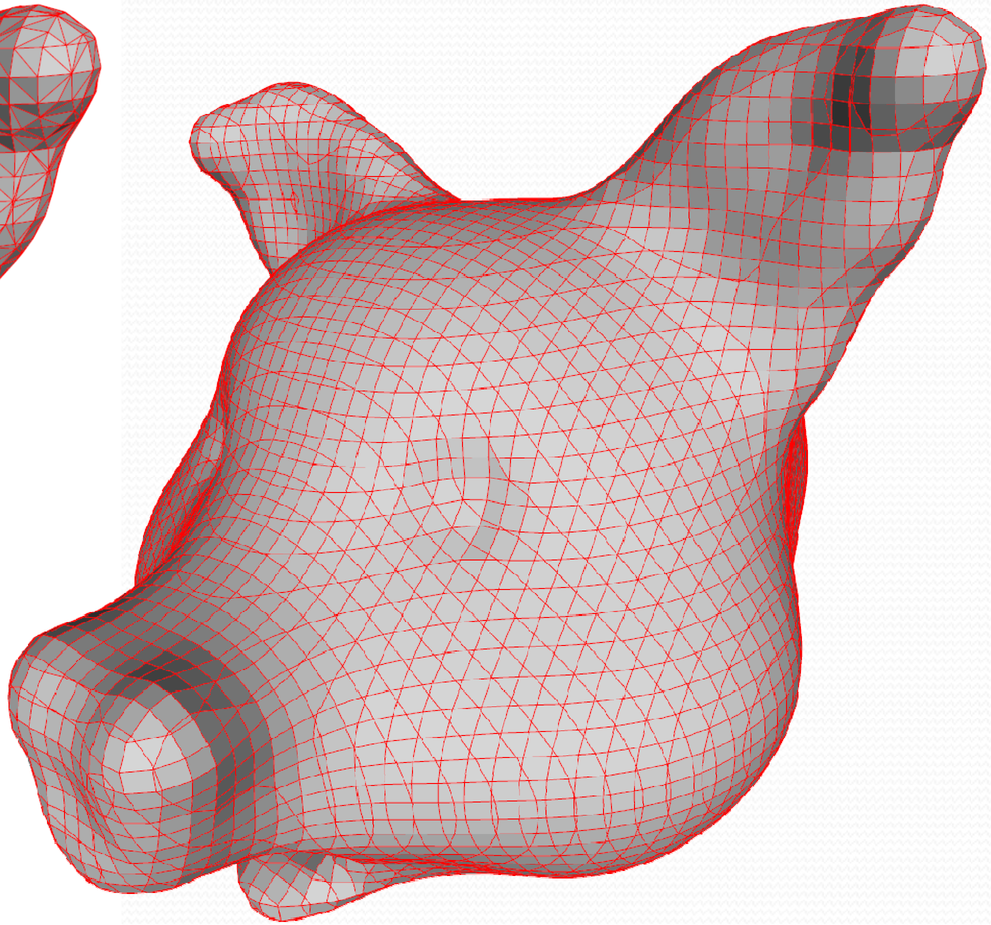


# Reconstruction examples

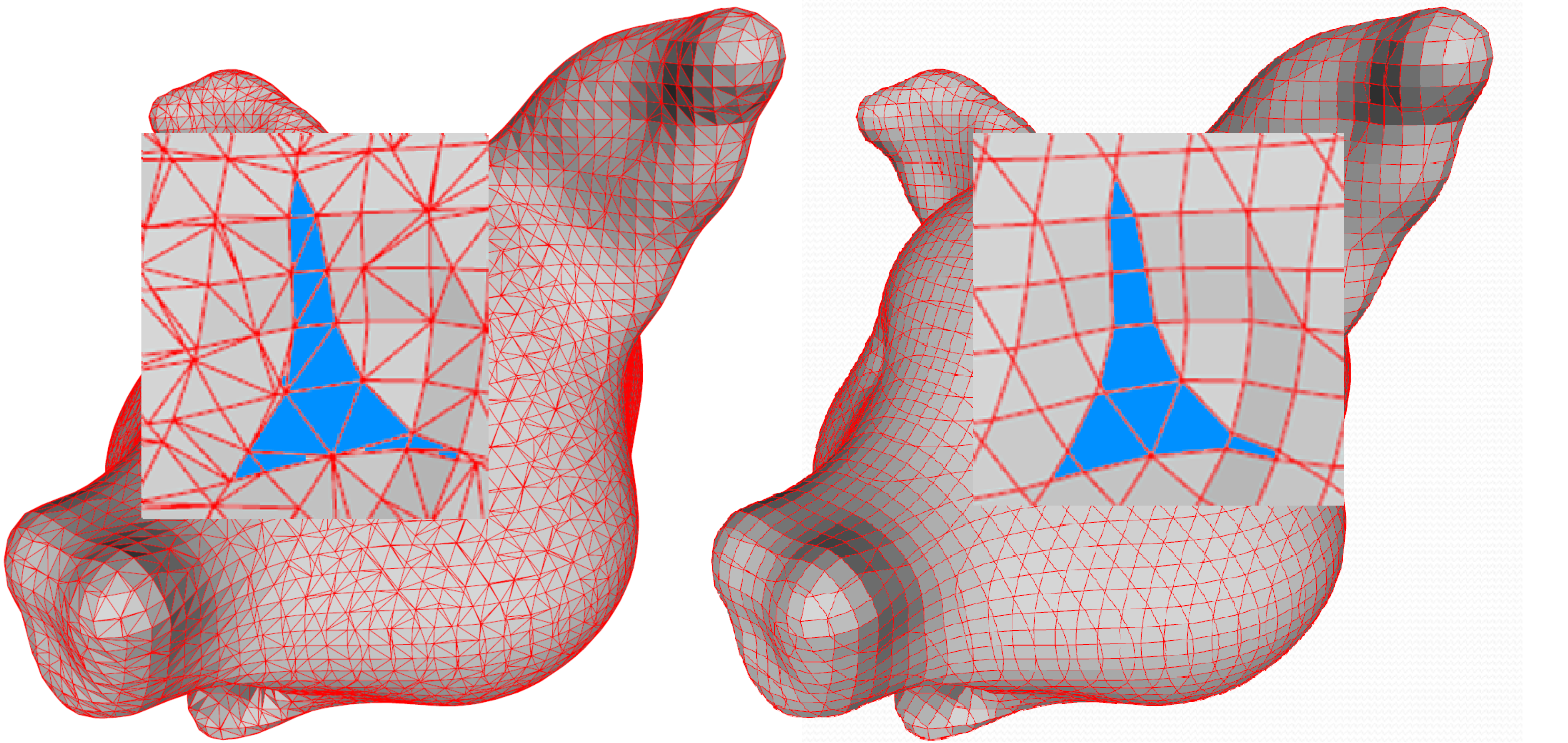


# Reconstruction examples



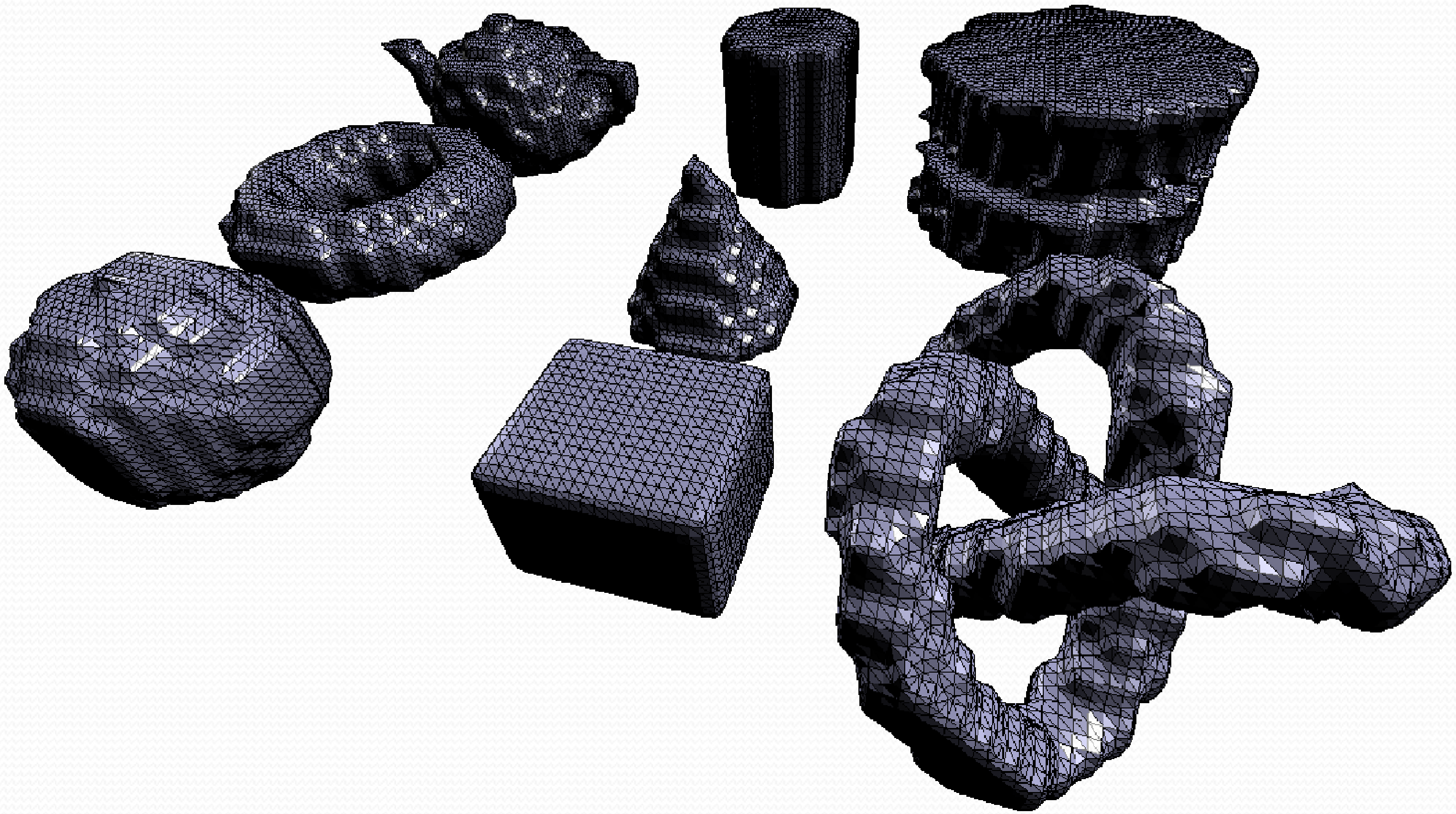


# Reconstruction examples



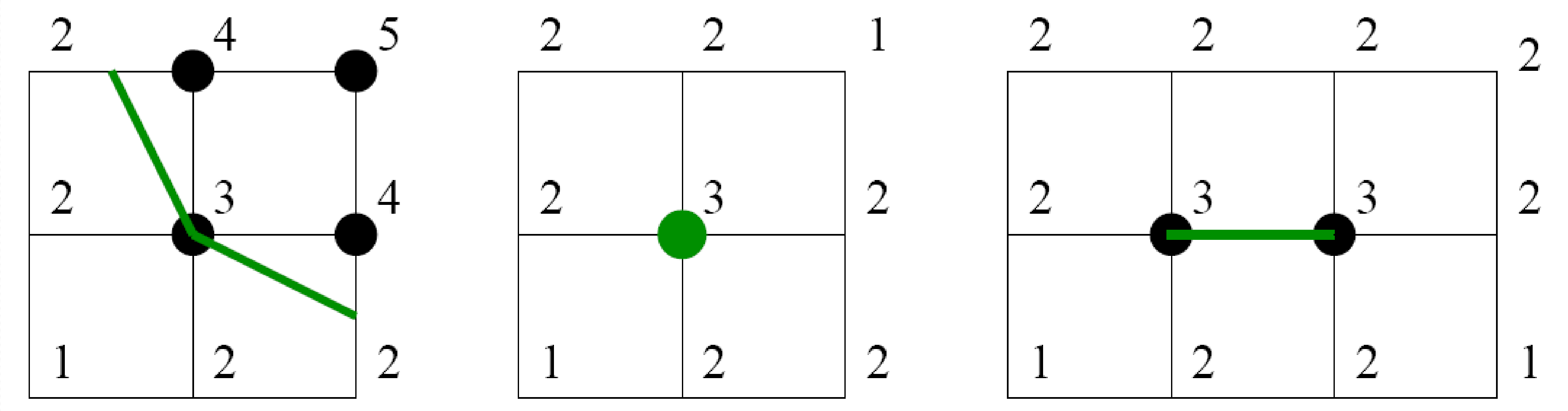
# Limitations of original MC

- Number of triangles



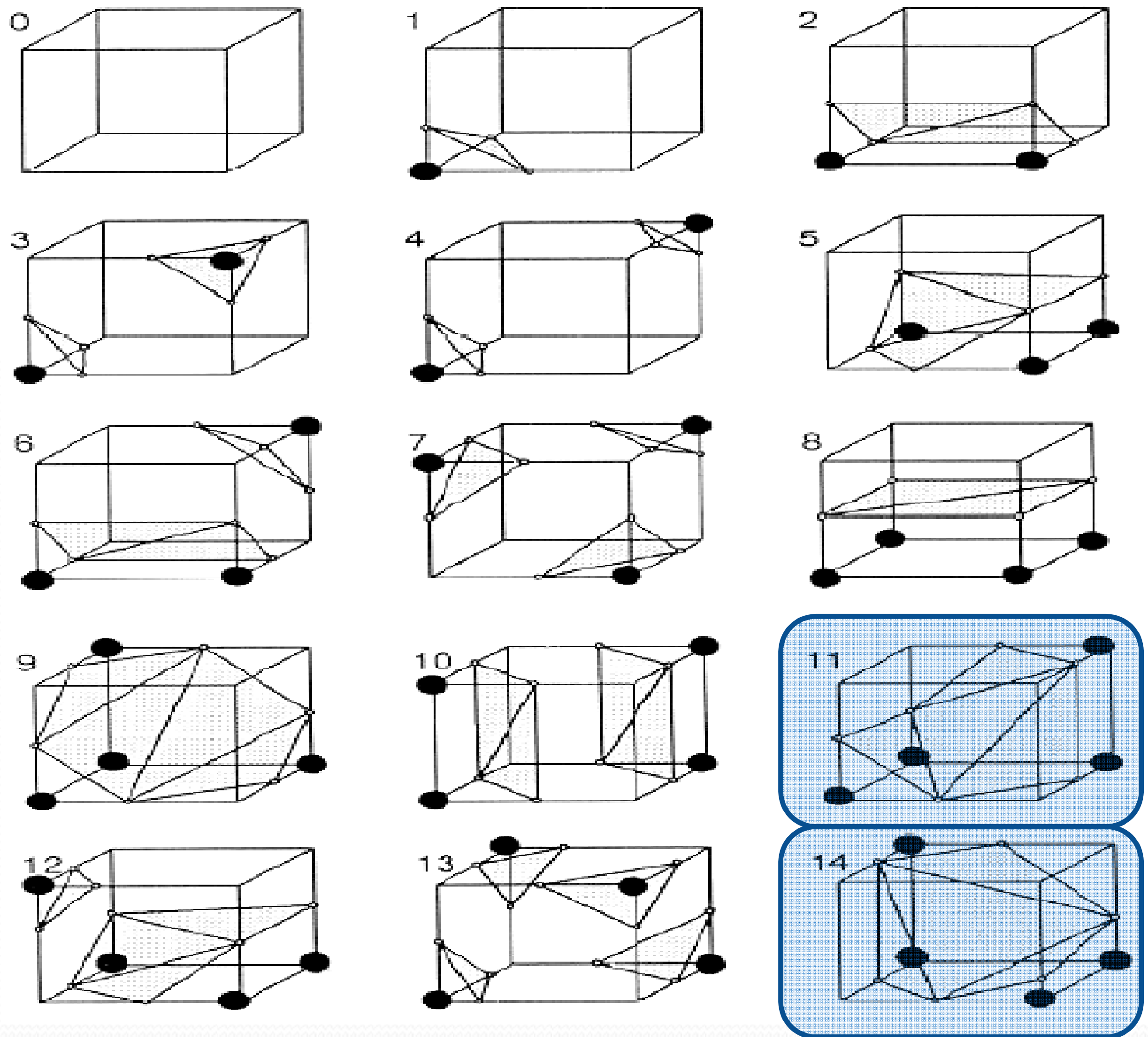
# Limitations of original MC

- Triangle quality. It might create edges with null length and triangles with null area.



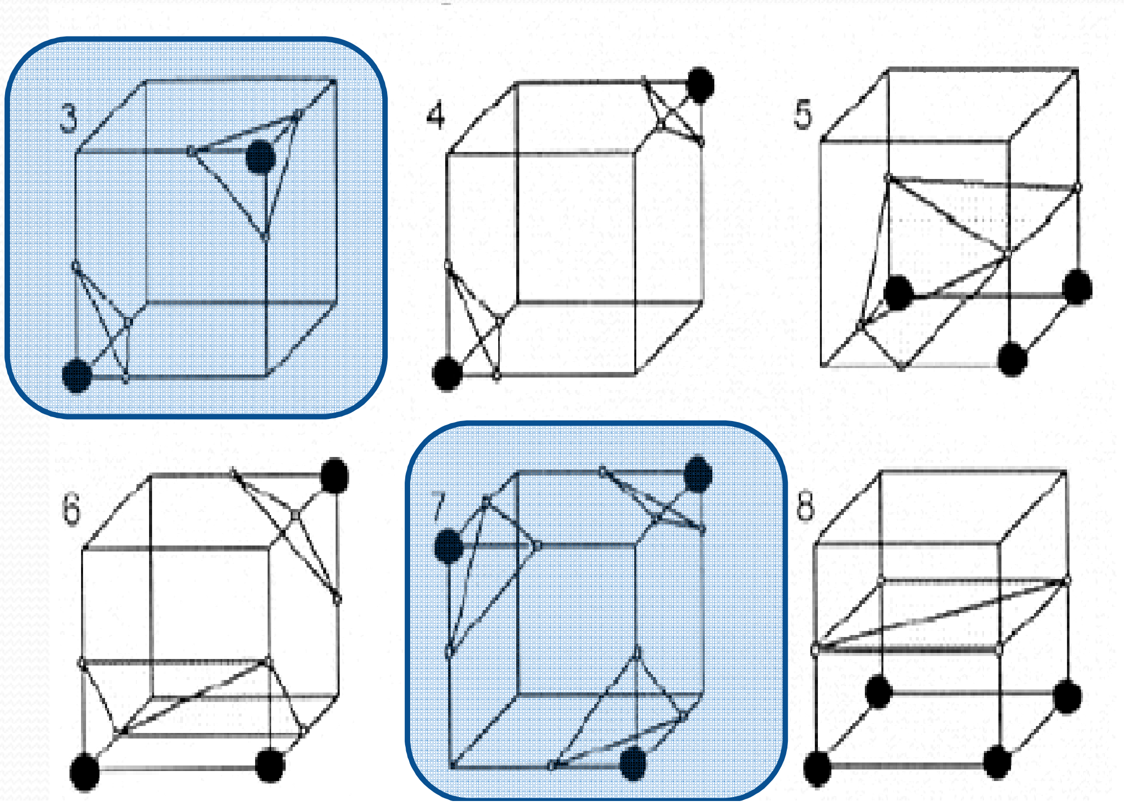
# Limitations of original MC

- One case is duplicated



# Limitations of original MC

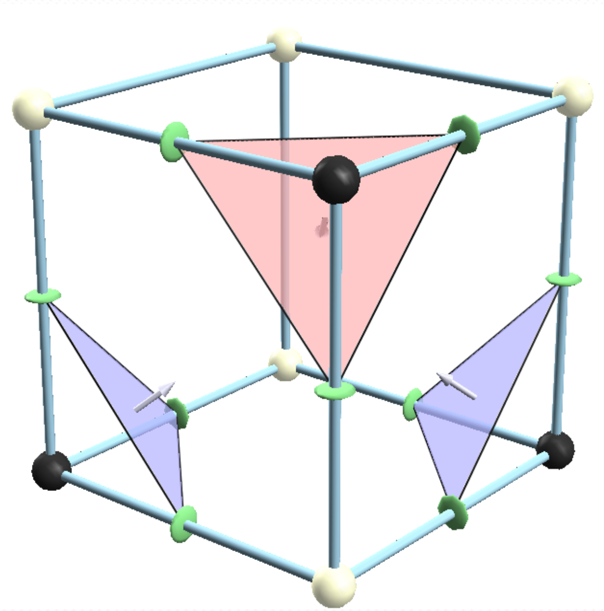
- The output surface might include holes! Some ambiguous faces are not reconstructed consistently.



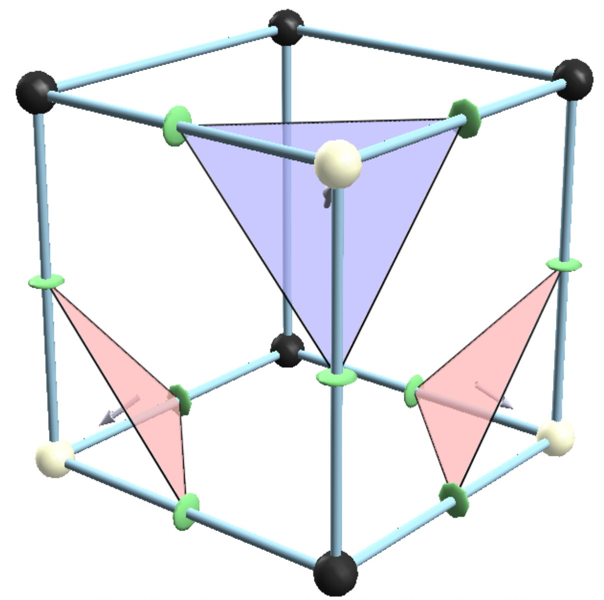
# Limitations of original MC

Ambiguous faces:

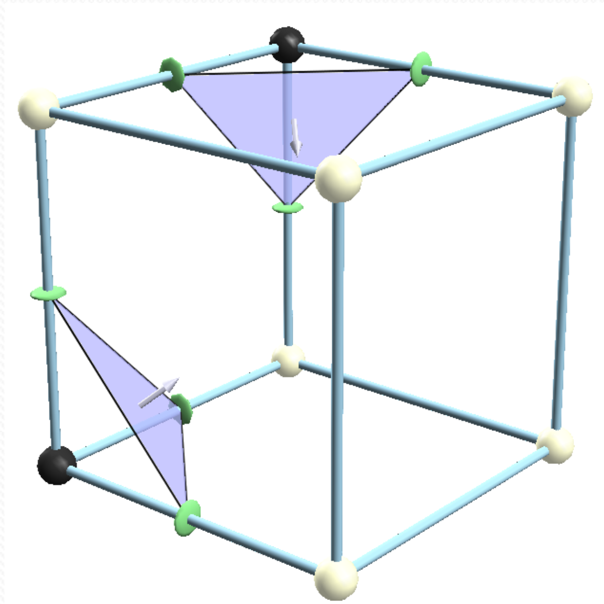
- Some cases separate black vertices (ex. case 3)
- Some other cases join black vertices (ex. case  ~7)



Case 7

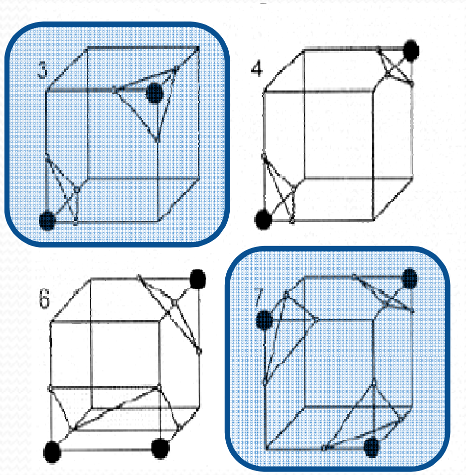


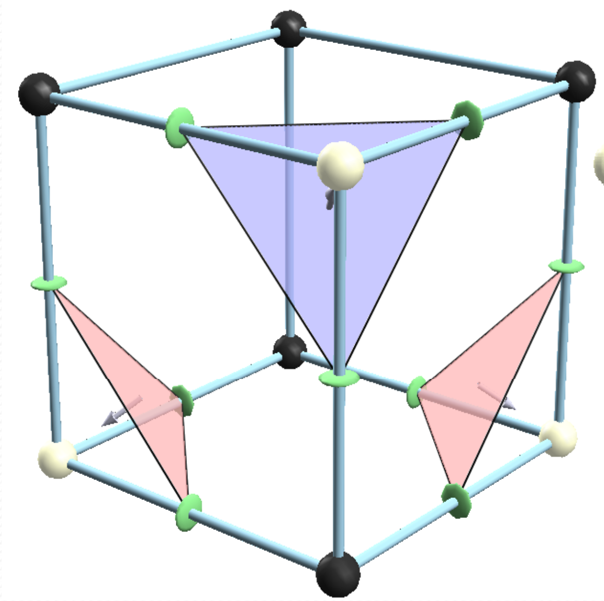
Case 7 complementat



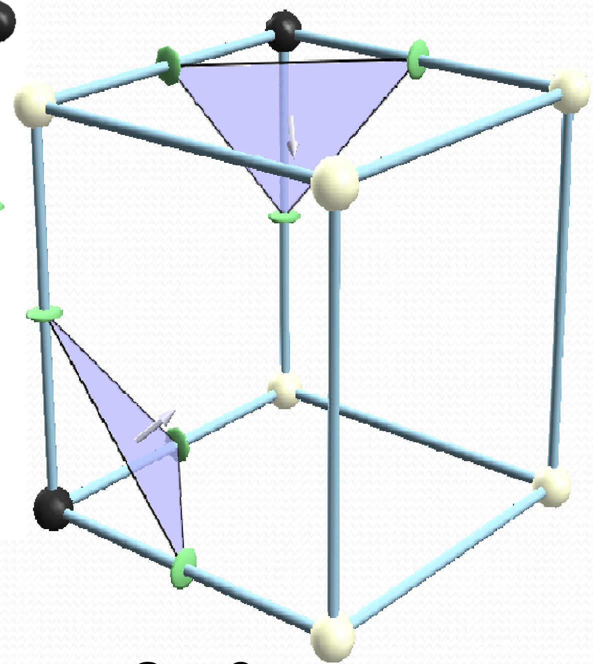
Case 3

# Limitations of original MC



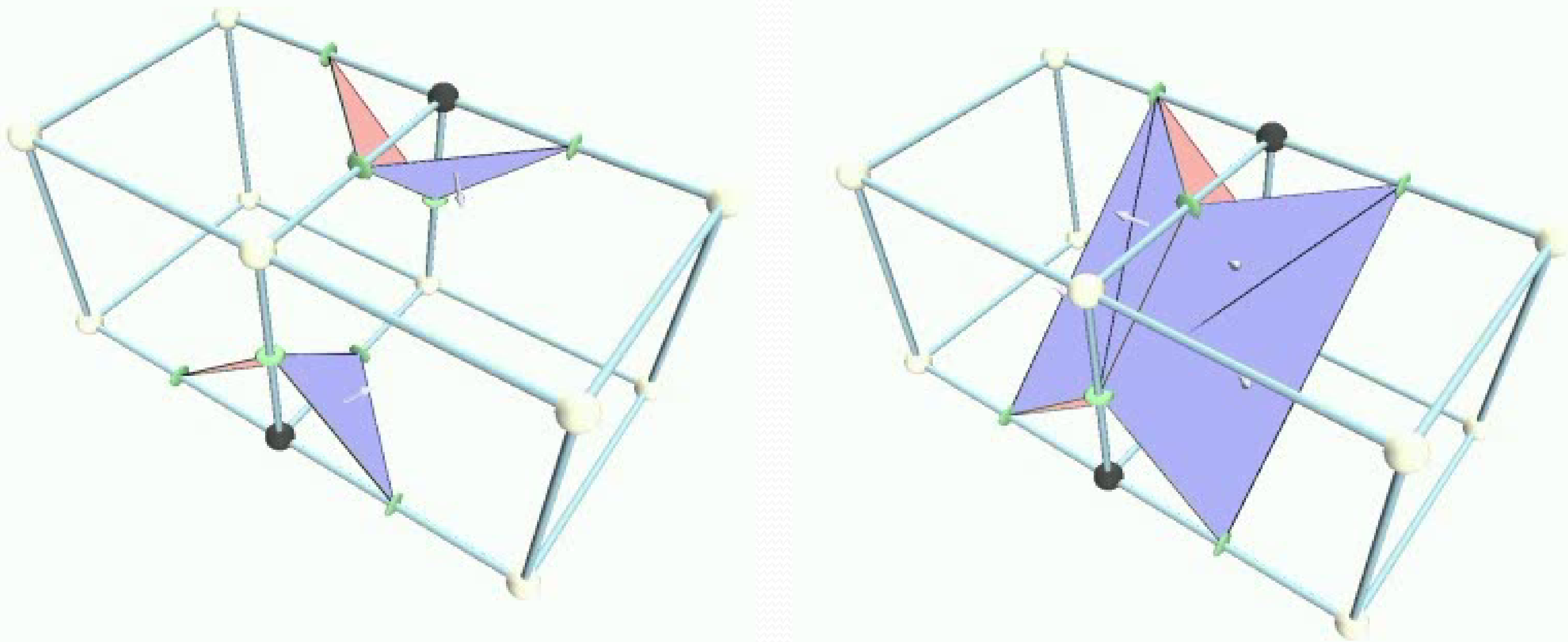



Cas 7 complementat

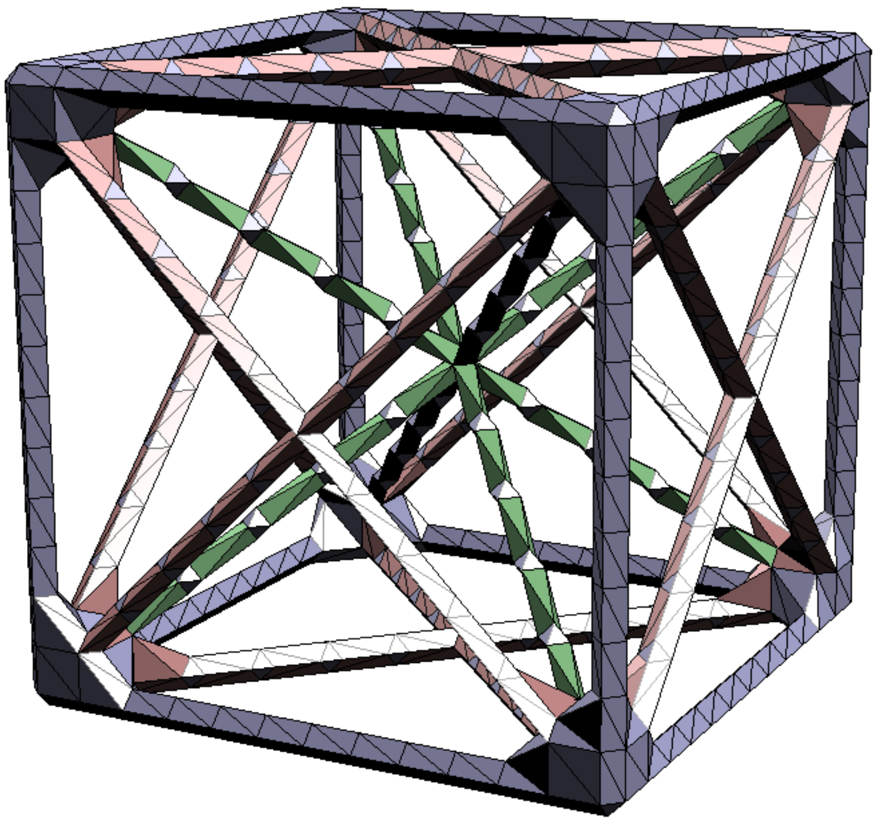


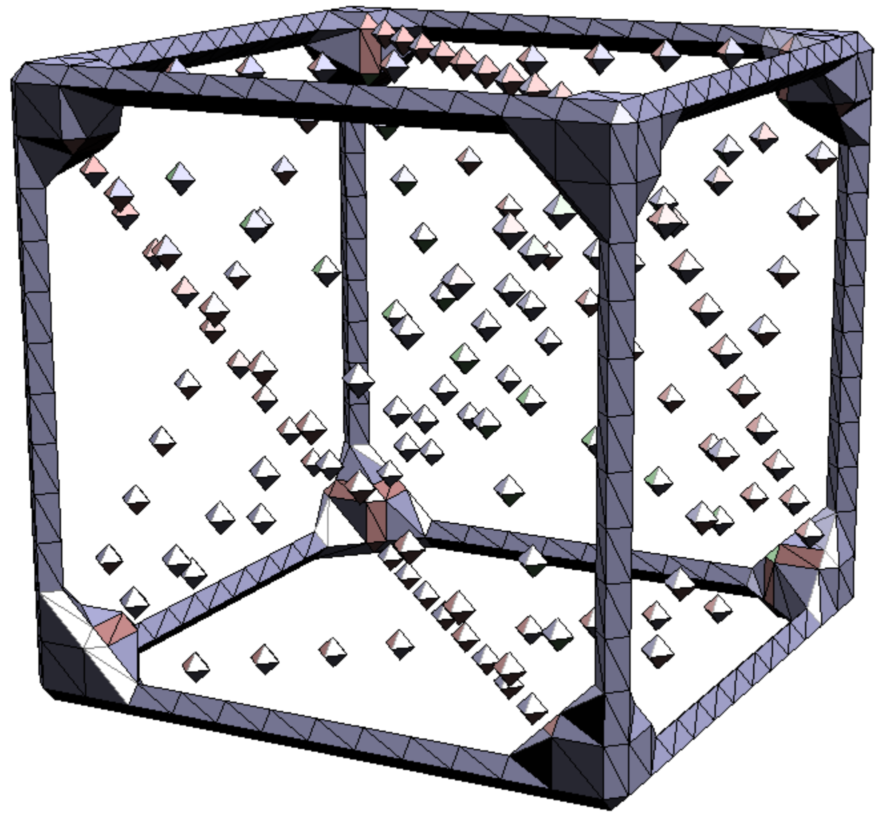
Cas 3

# Consistent reconstructions



# Examples

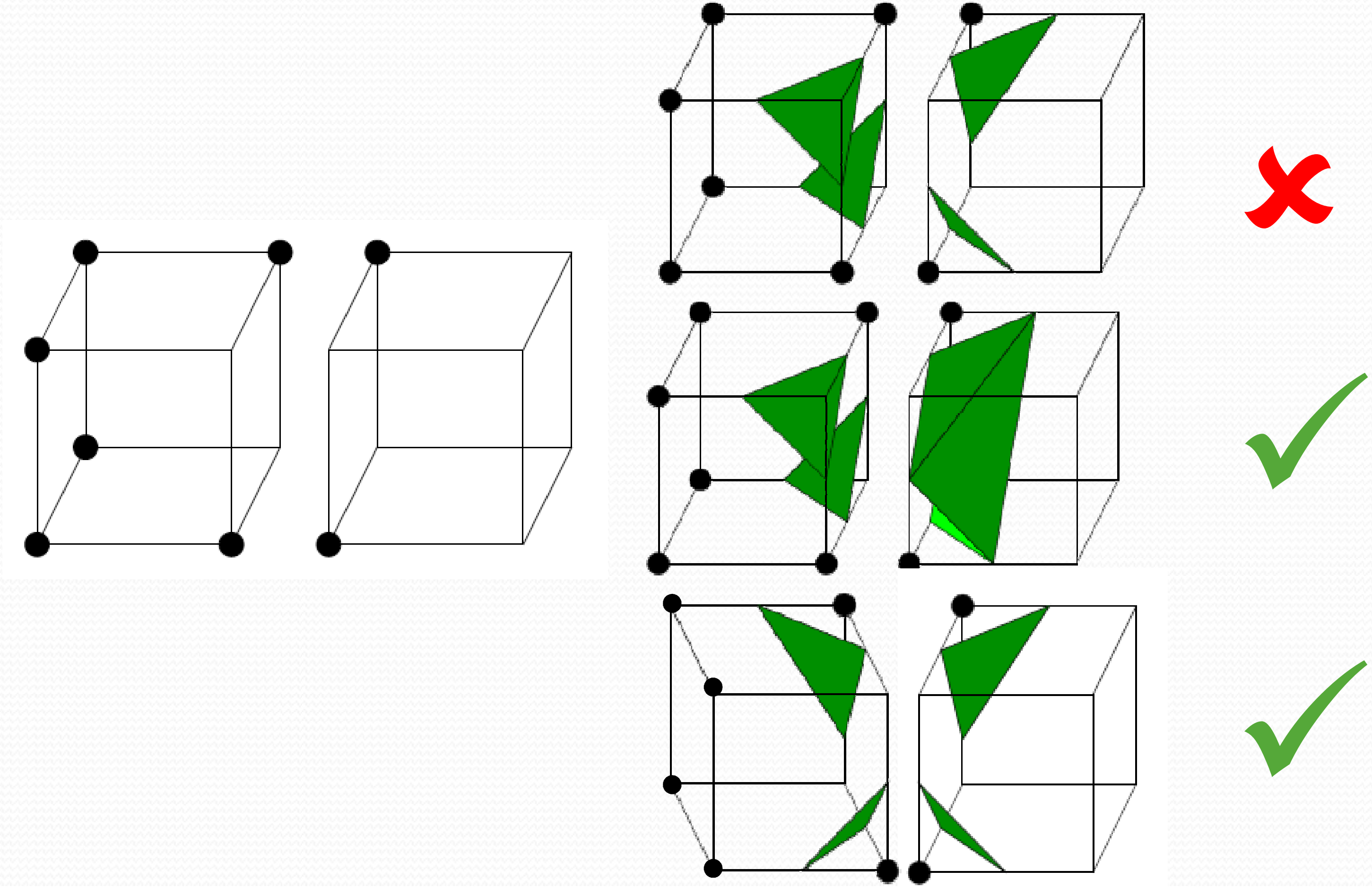




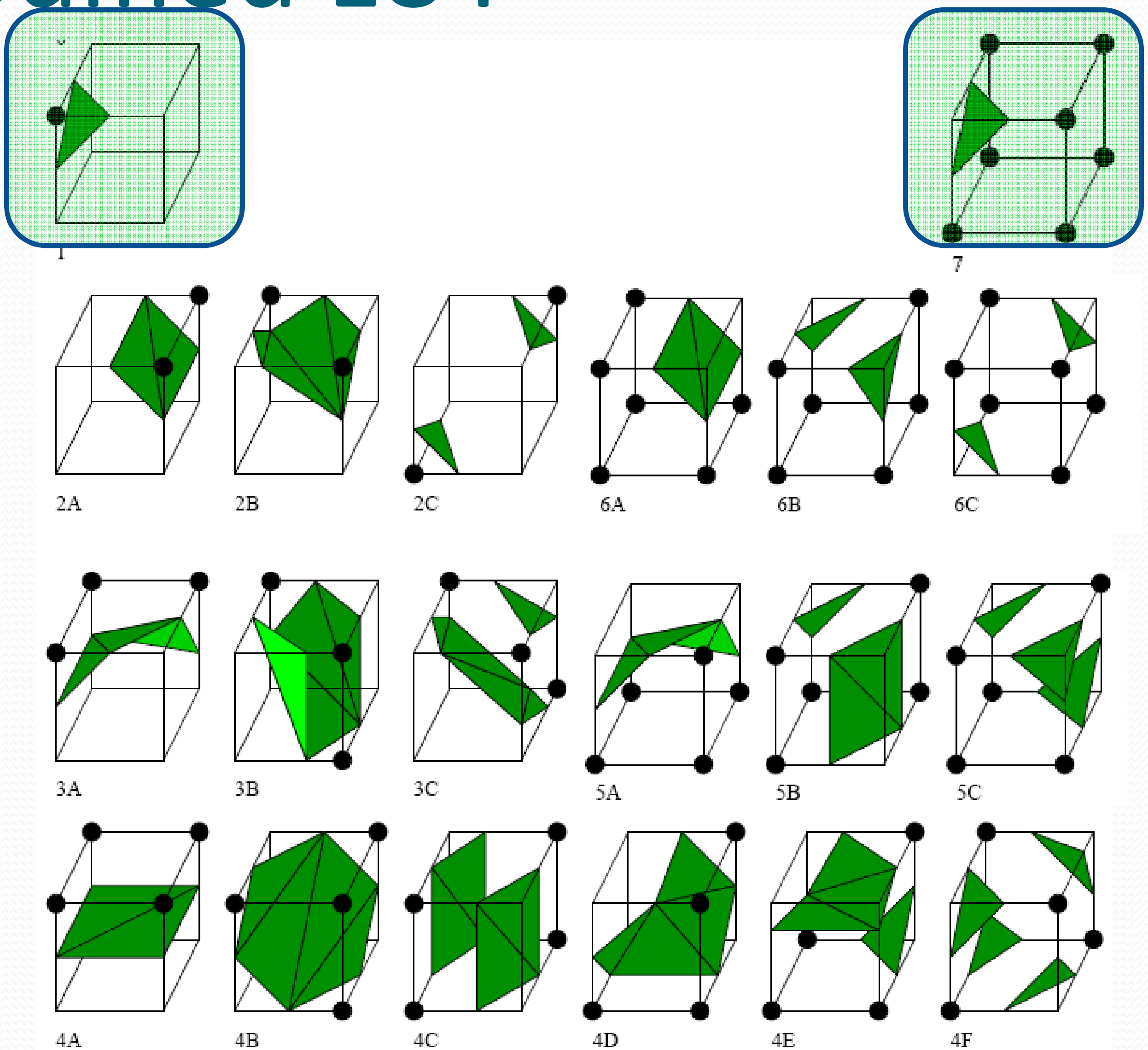
[Demo Cub; Demo bunny]

# Modified LUT

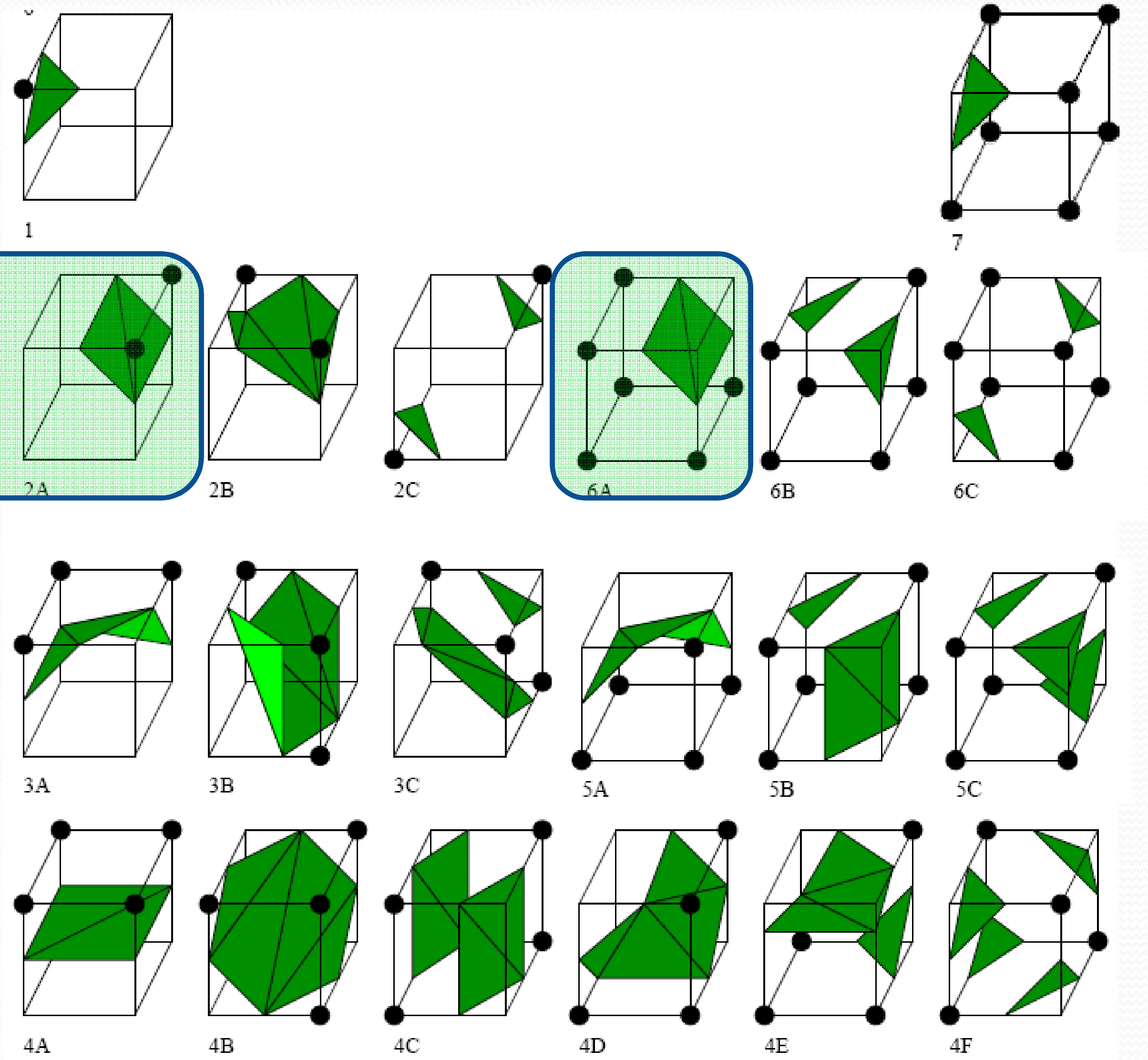
- For any pair of configurations, the reconstruction of the shared face must be consistent.



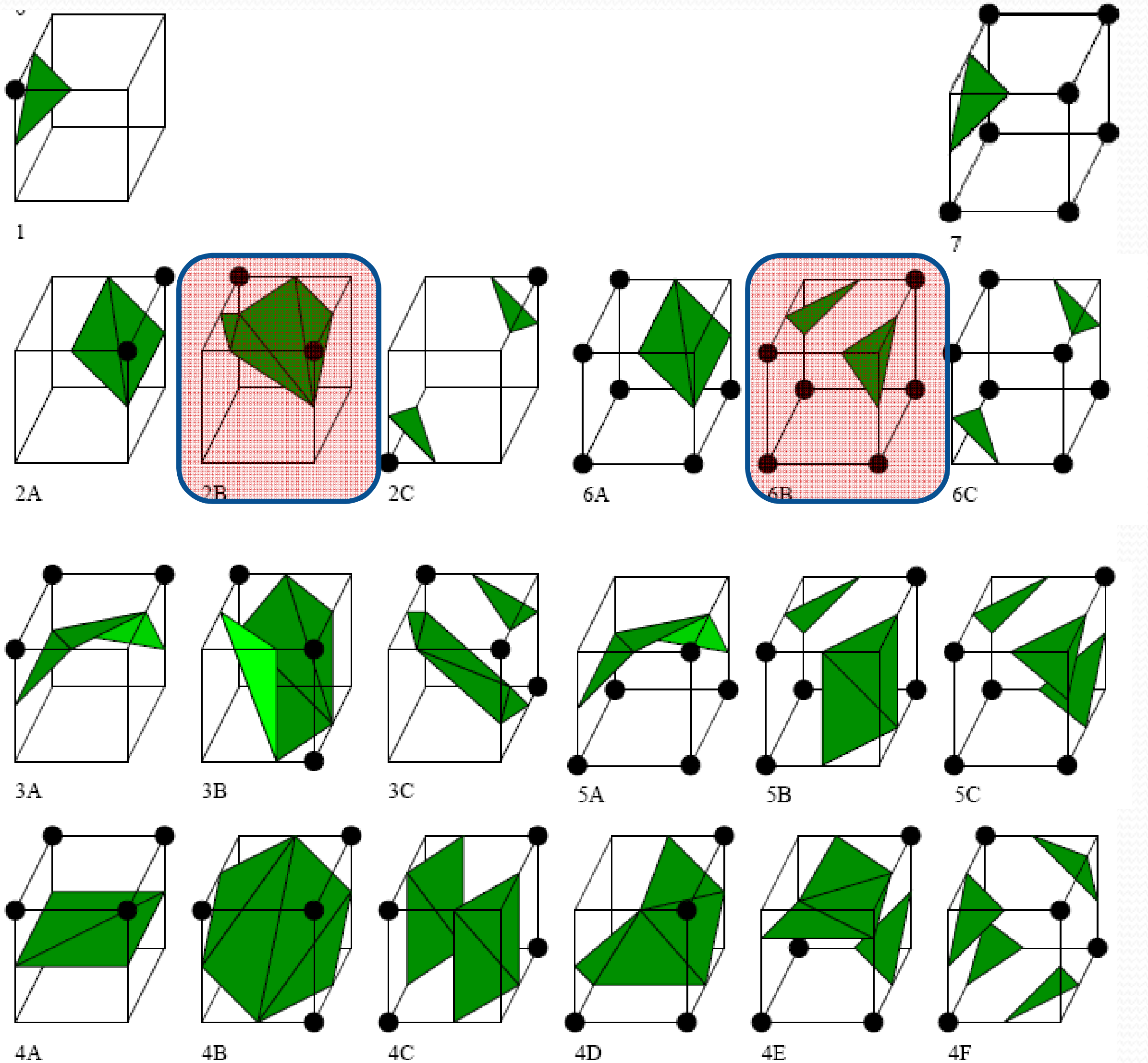
# Modified LUT



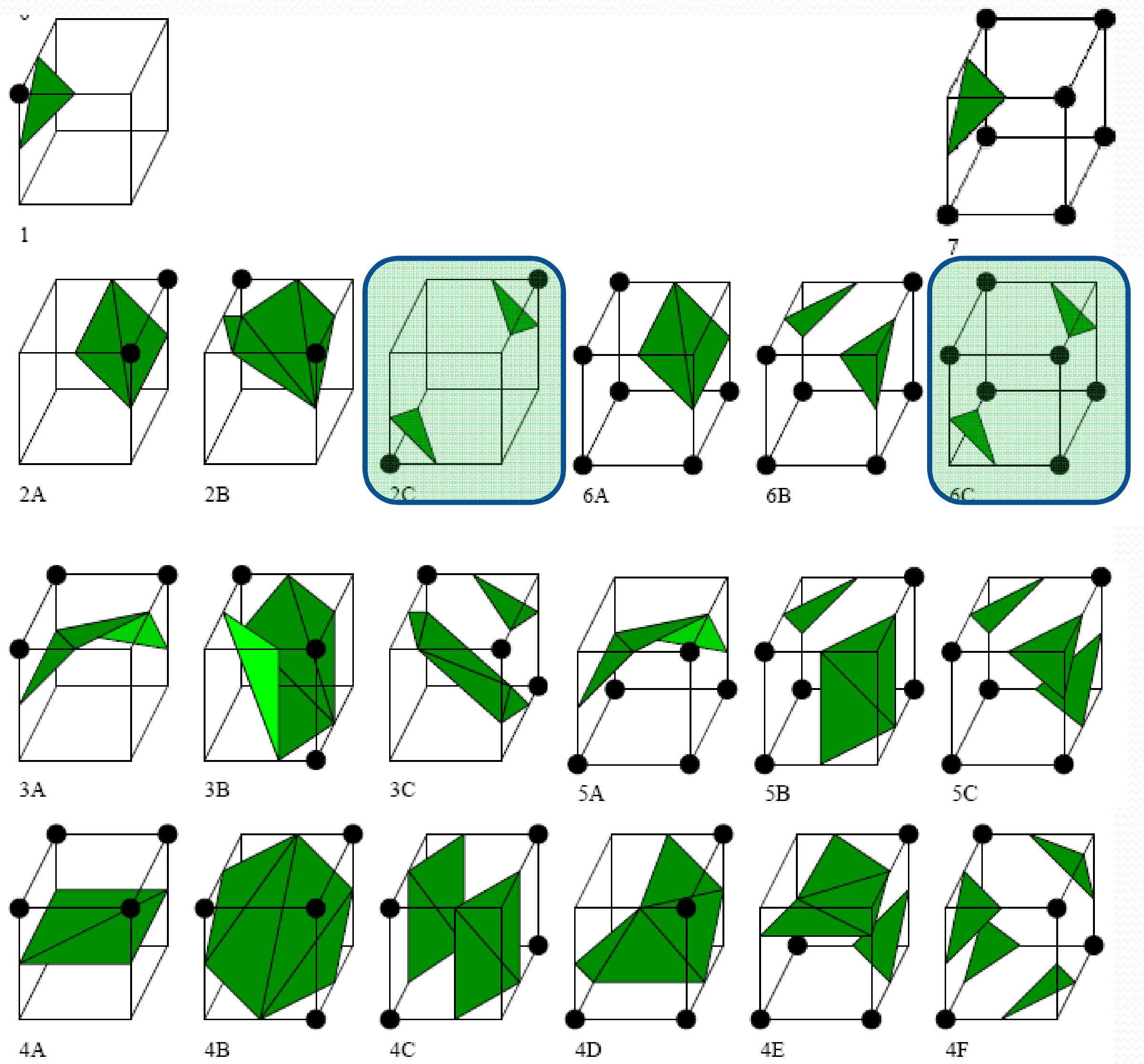
# Modified LUT



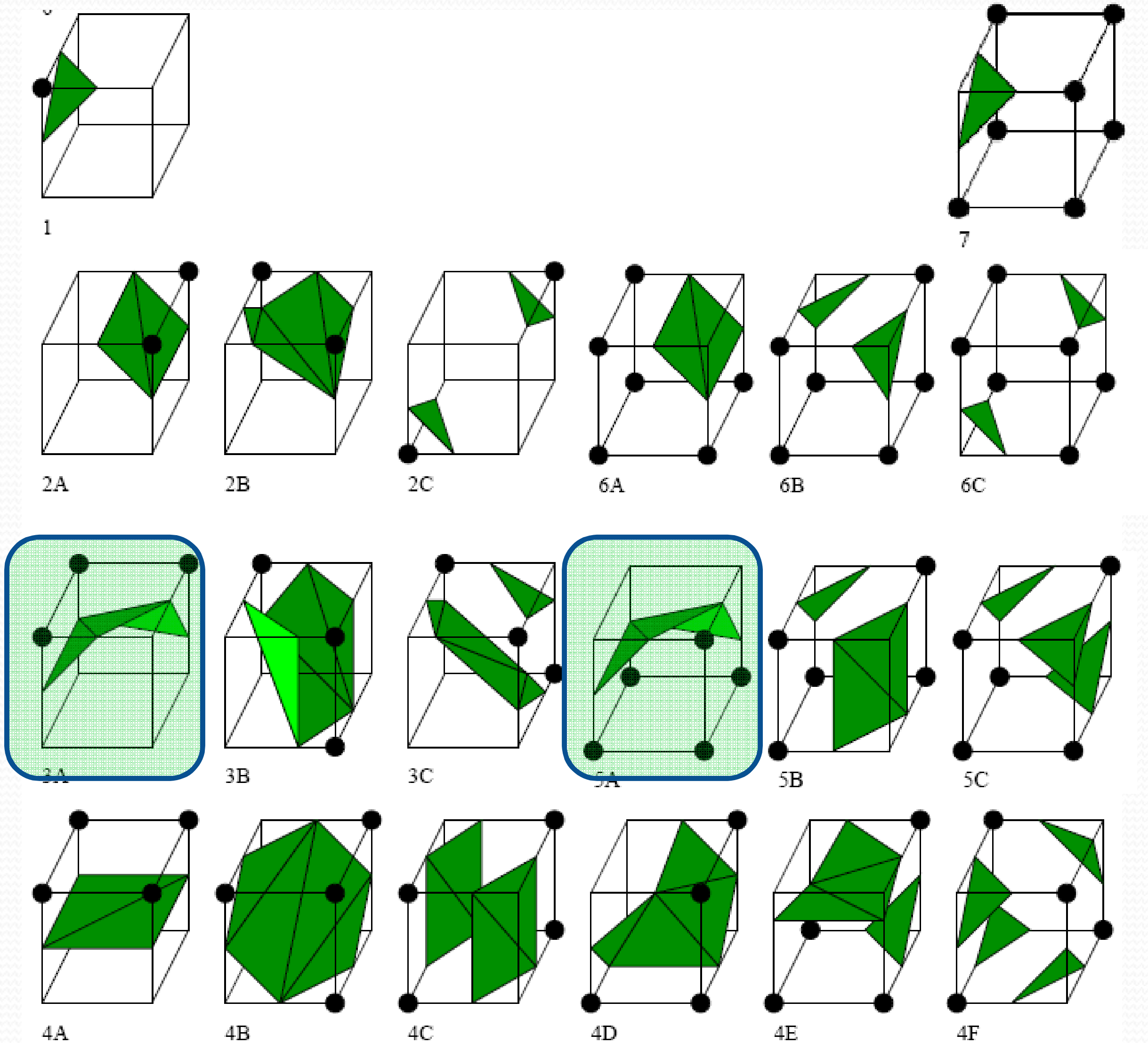
# Modified LUT



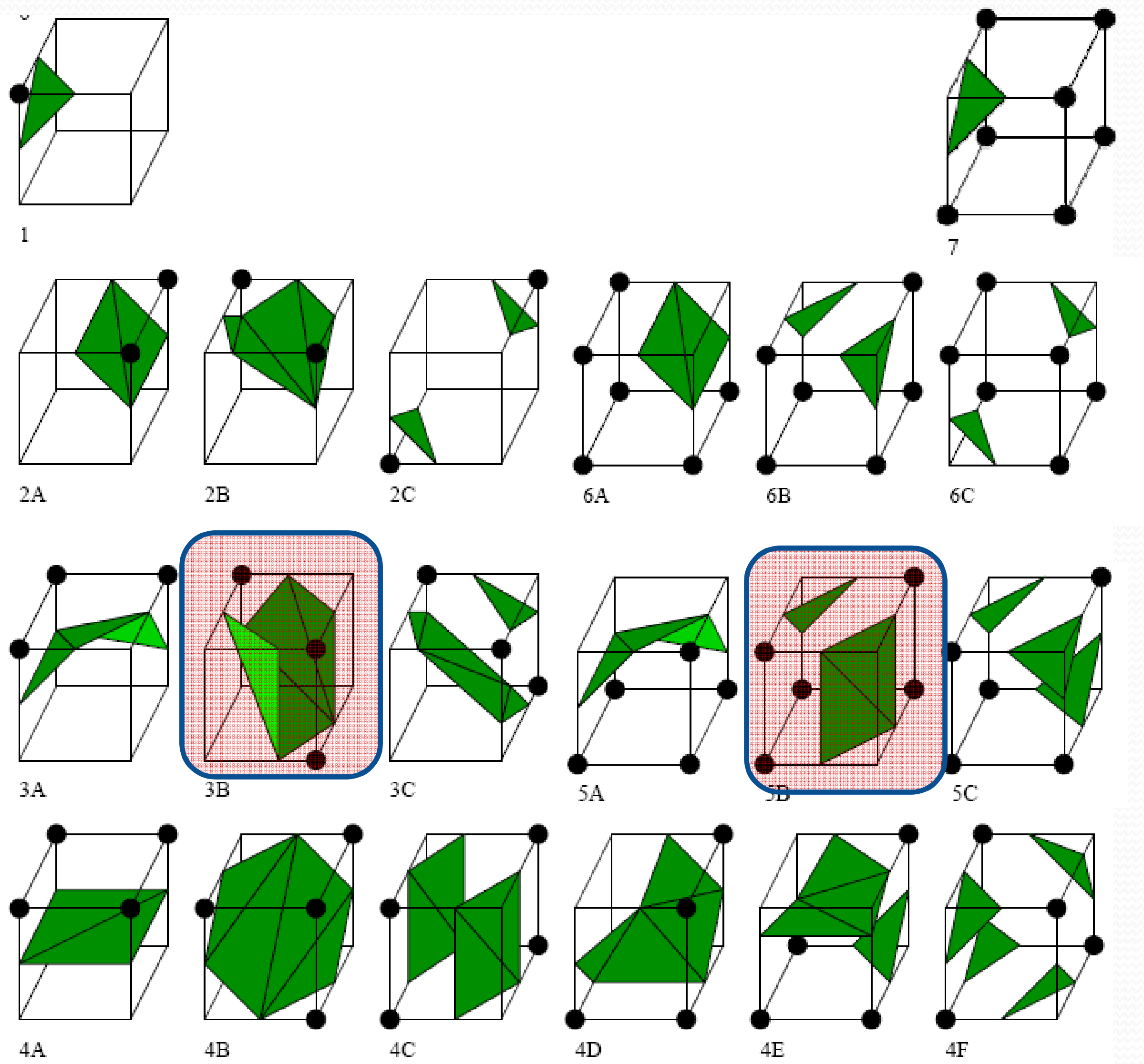
# Modified LUT



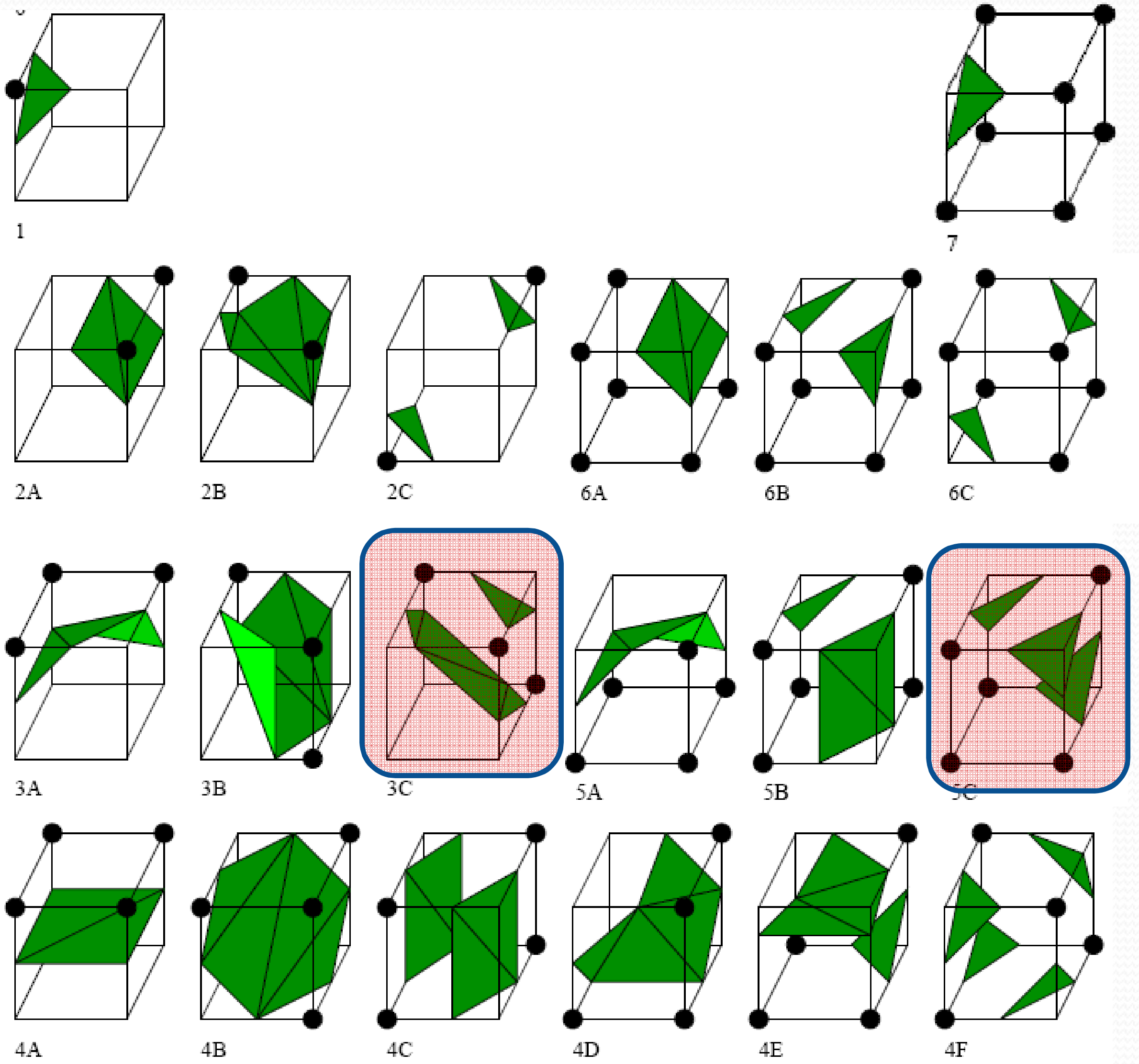
# Modified LUT



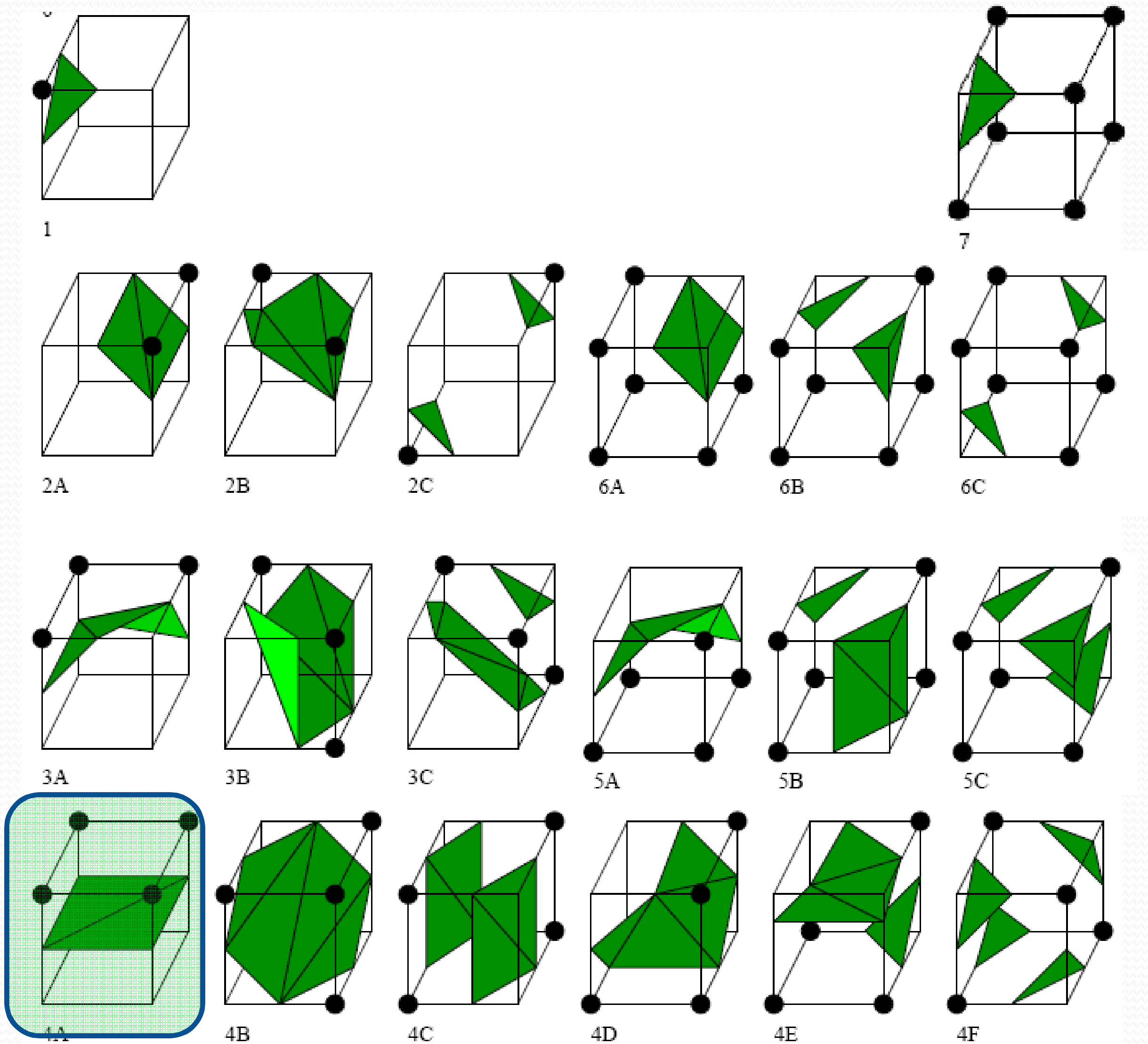
# Modified LUT



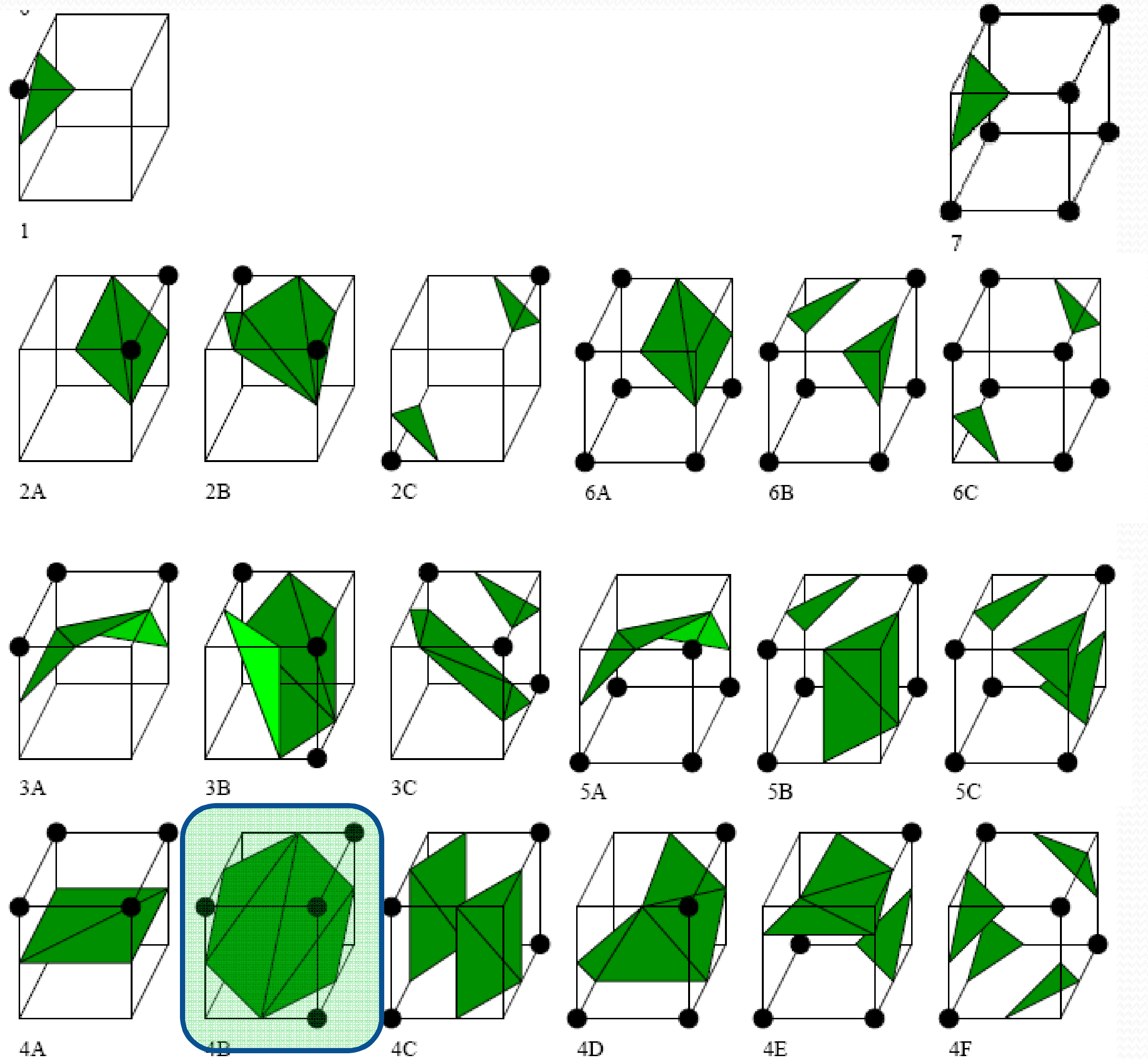
# Modified LUT



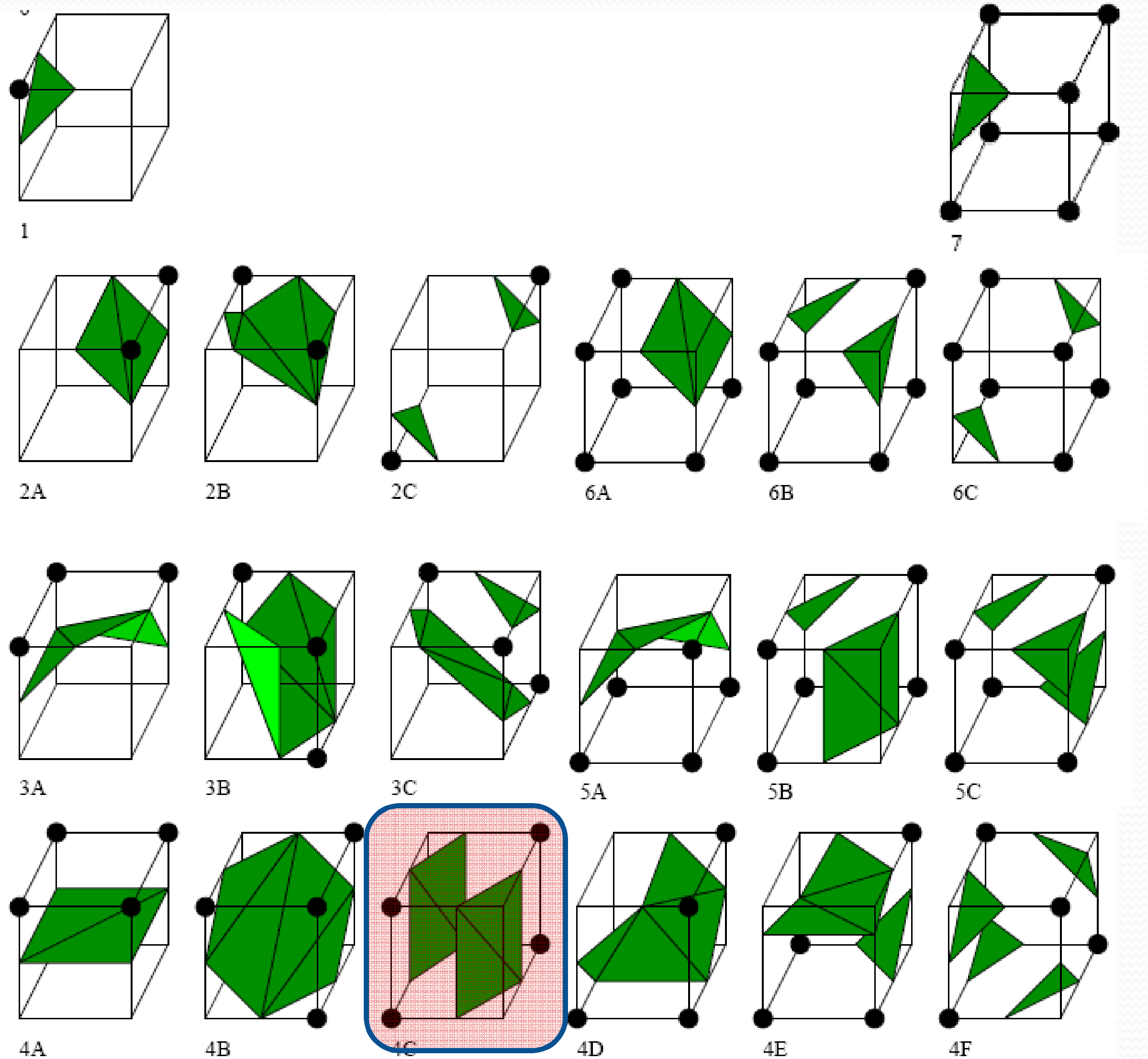
# Modified LUT



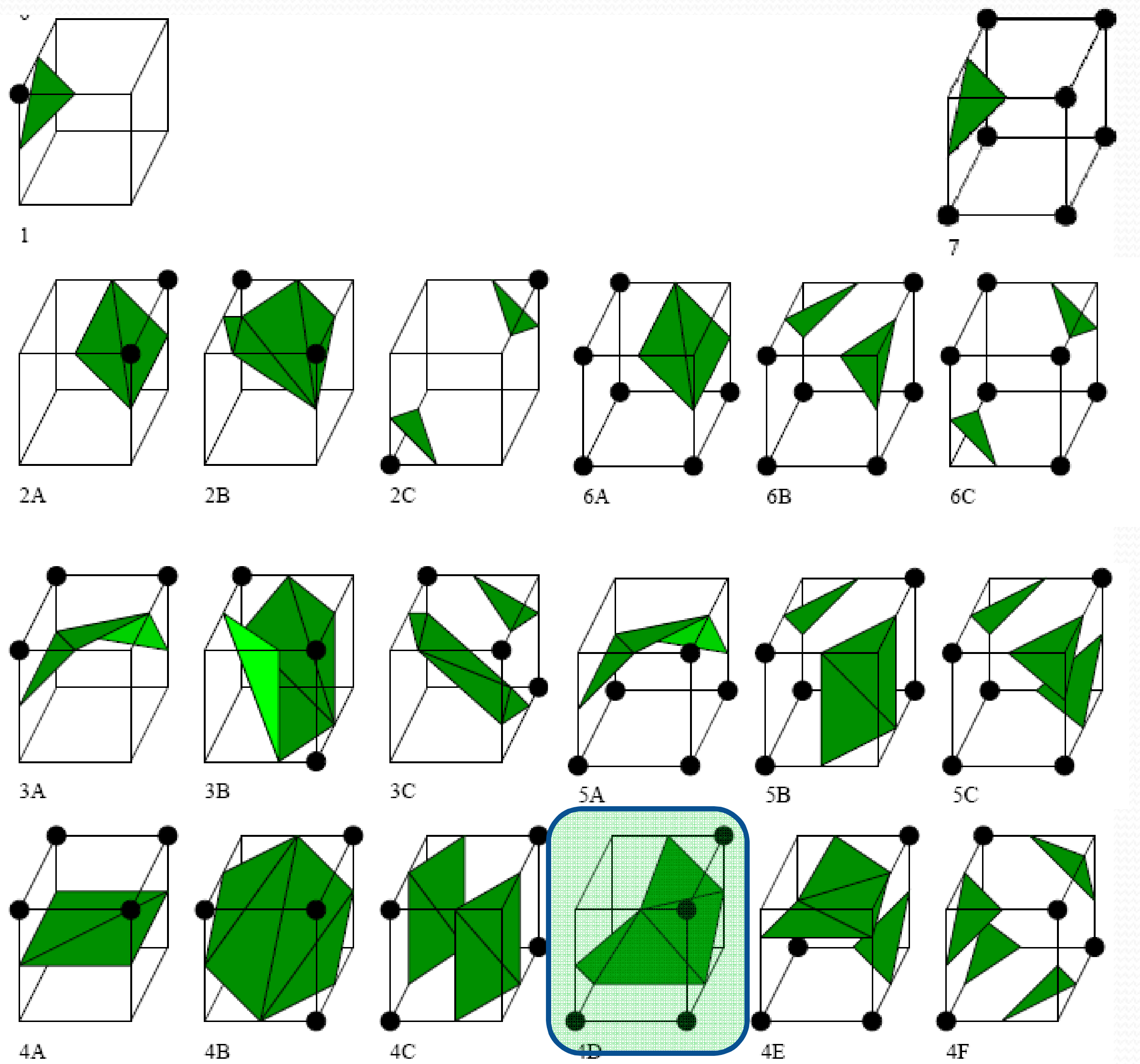
# Modified LUT



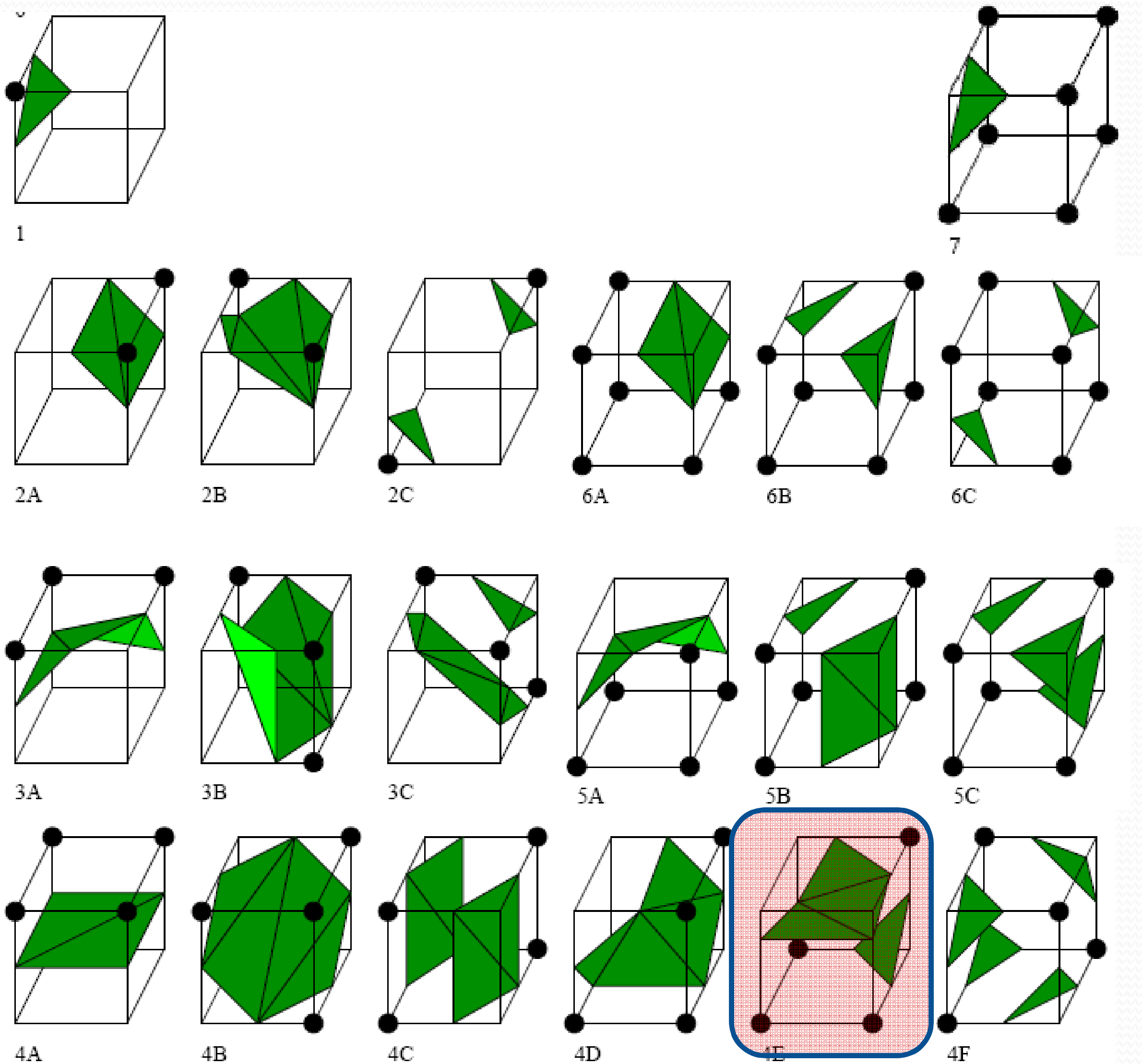
# Modified LUT



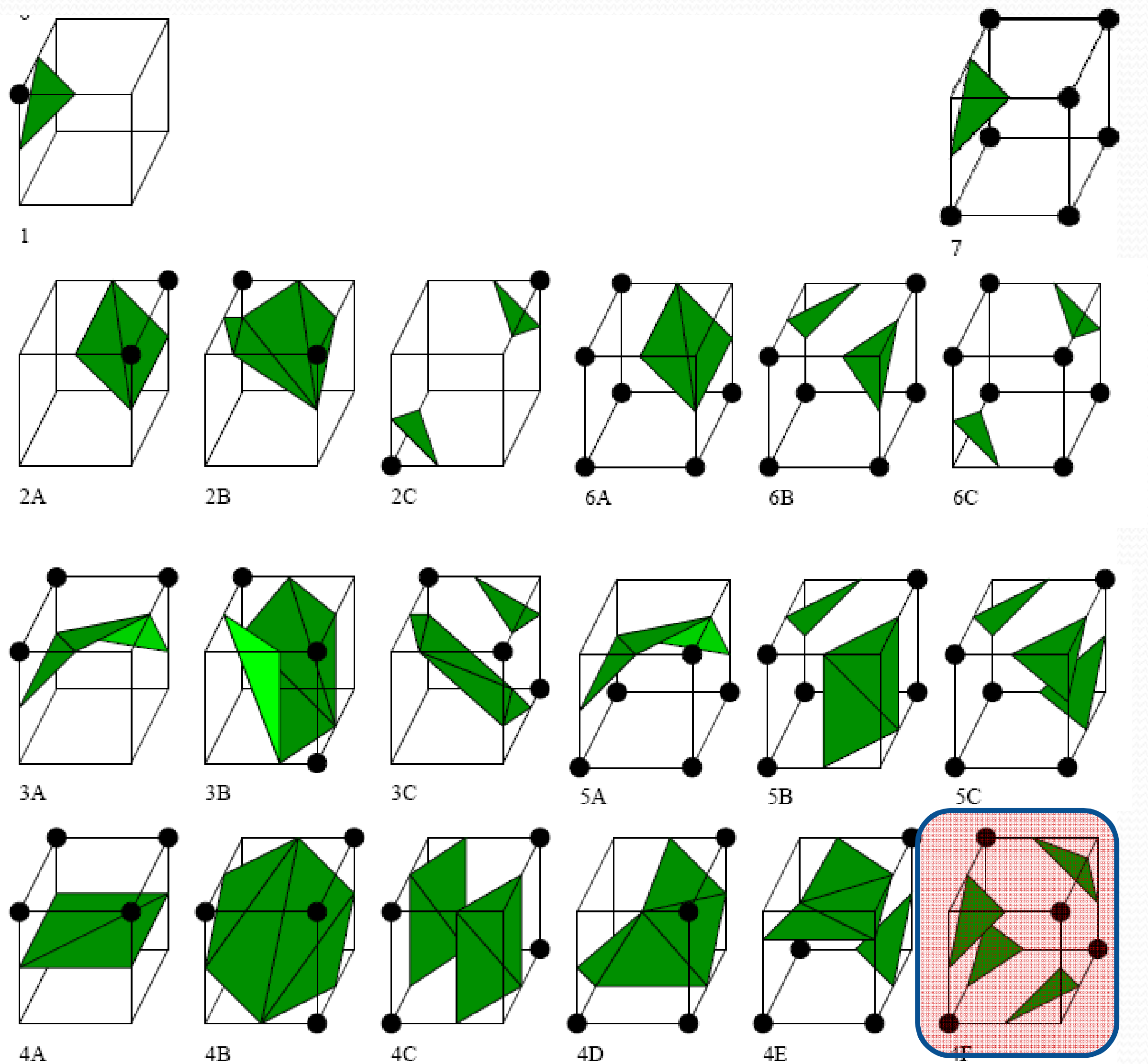
# Modified LUT



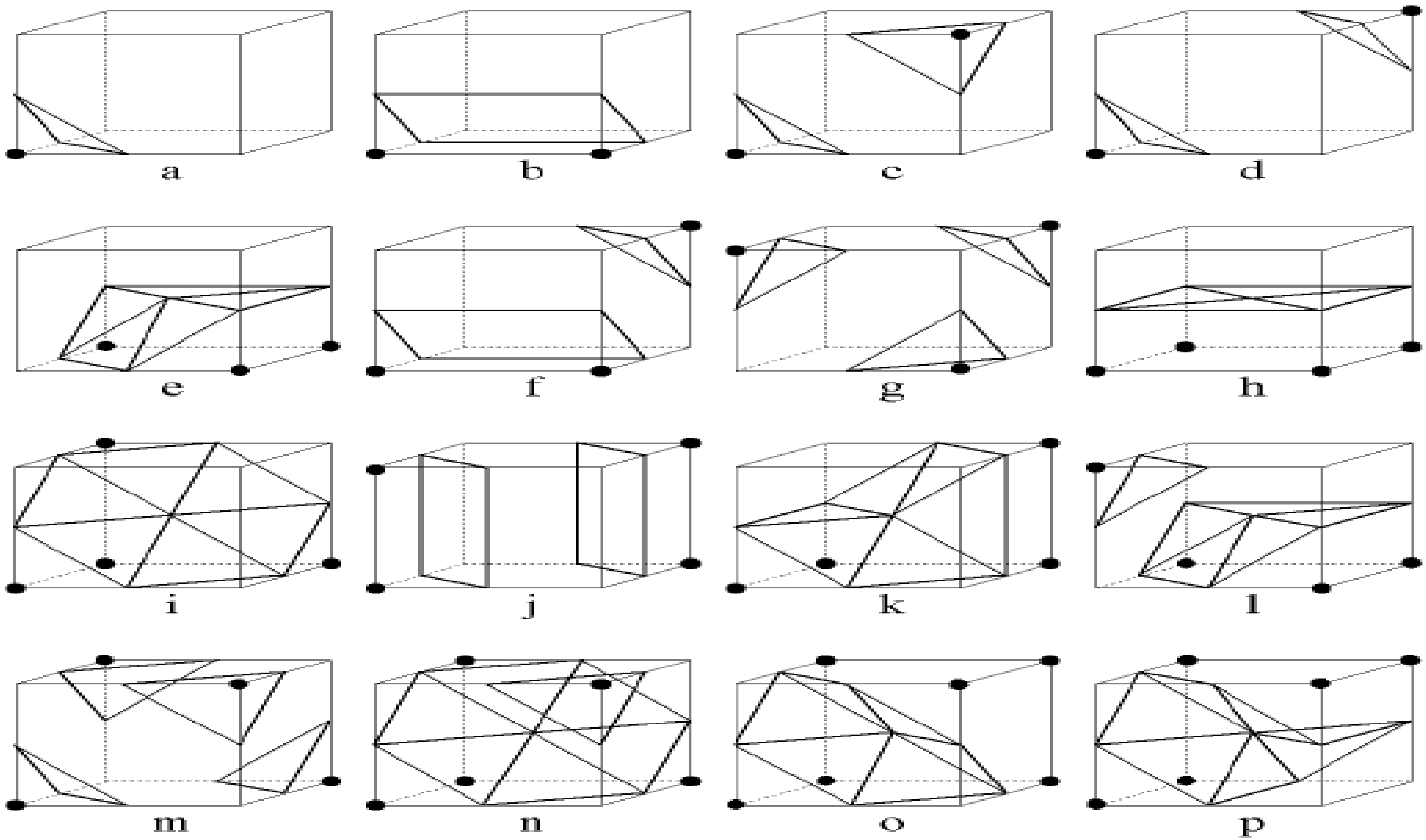
# Modified LUT



# Modified LUT



# Another consistent LUT

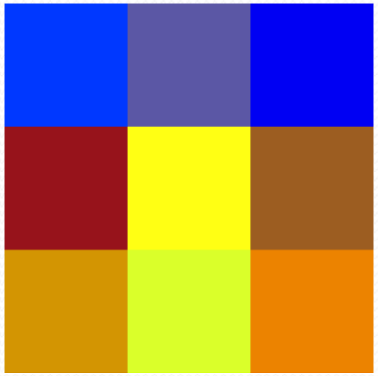


# Connection with pixel concept

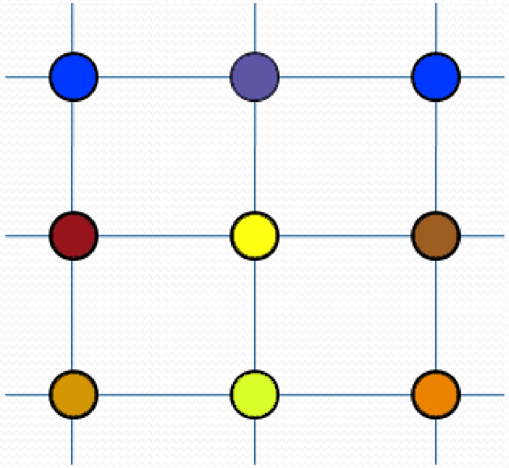
- A digital image is a table with (color, intensity) values
- Each value has two possible interpretations, depending on the pixel definition:

| | | |
|---|---|---|
| 12 | 23 | 25 |
| 45 | 76 | 123 |
| 28 | 245 | 12 |

Imatge digital



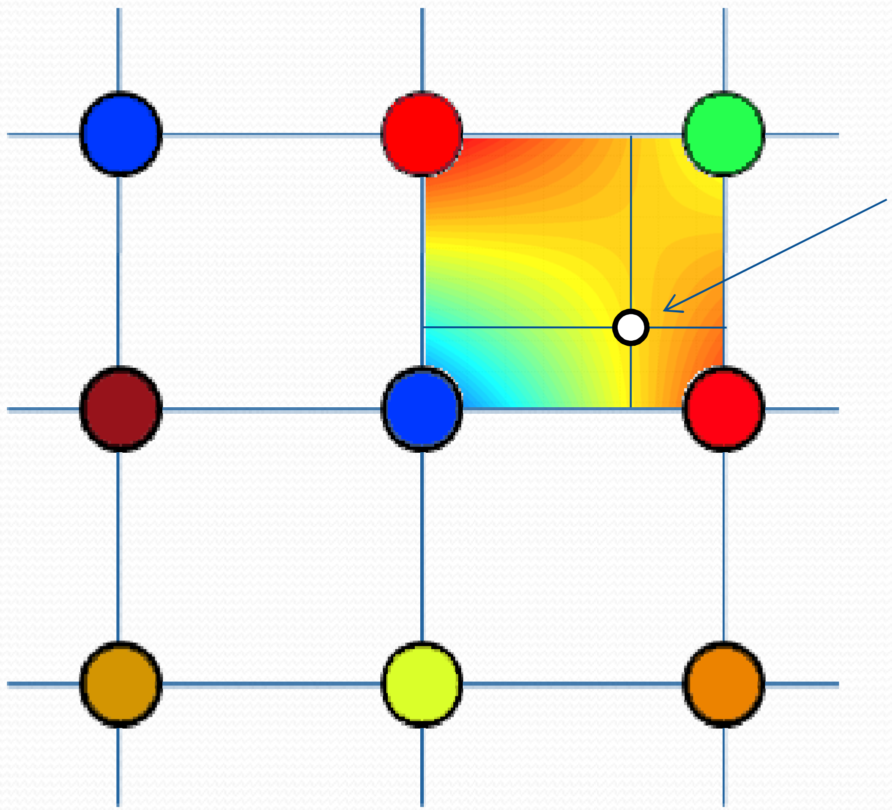
Pixel = regió quadrada



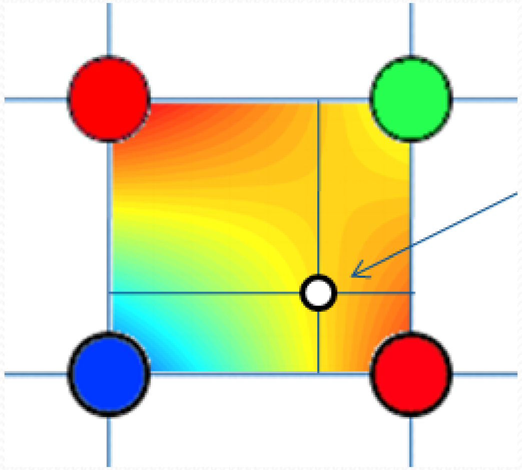
Pixel = mostra puntal

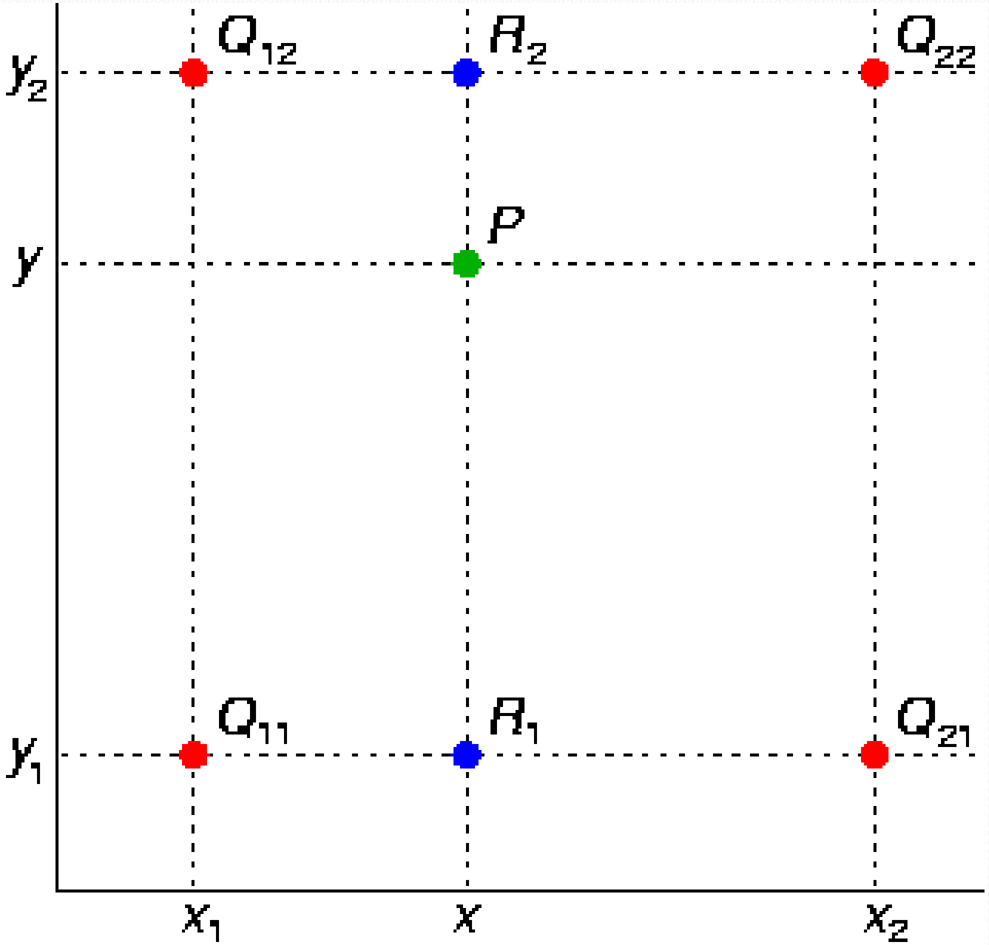
# Connection with pixel concept

- A common interpolant in the plane is the bilinear *interpolation*

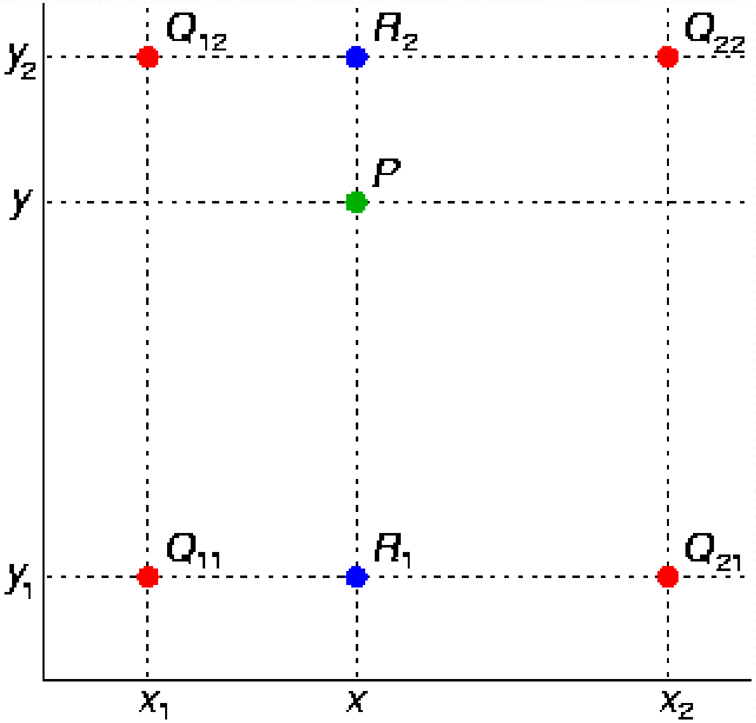


# Bilinear interpolation





# Bilinear interpolation




$$
\begin{aligned}
f(x,y) \approx\ & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) \\
& + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) \\
& + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\
& + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1).
\end{aligned}
$$

# Connection with pixel concept

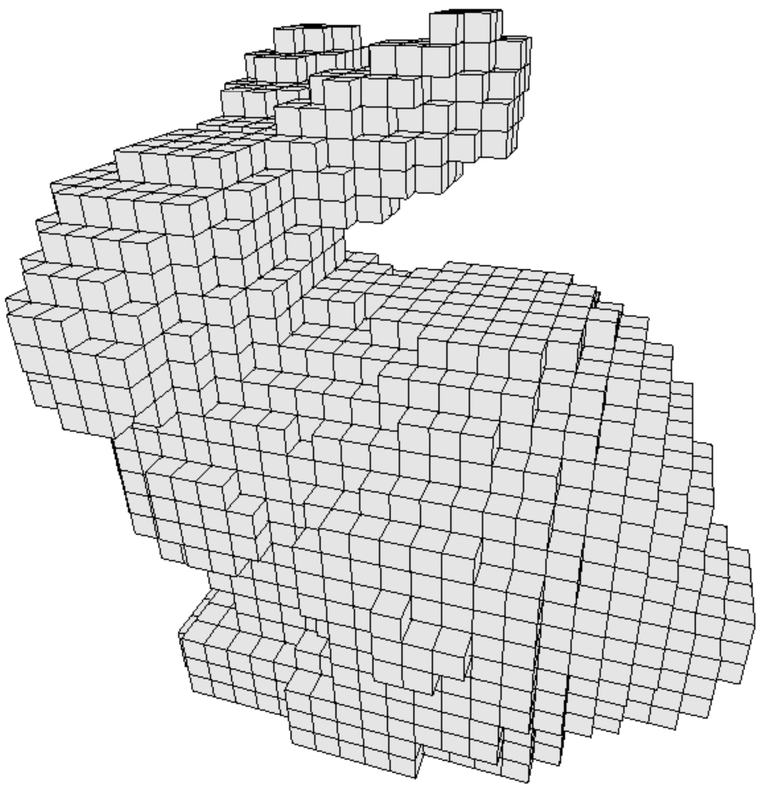
- Zoom

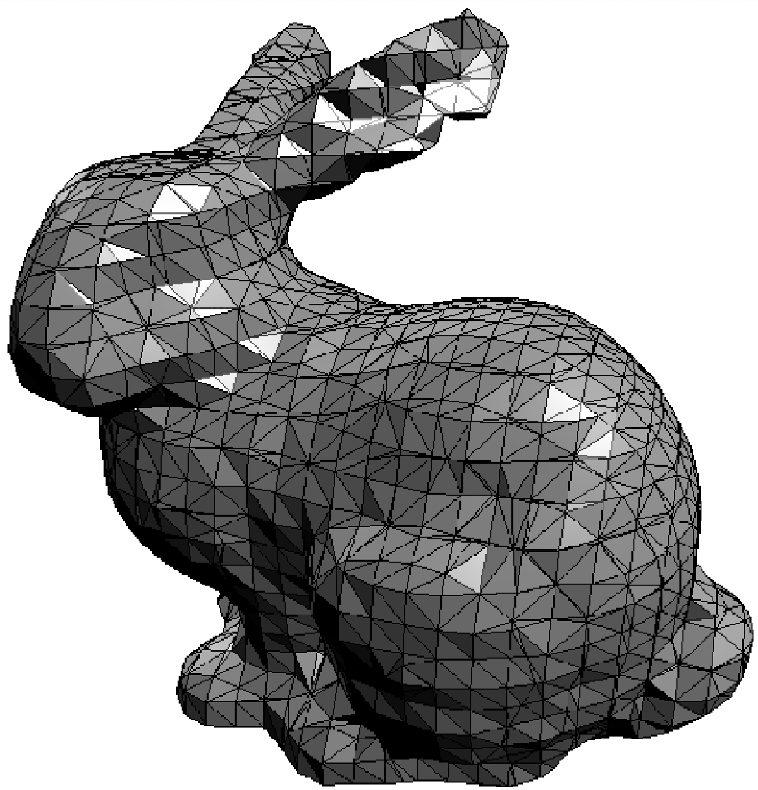
Pixel = regió quadrada

Pixel = mostra puntal; interpolació bilinial

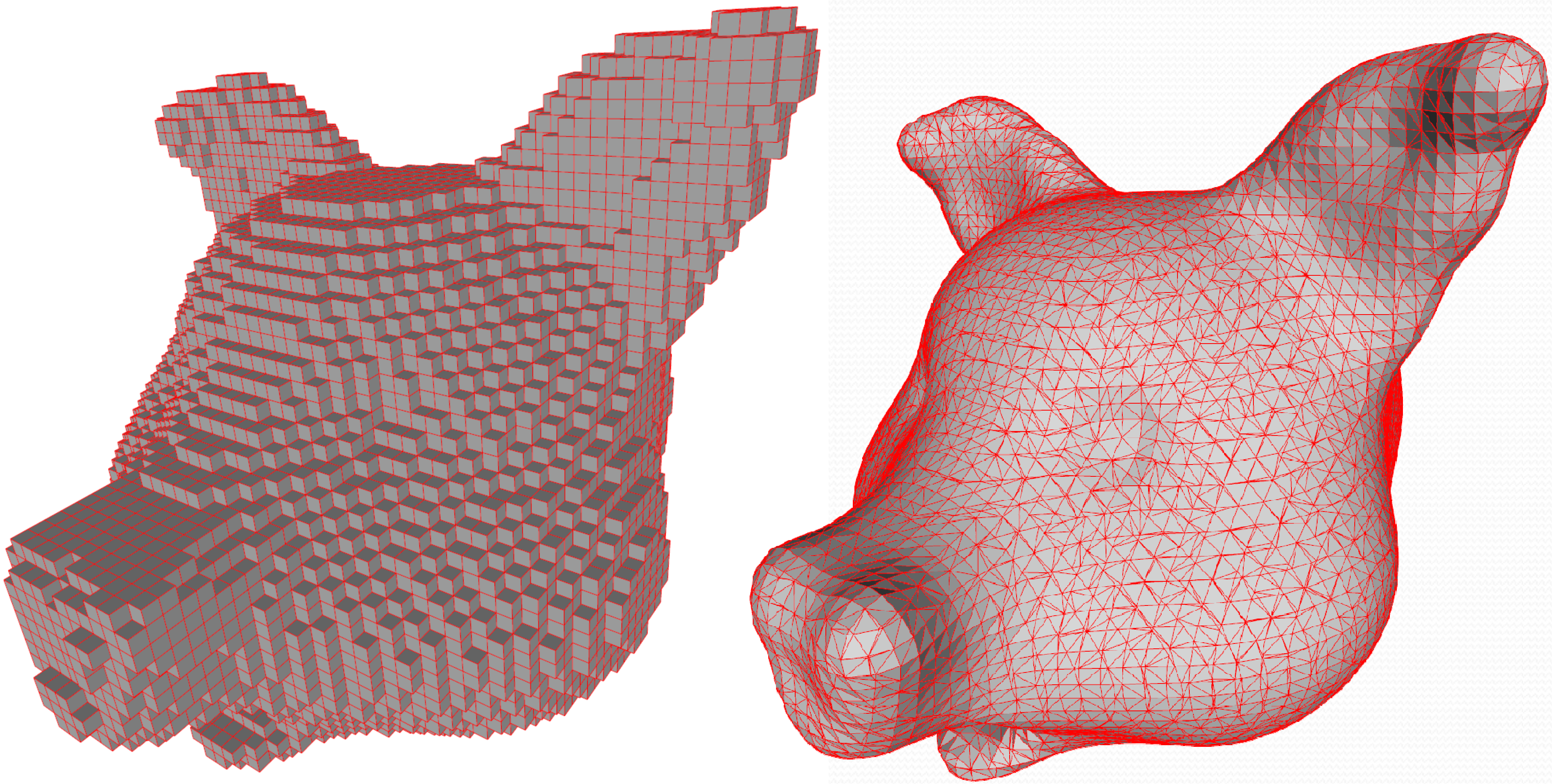
# Marching Cubes

- Marching Cubes was a big improvement over cuberille:





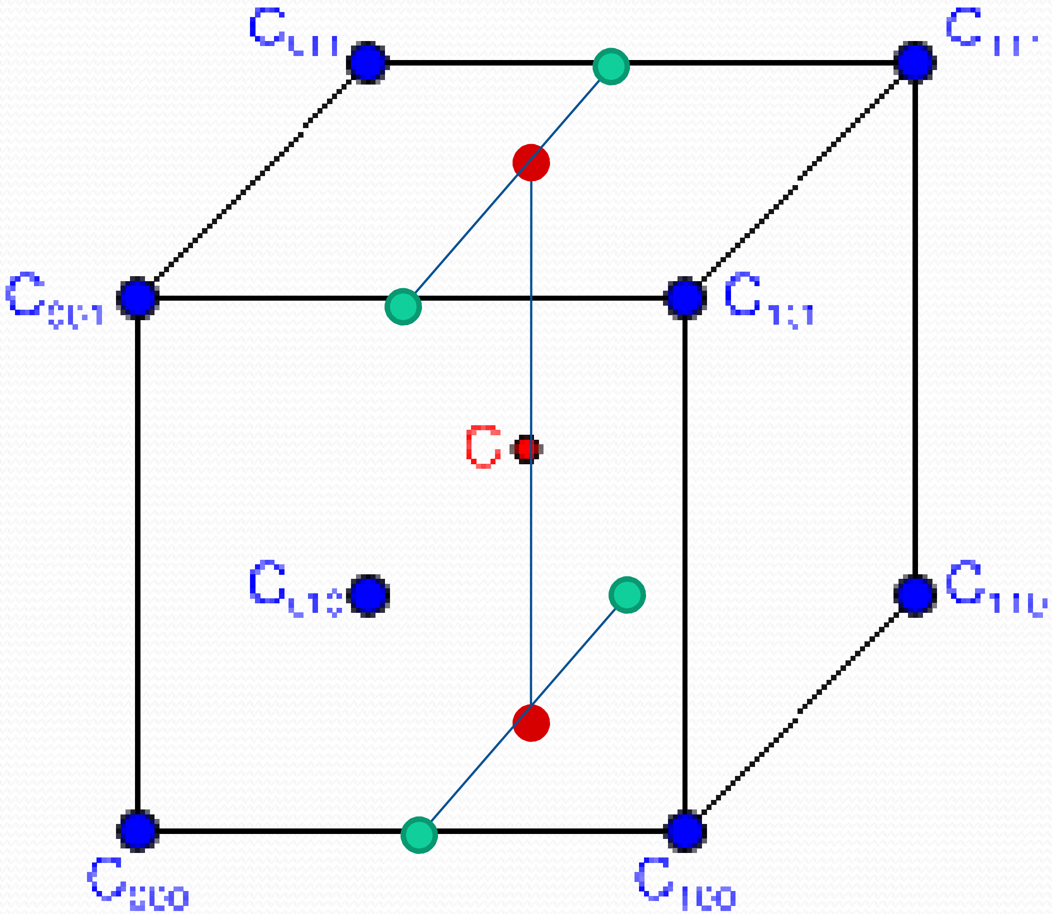
# Marching Cubes



# Interpolation in MC

- MC uses:
  - *Linear interpolation* to fix the position of vertices along edges
  - A linear surface (triangles) to join these vertices

# Trilinear interpolation




[Demo Lewiner]