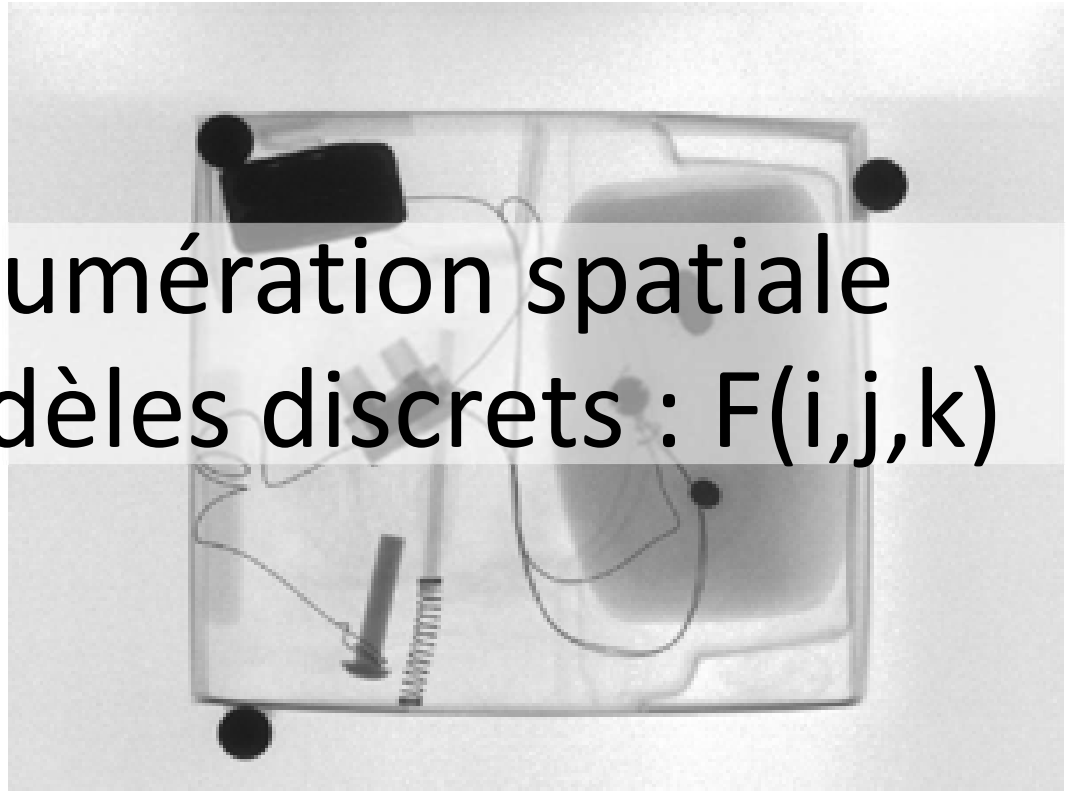
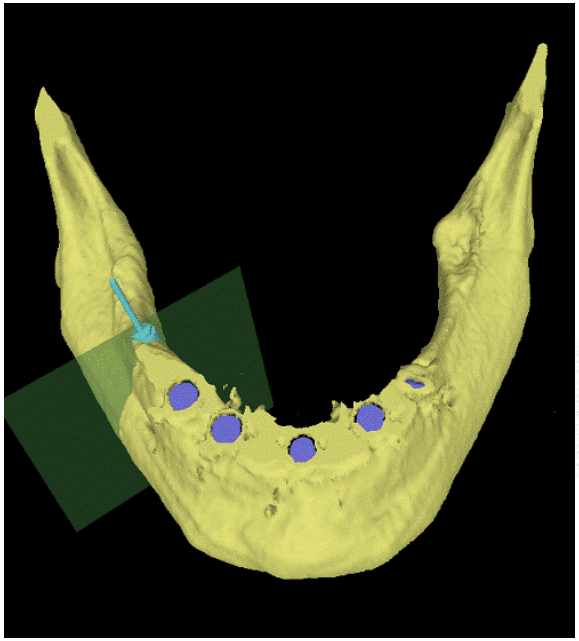
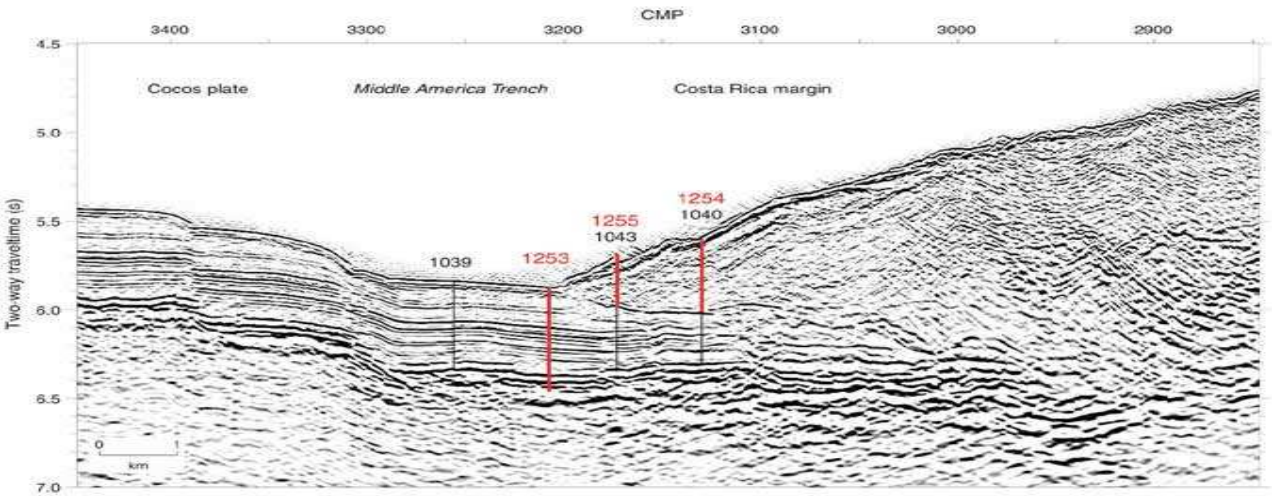
L ’énumération spatiale ou modèles discrets : F(i,j,k)

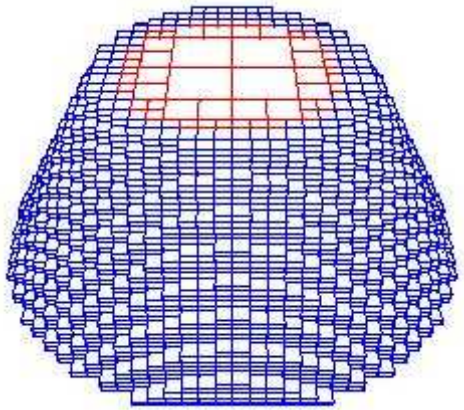


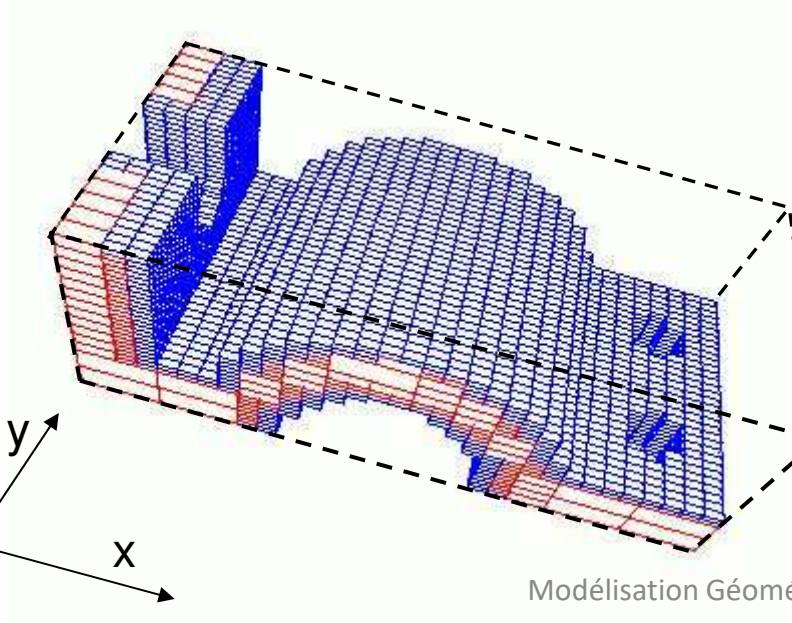


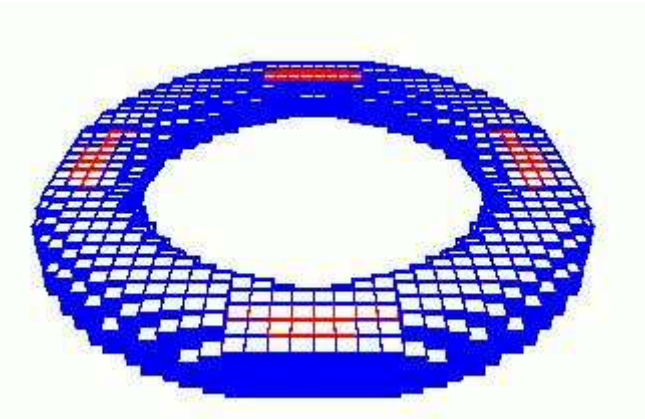


Les modèles d'énumération

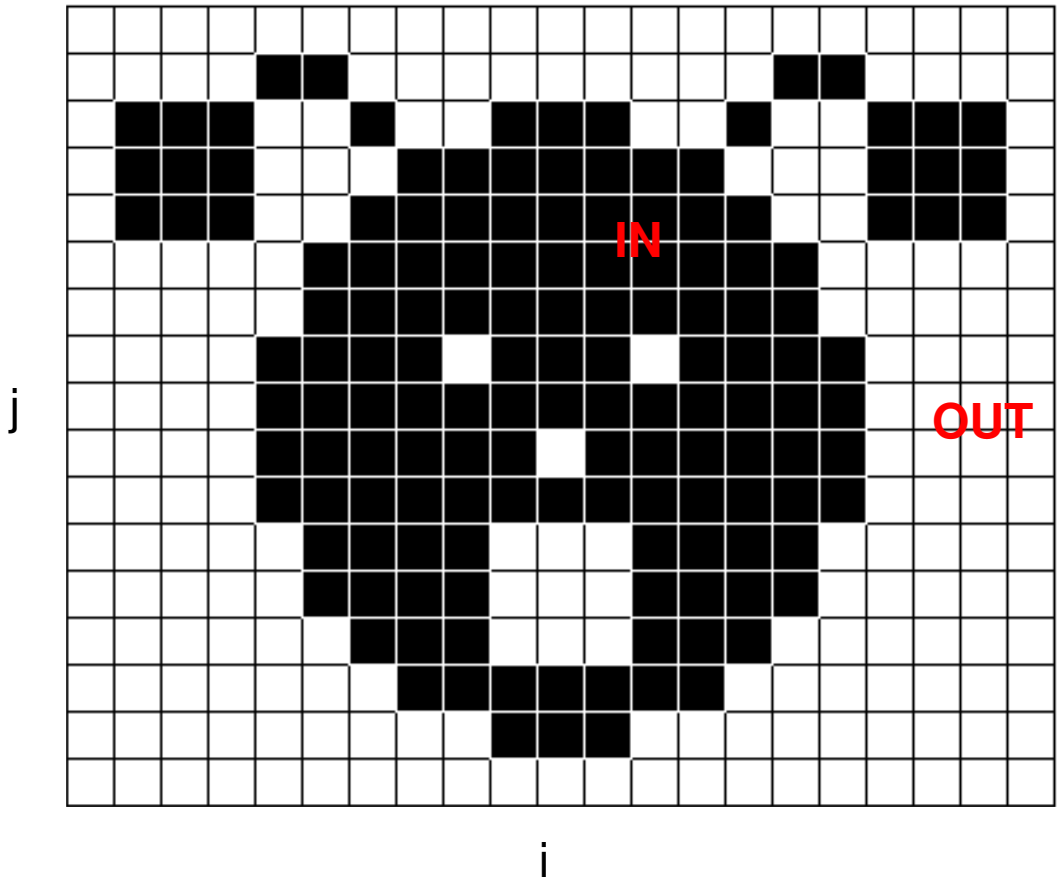
1. La grille (l'image), le voxet
2. Le quadtree, l 'octree
3. Le polytree (MC)





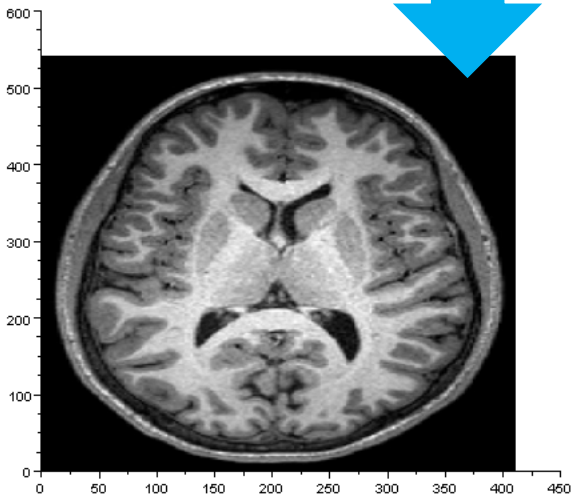


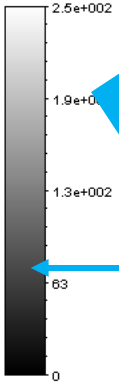
La grille (le principe)



«Reconstruction» géométrique

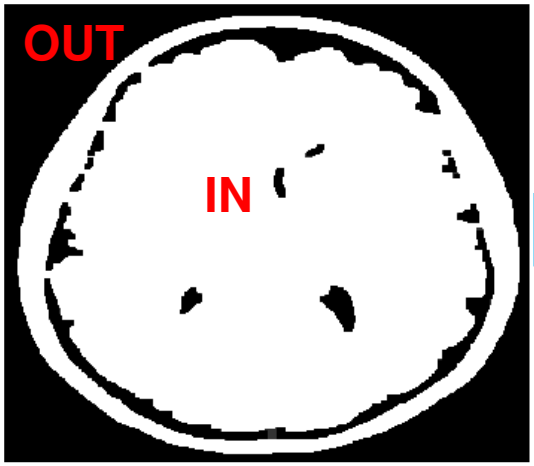
Acquisition





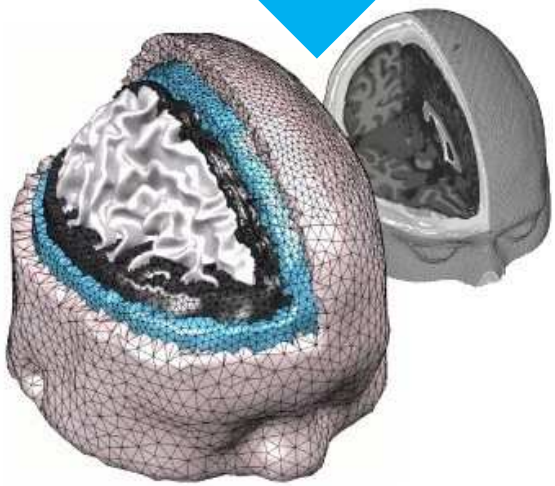
Seuillage

seuil



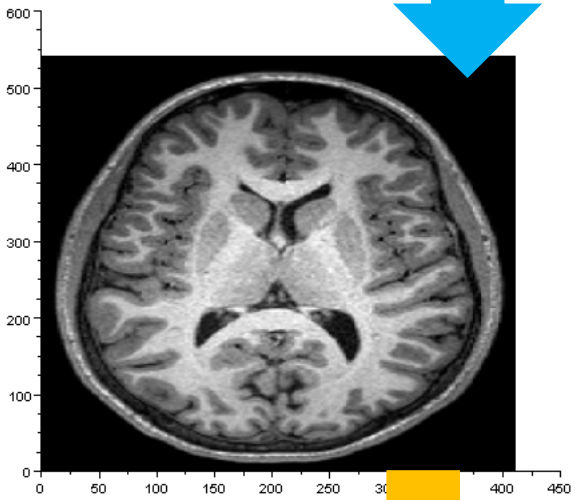
Exploitation

Conversion

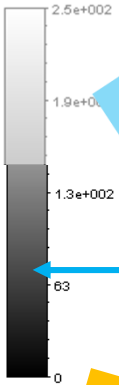


«Reconstruction» géométrique

Acquisition



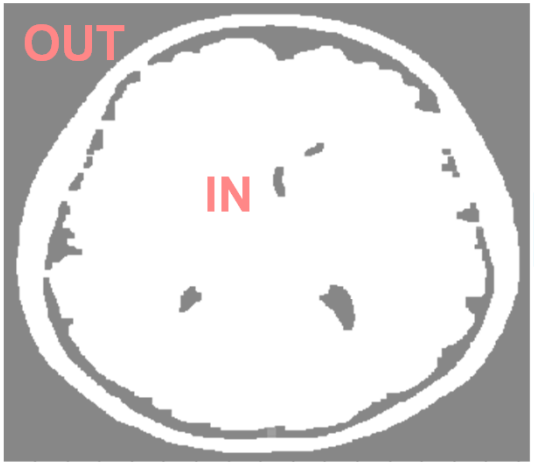
Exploitation



Seuillage

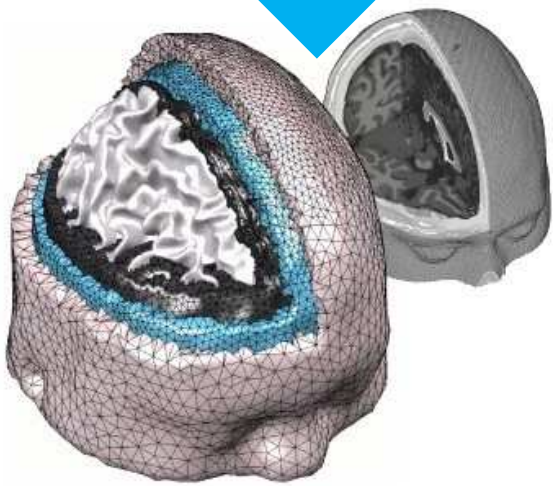
seuil

Conversion

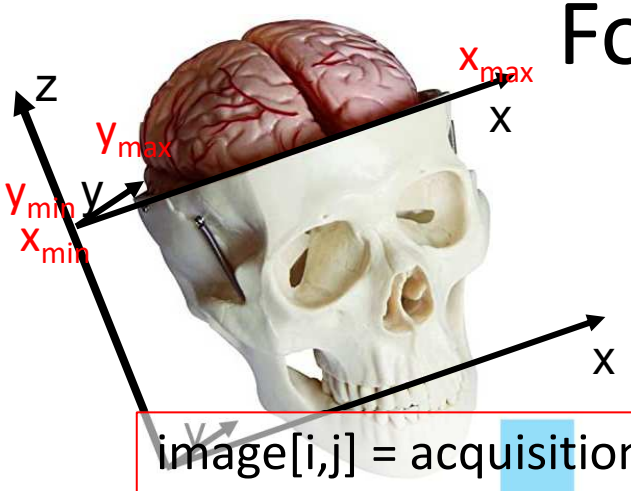


Exploitation

Conversion



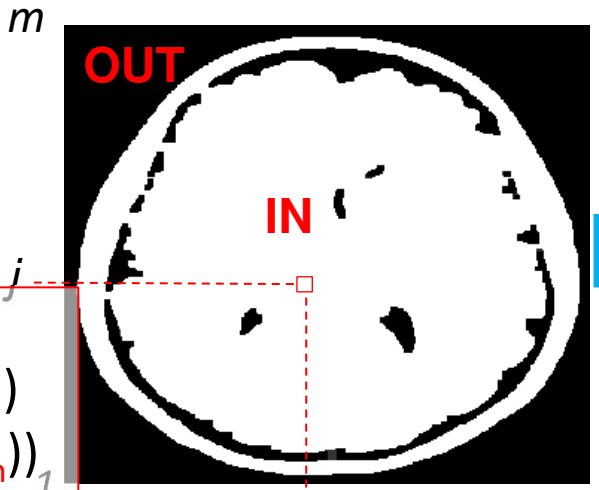
Fonctions de mapping: x,y ➔ i,j



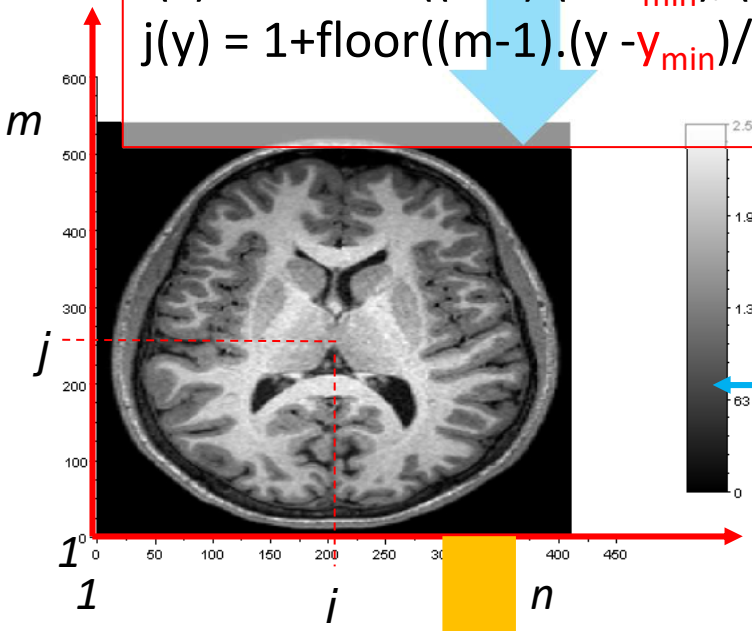
image[i,j] = acquisition(x,y)

$i(x) = 1+floor((n-1).(x - x_{min})/(x_{max} - x_{min}))$

$j(y) = 1+floor((m-1).(y - y_{min})/(y_{max} - y_{min}))$



Exploitation



Seuillage

Normalisation : $[v_{min}, v_{max}]$ ➔ [0,1]

v_{min} = MIN(image[i,j]);

v_{max} = MAX(image[i,j])

imageN[i,j] = (image[i,j] –v_{min})/(v_{max} - v_{min}))

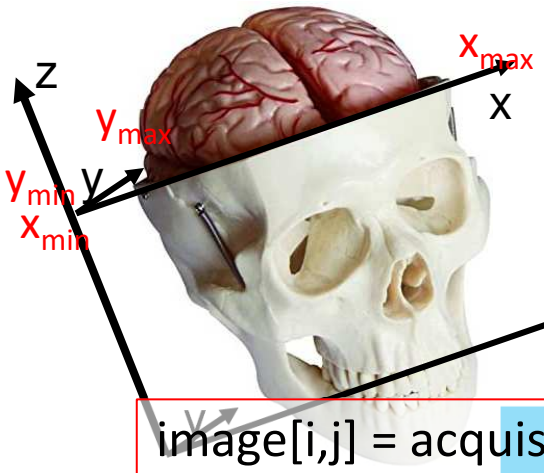
Seuil

Seuillage : [0.,1.] ➔ {0,1} ou {FAUX, VRAI}

imageSeuillée[i,j] = floor(1+imageN[i,j]-seuil)

Exploitation

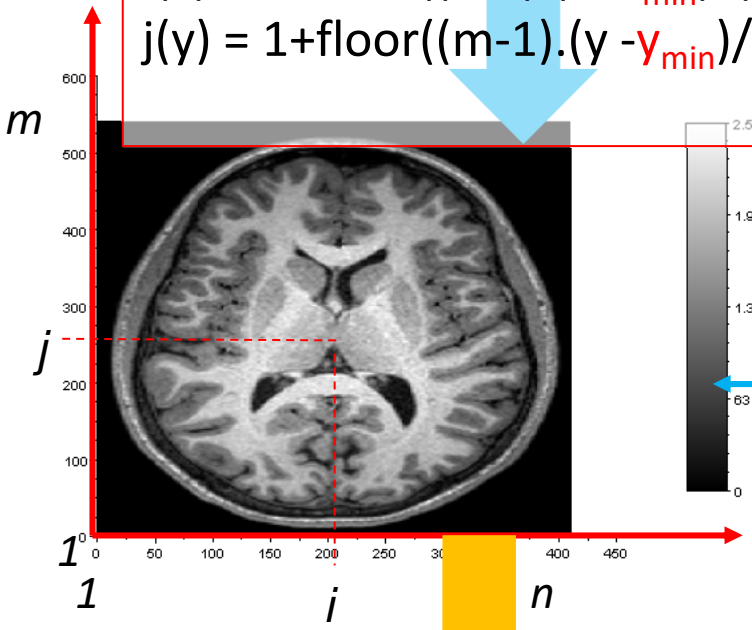
PointDans(x,y) ?



image[i,j] = acquisition(x,y)

i(x) = 1+floor((n-1).(x -x_{min})/(x_{max} - x_{min}))

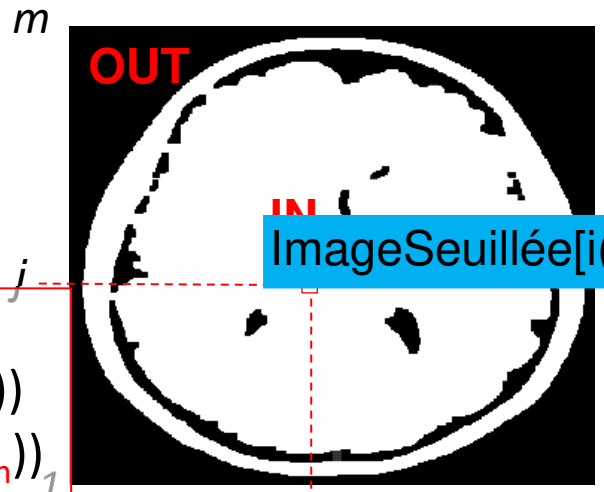
j(y) = 1+floor((m-1).(y -y_{min})/(y_{max} - y_{min}))



Seuillage

seuil

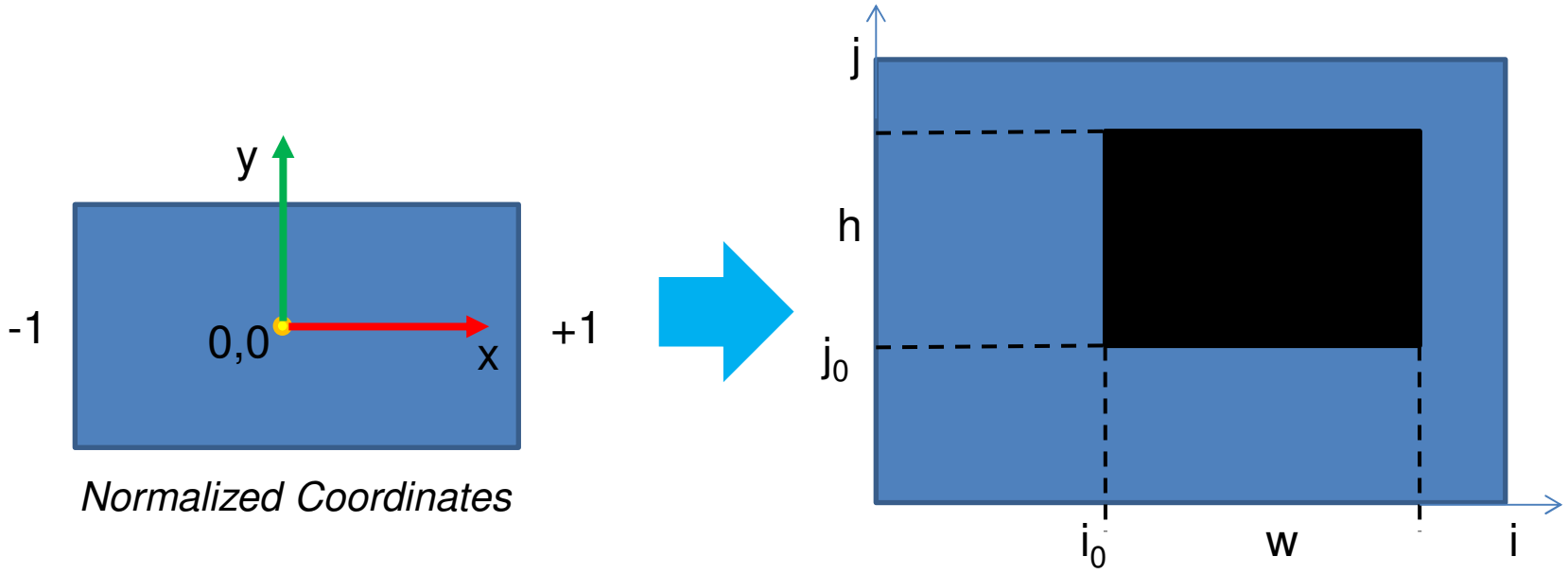
Image[i(x),j(y)] >= seuil ?



ImageSeuillée[i(x),j(y)] ==1 ?

Exercice : OpenGL glviewport

- Ecrire la transformation de l'espace normalisé (d'OpenGL) vers la fenêtre (viewport) défini par (i_0, j_0, w, h)

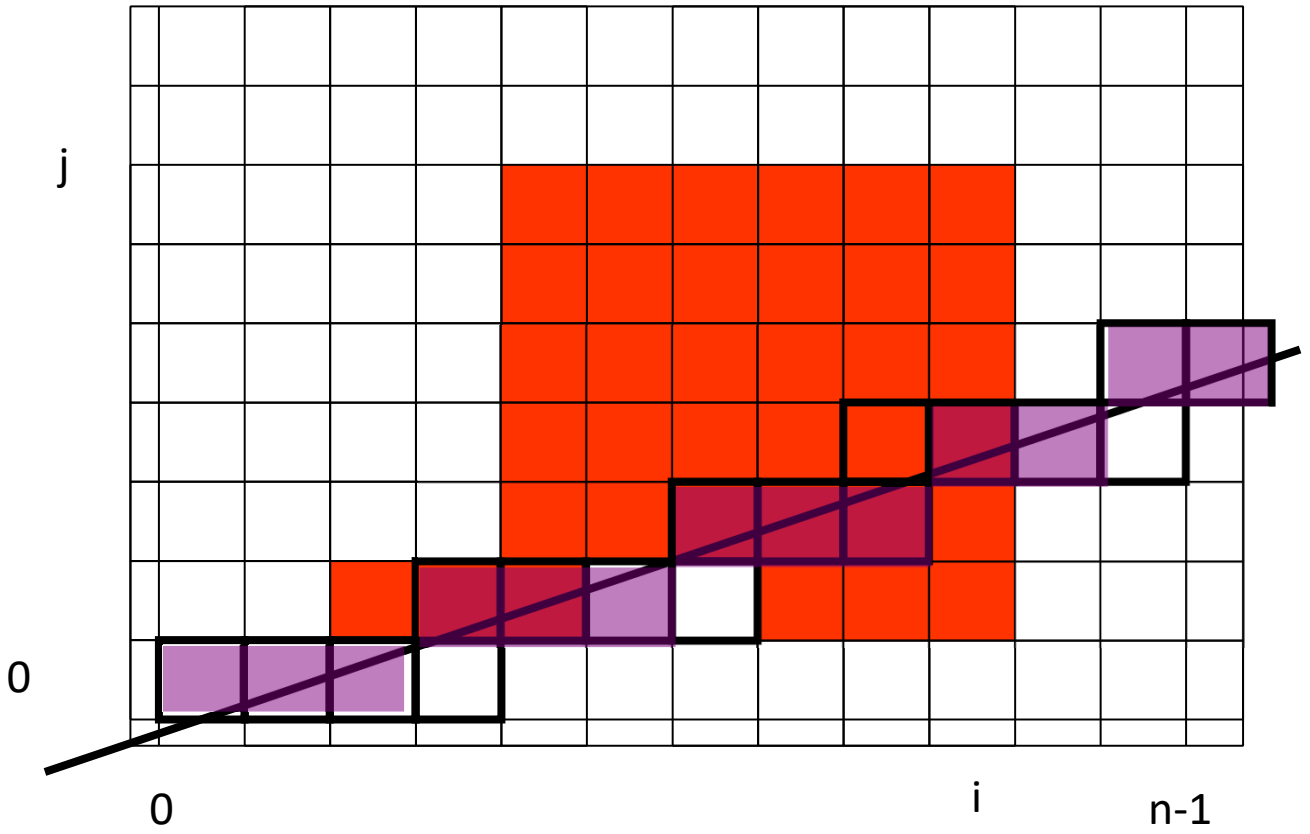


Normalized Coordinates

void glViewport(GLint i0, GLint j0,
GLsizei w, GLsizei h);

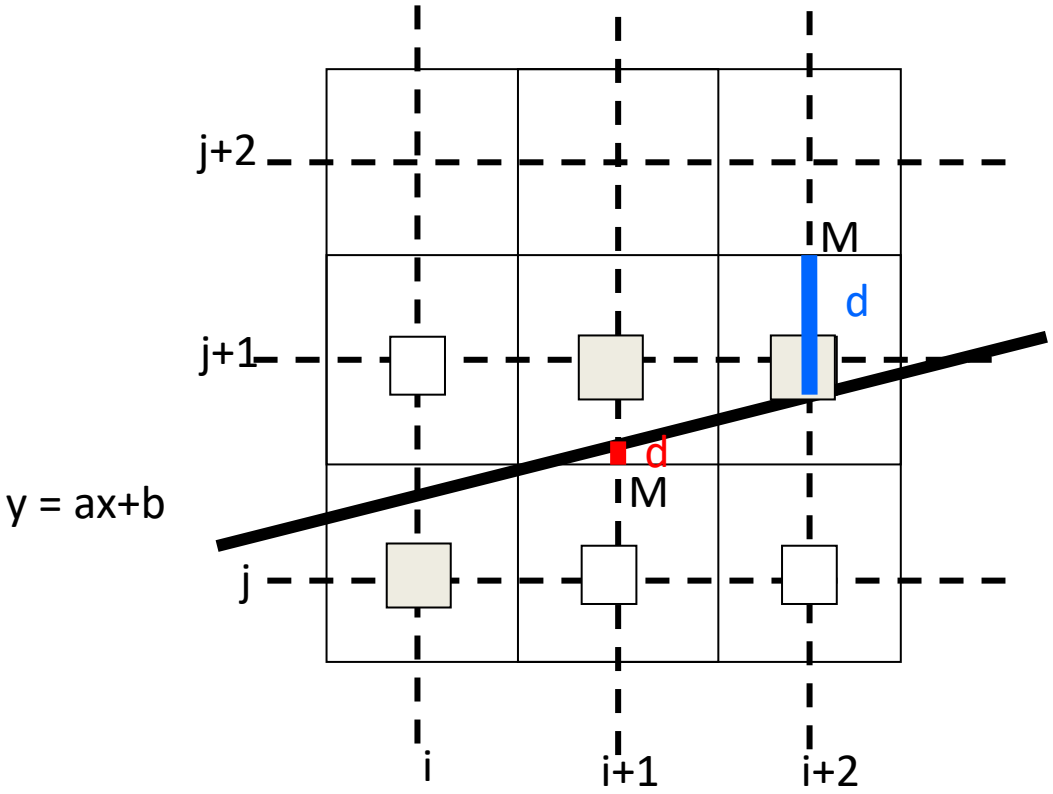
i(x) = …
j(y) = …

Intersection avec une droite



Tracé d'un segment

(Mid-point Algorithm)



Cas : 0<a<1

```
WritePixel(i0,j0)
d=(a*(i0+1)+b)-(j0+1/2)
For(i=i0+1;i<=i1;i++){
    Si (d<0) d=d+a
    Sinon    j=j+1
             d=d+a-1
    WritePixel(i,j)
}
```

Calculs & opérations sur les grilles

Algorithmes simples

- Localisation « PointDans() ? »
- Calcul de l’aire ?
- Opérations booléennes ?

Calculs & opérations sur les grilles

Algorithmes simples

- Localisation « PointDans() ? »
- Calcul de l'aire
- Opérations booléennes : simple quand les 2 géométries sont dans le même espace (même extension, même « précision »).

Calculs & opérations sur les grilles

Algorithmes simples

- Localisation « PointDans() ? »
- Calcul de l'aire
- Opérations booléennes : simple quand les 2 géométries sont dans le même espace (même extension, même « précision »).

Opérations non conservatives

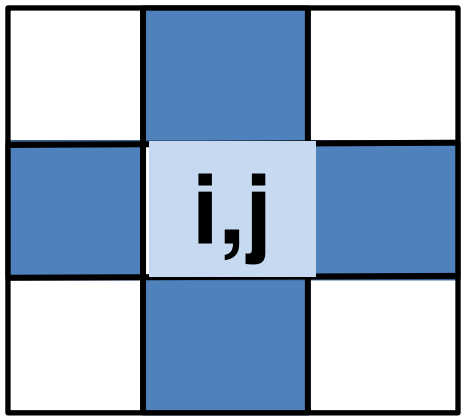
Certaines opérations ne conservent pas les caractéristiques de forme de la géométrie :

- Changement d'échelle (taille des cellules)
- Rotations

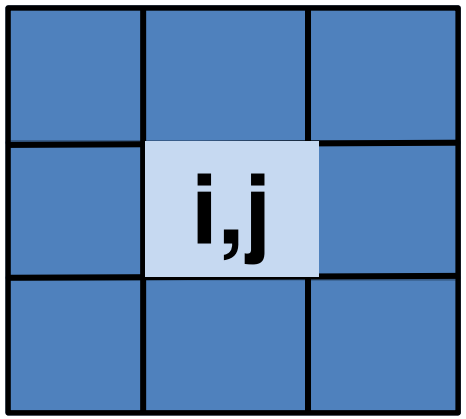
Topologie et voisinage dans des espaces discrets

Intérieur : C(i,j) intérieur à X si son ***voisinage*** *strictement inclus dans X*
Adhérence : C(i,j) adhérent à X si son ***voisinage*** *intersecte X*
Frontière : C(i,j) appartient à la frontière de X
ssi C(i,j)dans X et son ***voisinage*** *intersecte* $C_E(X)$

Voisinage 4 connexité



Voisinage 8 connexité



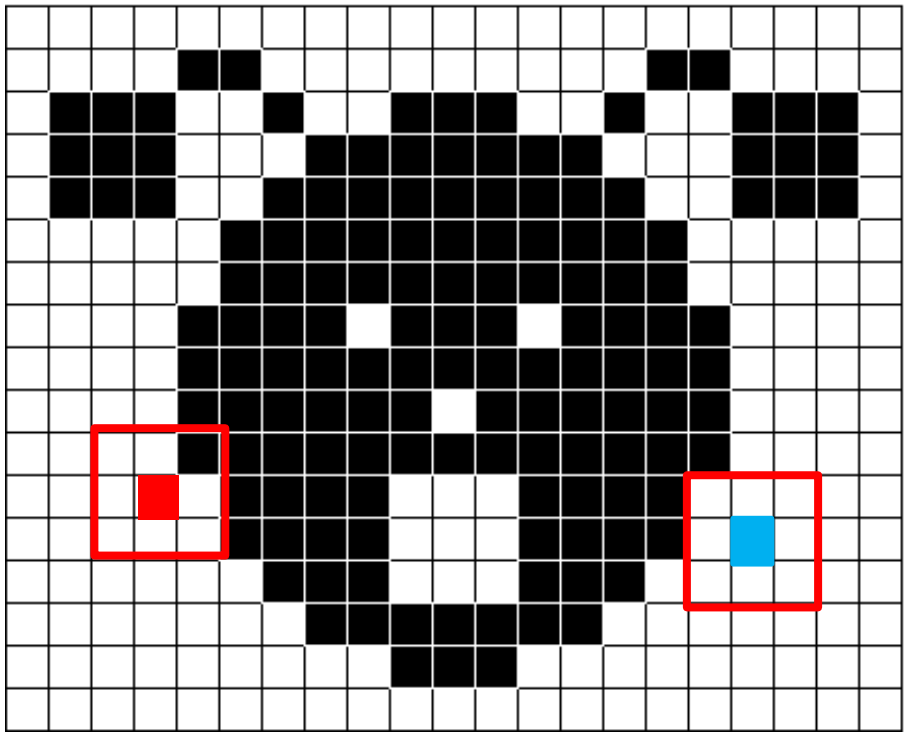
ATTENTION : Dans ≠ intérieur

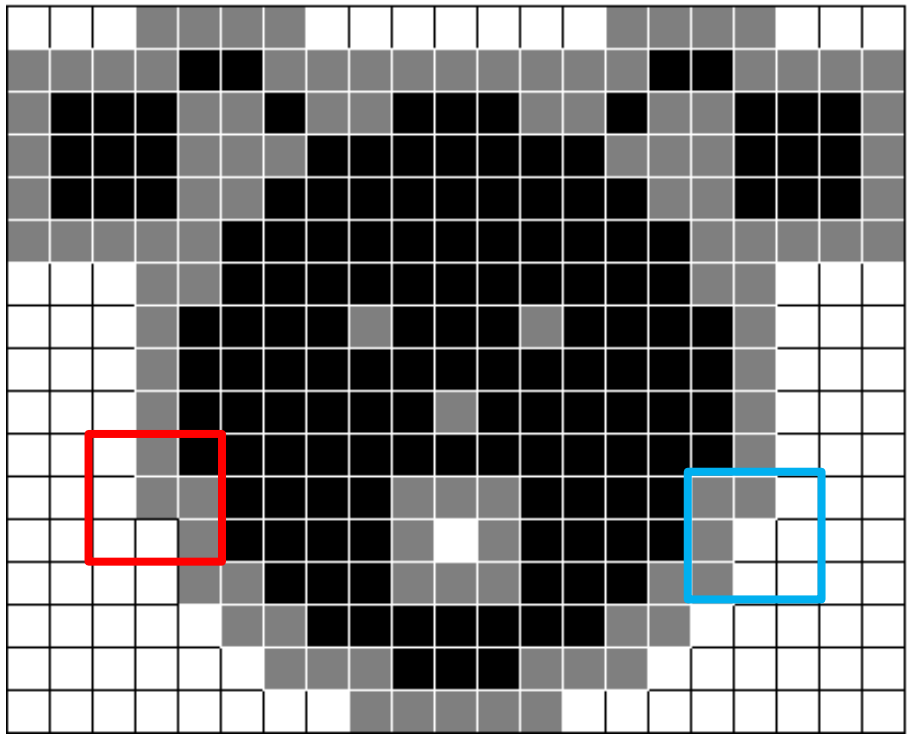
Dilatation $\delta_B(X)$

La dilatation morphologique de *X* avec un élément structurant *B* est définie comme la somme de Minkowski avec cet élément

B

$$\delta_B(X) = X \oplus B \qquad \delta_B(X) = \{x \mid \check{B}_x \cap X \neq \emptyset\}$$





(from wikipedia)

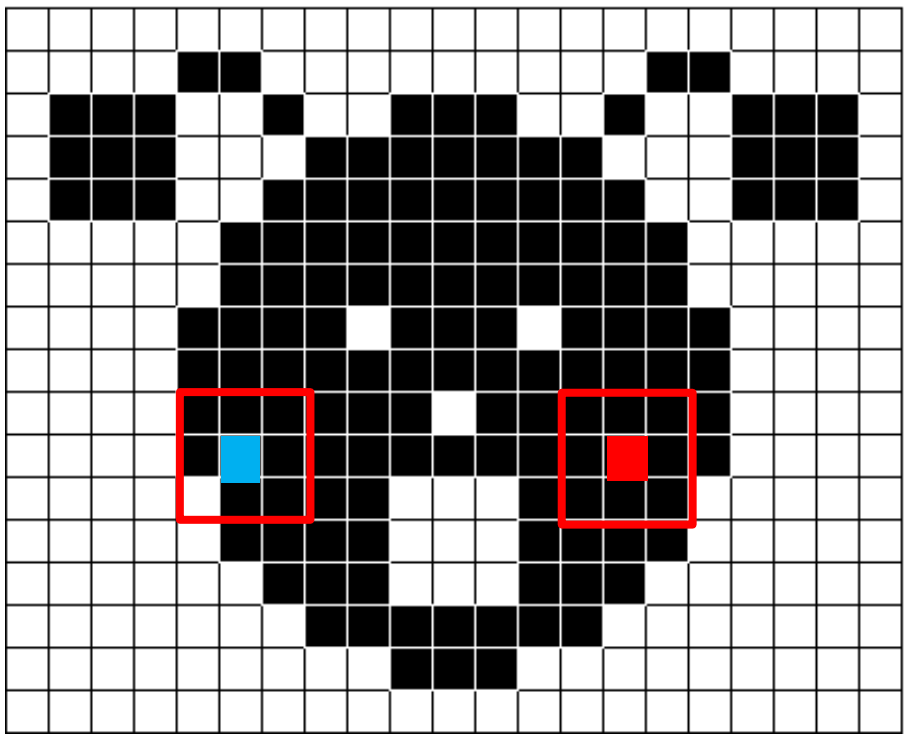
Sur cet exemple c'est aussi **l'Adhérence** pour le 8-voisinage

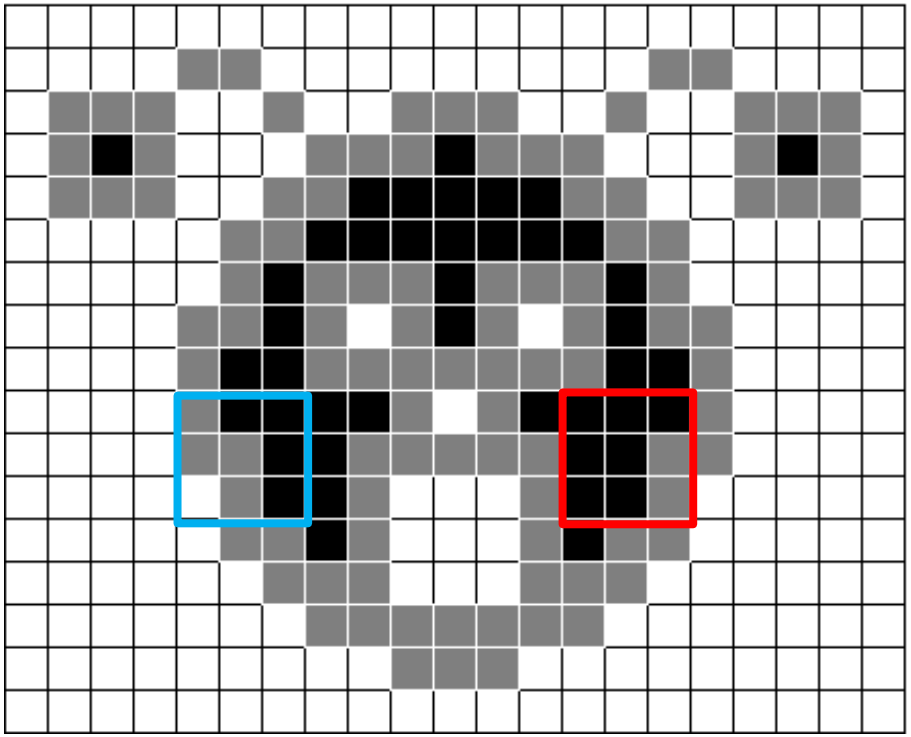
Erosion $\varepsilon_B(X)$

L'érosion morphologique avec un élément structurant B est définie comme l'ensemble des points de X qui contiennent B.

B

$$\epsilon_B(X) = X \ominus B = \{x \mid B_x \subset X\}$$





(from wikipedia)

Sur cet exemple c'est aussi **l'Intérieur** pour le 8-voisinage

Morphologie mathématique

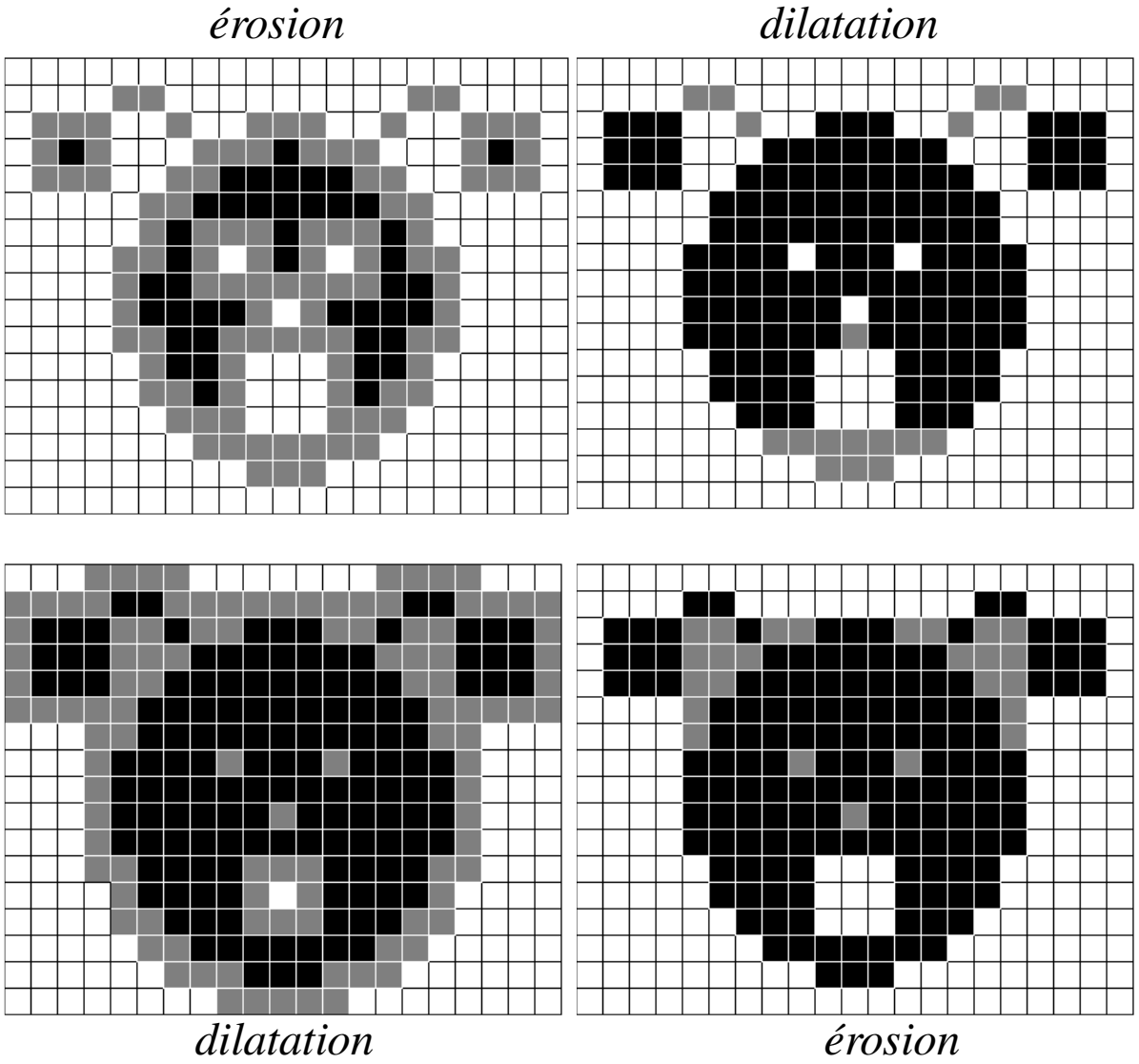
Filtres morphologique

- ouverture

$$\delta_B(\varepsilon_B(X))$$

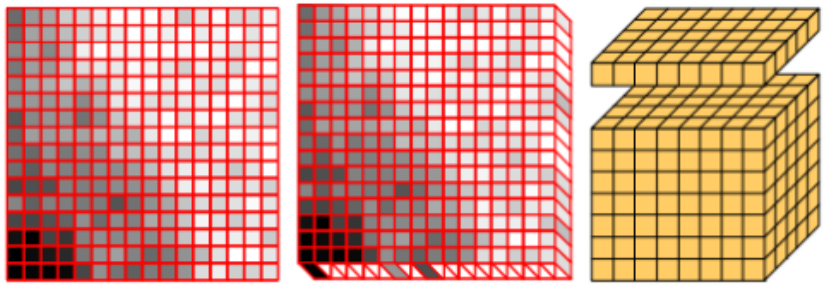
- fermeture

$$\varepsilon_B(\delta_B(X))$$

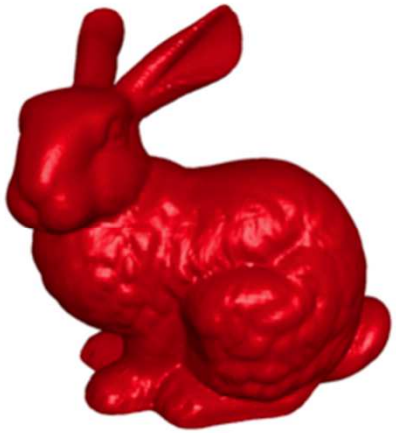


« Grilles 3D » : voxelisation

Acquisition d'une série d'images

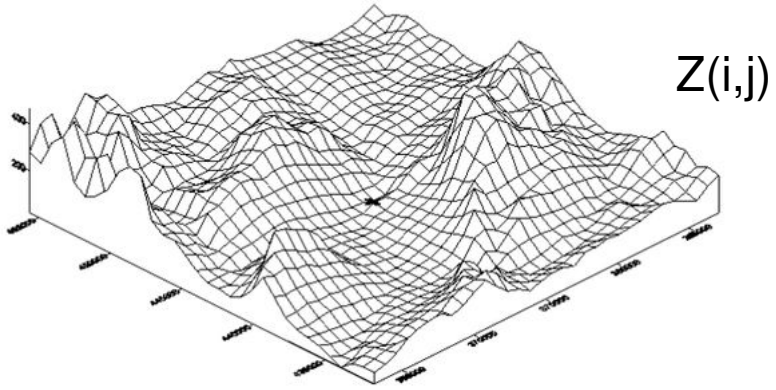


Extraction et visualisation

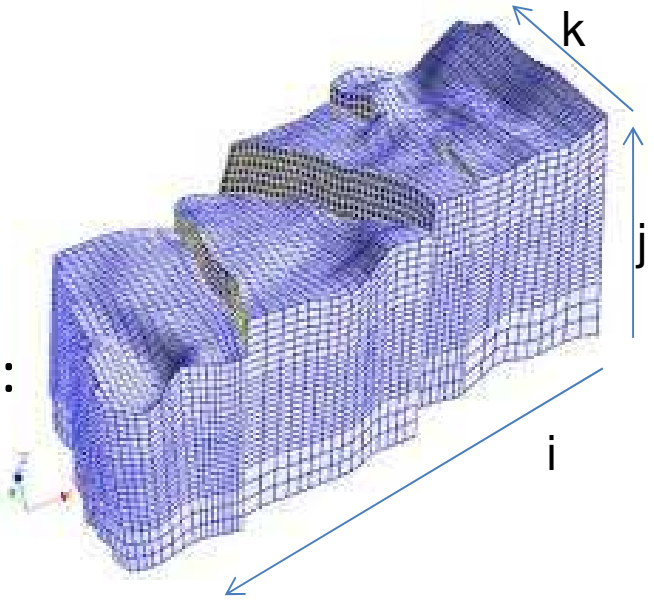


Exemples en Géosciences

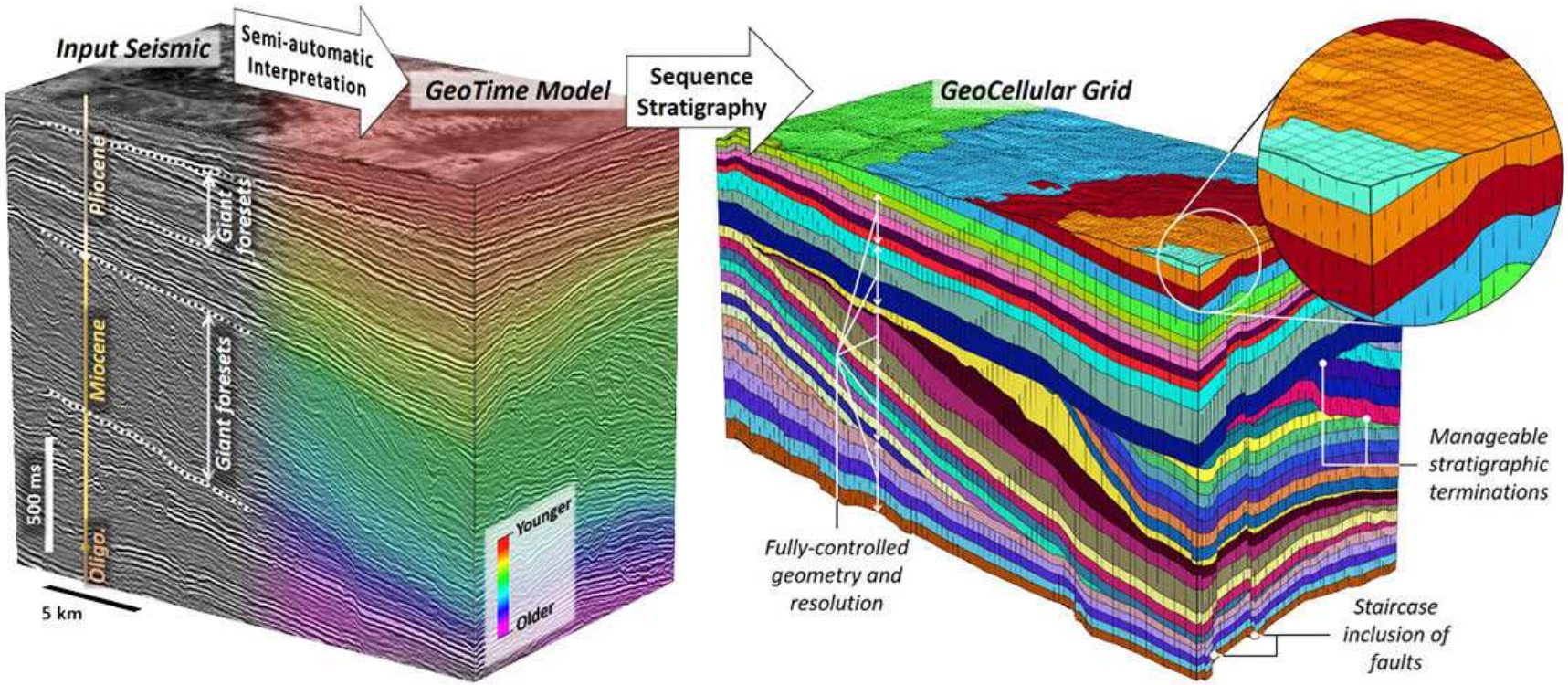
- MNT (DEM)



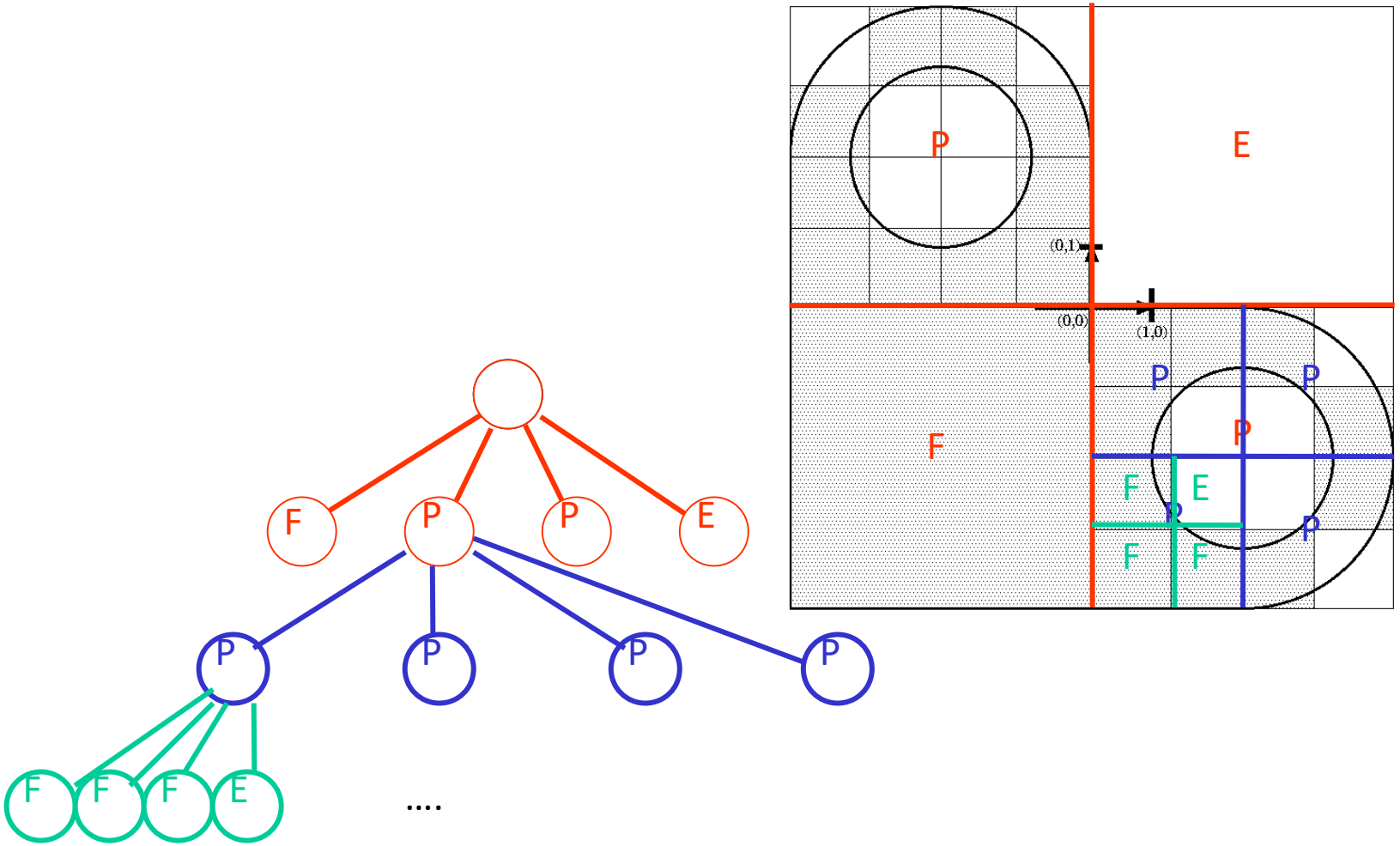
- Grilles stratigraphiques :
  - Déformations / Dicontinuité : mapping complexe,
  - GS(i,j,k) = propriétés de la roche : perméabilité, porosité…
  - Couches incomplètes : nodata



Exemple en géosciences



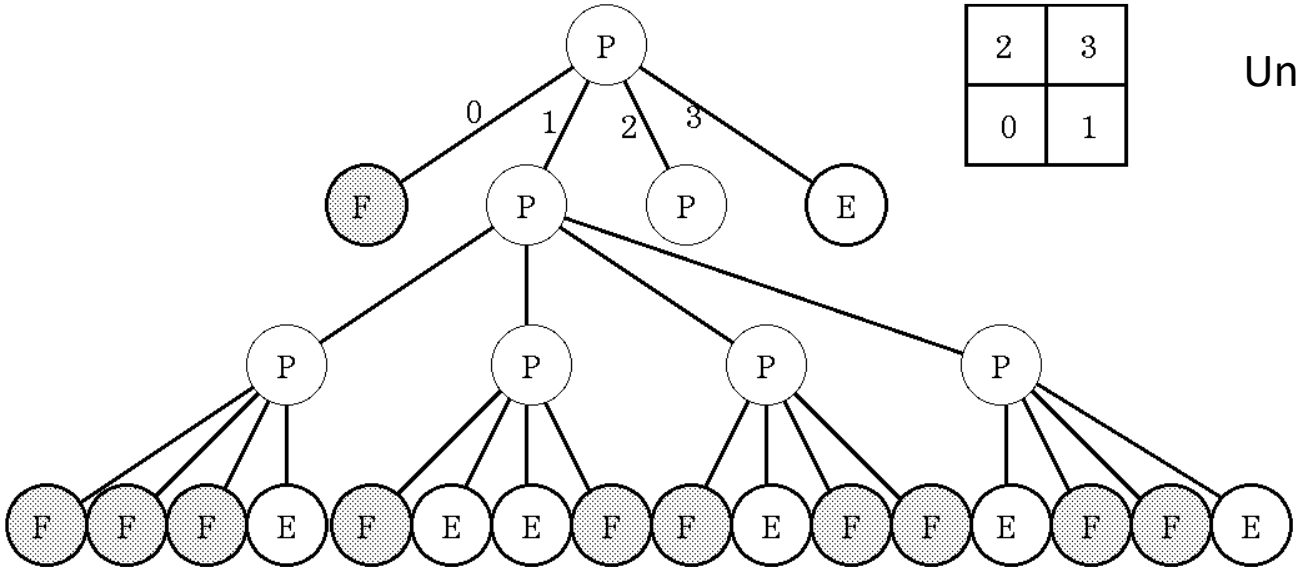
2. Le Quadtree (Principe)



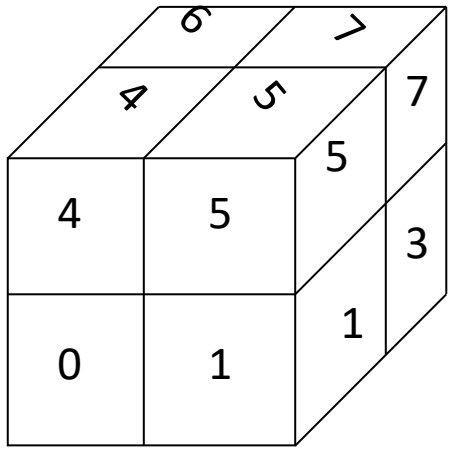
Représentation informatique

D : dimension de l ’espace

Quadtree (D=2), Octree (D=3), Polytree



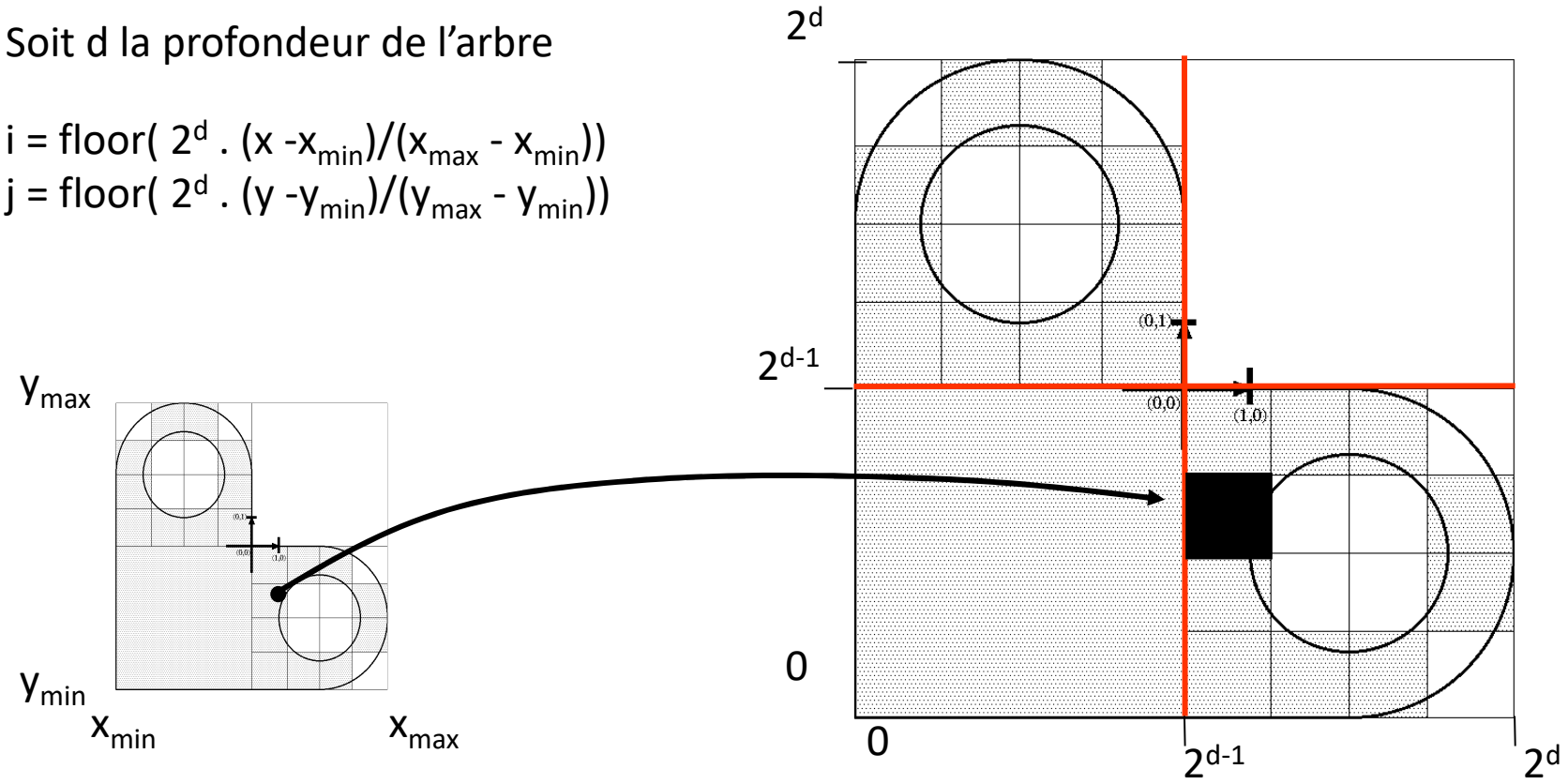
Un arbre : à 2^D branches



Quadtree : profondeur & codage

Soit d la profondeur de l'arbre

$i = \text{floor}(2^d . (x - x_{min})/(x_{max} - x_{min}))$
$j = \text{floor}(2^d . (y - y_{min})/(y_{max} - y_{min}))$

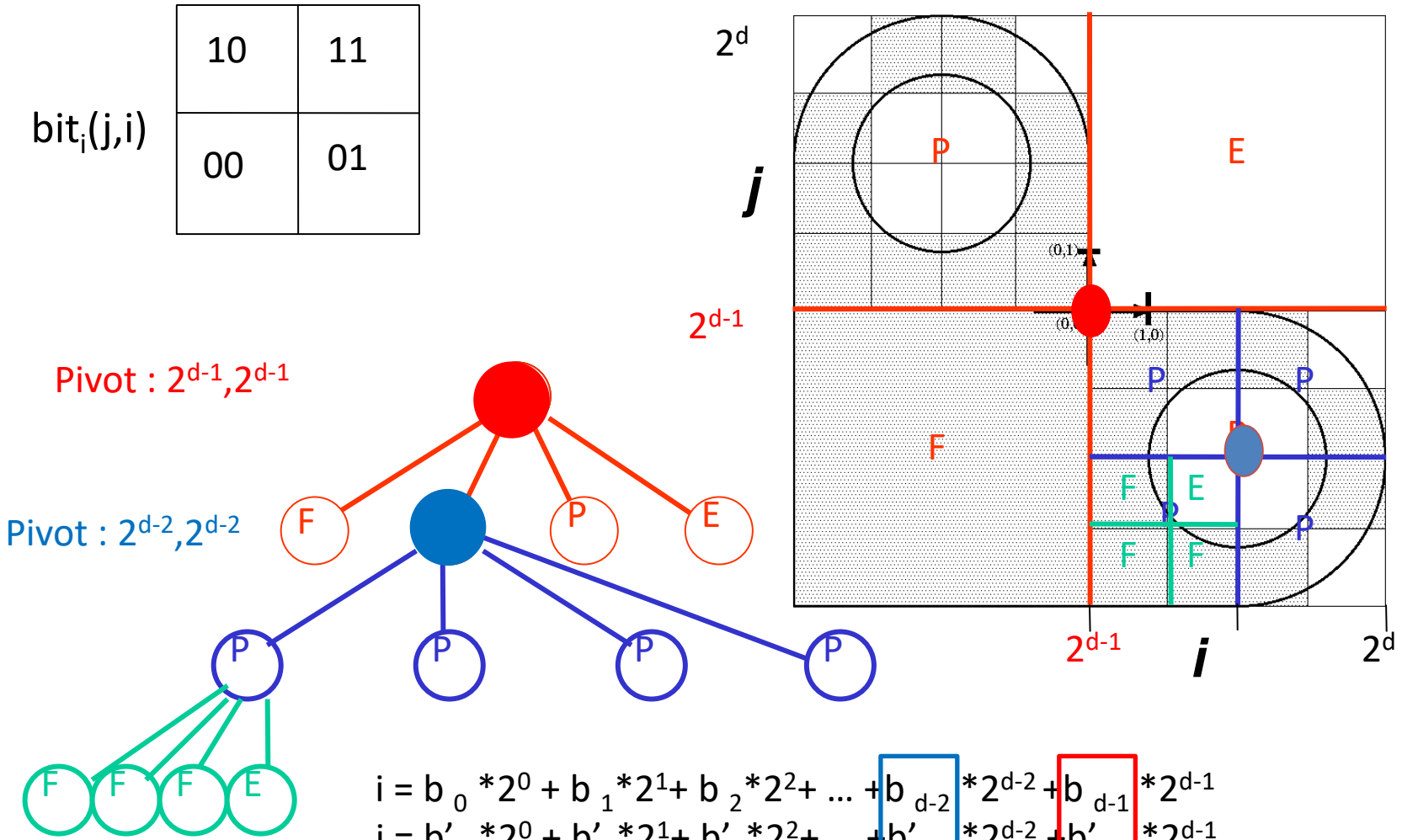


Codage : i ou $j=b_0 *2^0 + b_1*2^1+ b_2*2^2+ \ldots +b_{d-1} *2^{d-1}$ avec $b_k=1$ ou 0

points dans Quadtree

$bit_i(j,i)$

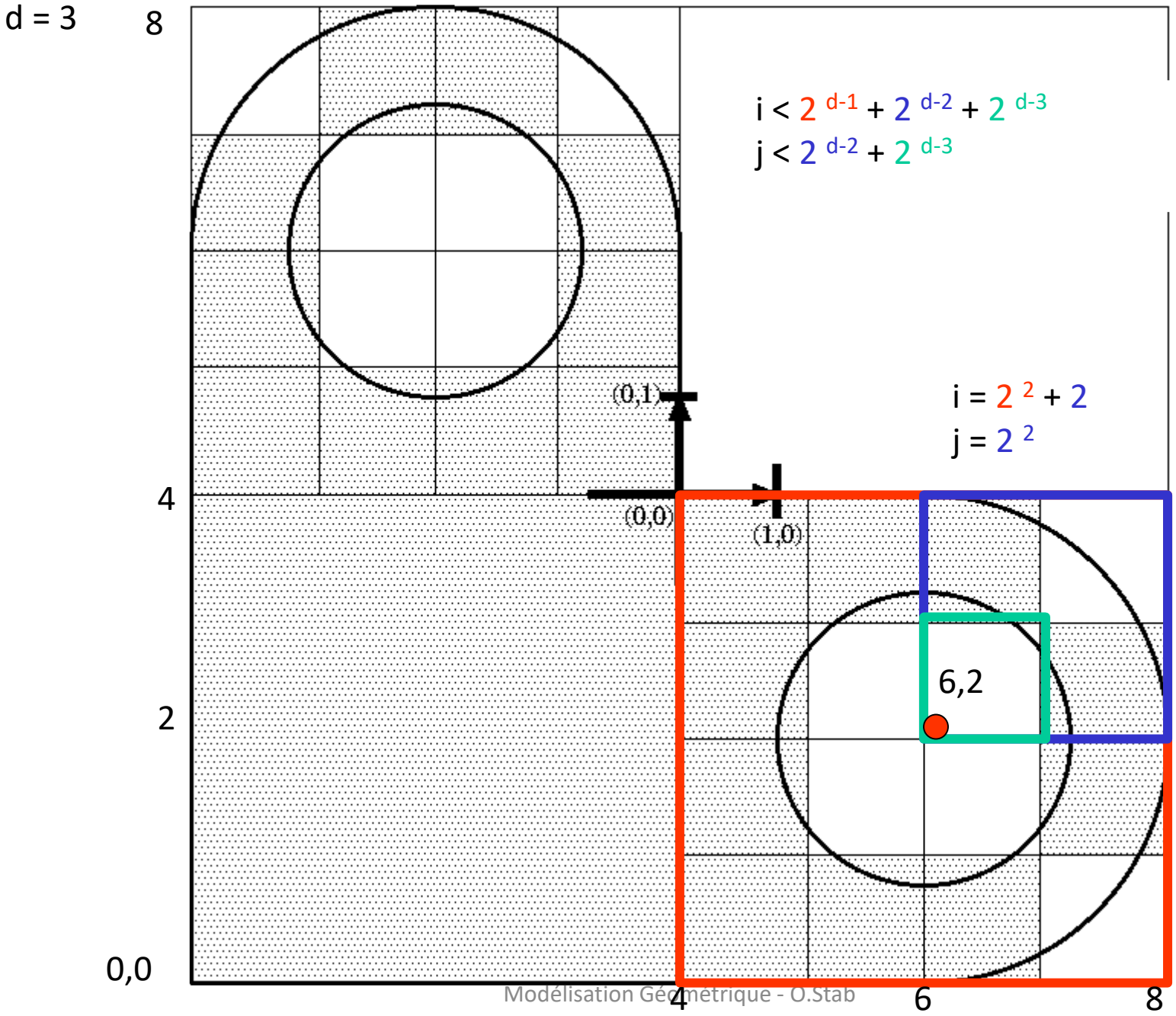
10	11
00	01



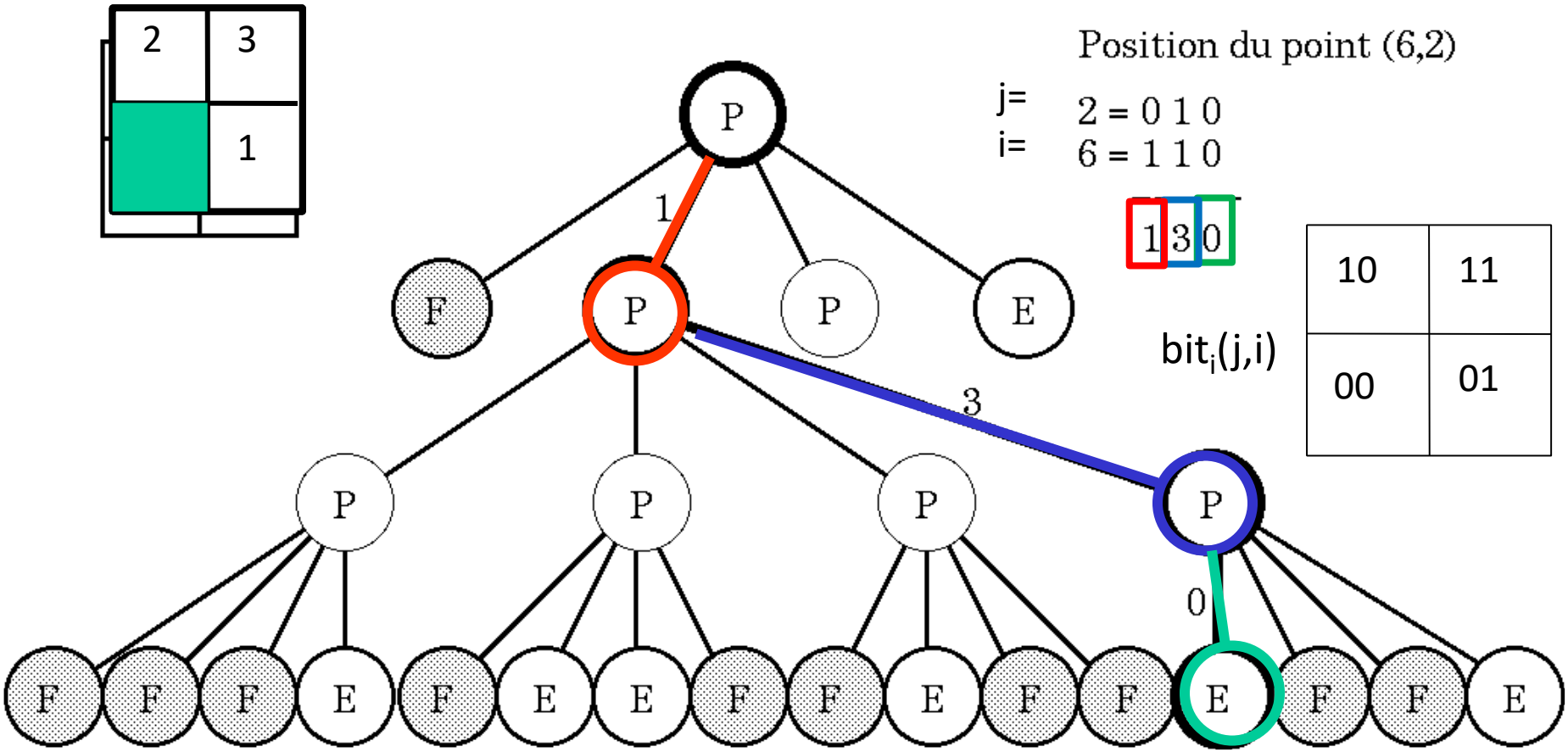
$i = b_0 *2^0 + b_1*2^1 + b_2*2^2 + \dots + b_{d-2}*2^{d-2} + b_{d-1}*2^{d-1}$

$j = b'_0 *2^0 + b'_1*2^1 + b'_2*2^2 + \dots + b'_{d-2}*2^{d-2} + b'_{d-1}*2^{d-1}$

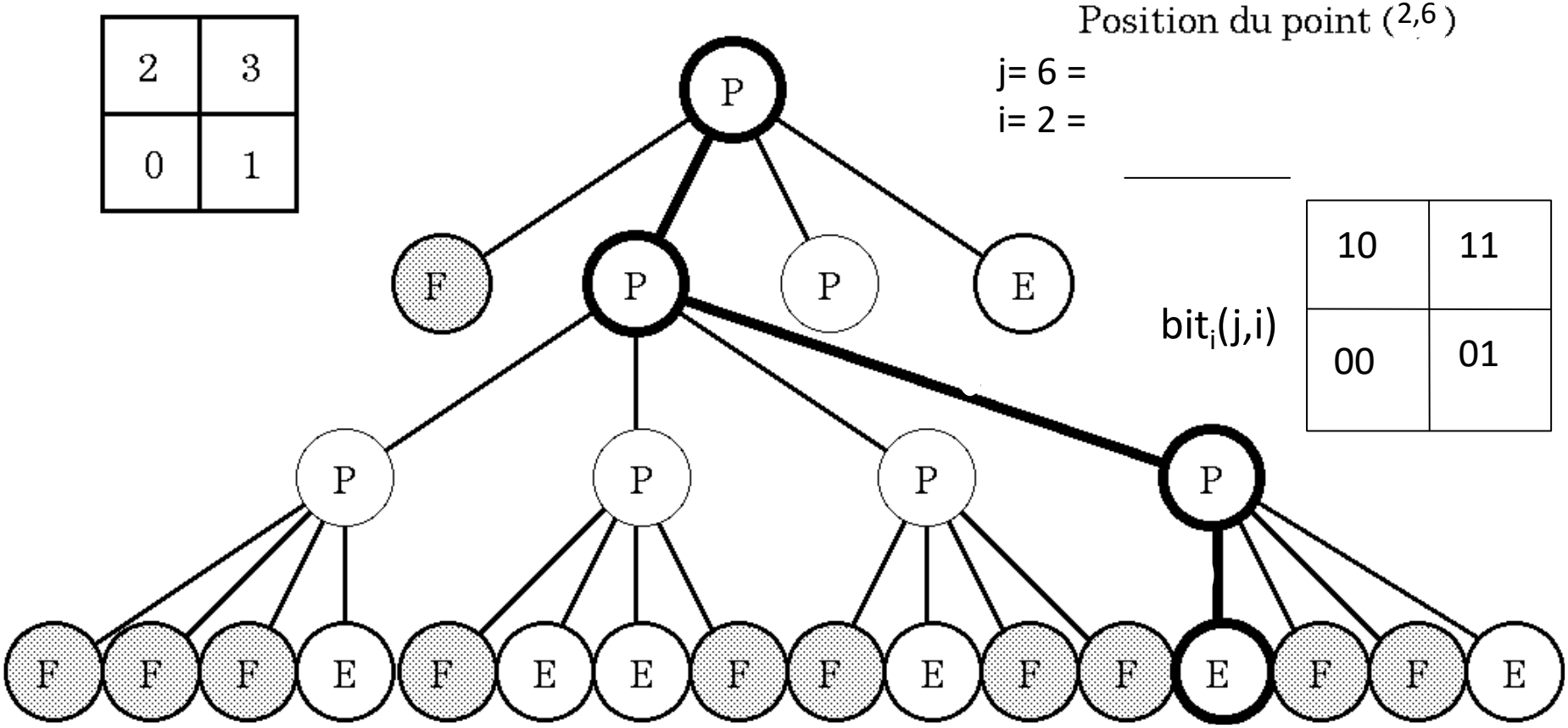
avec b_k=1 ou 0


d = 3
8
4
2
0,0
4
6
8
i < 2 d-1 + 2 d-2 + 2 d-3
j < 2 d-2 + 2 d-3
i = 2 2 + 2
j = 2 2
(0,1)
(0,0)
(1,0)
6,2

Localisation d ’un point



Localisation d ’un point exercice 1:



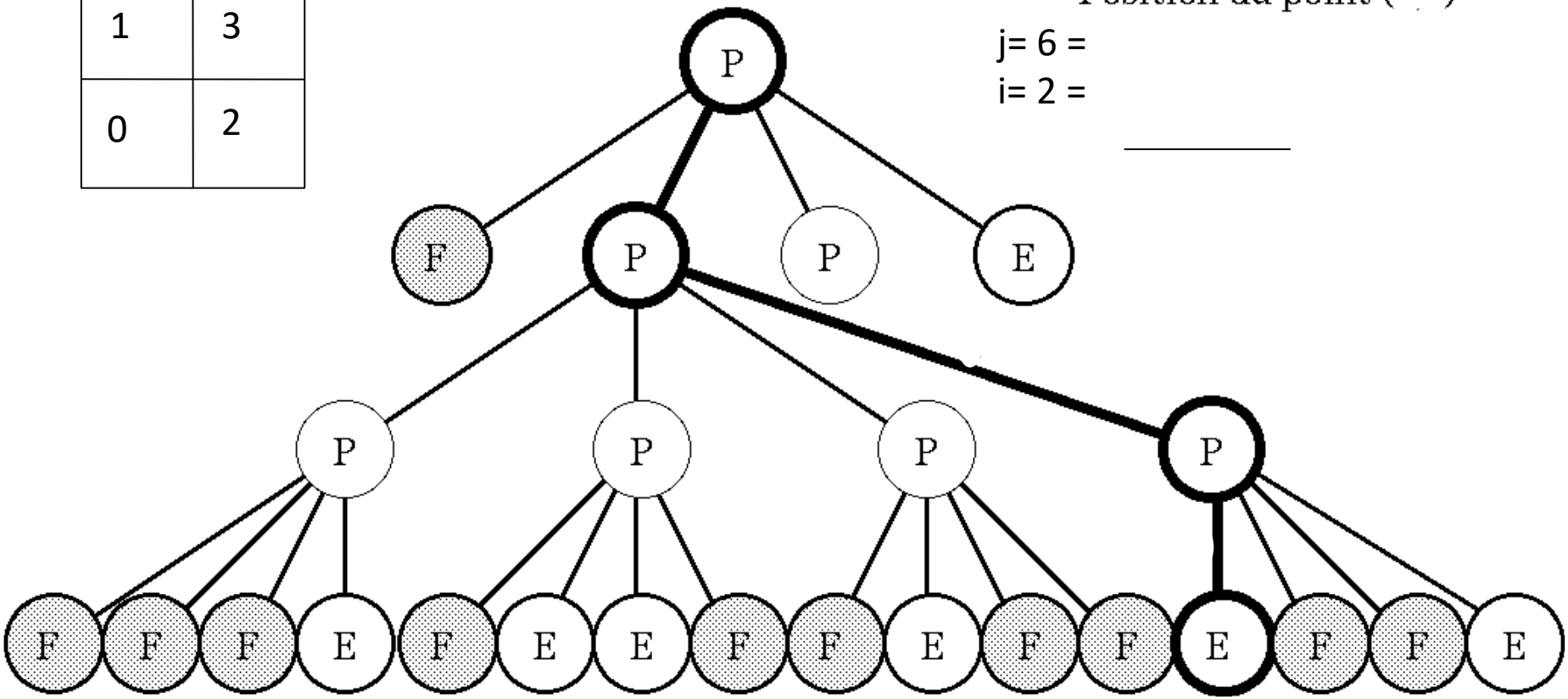
Localisation d ’un point exercice 2:

1	3
0	2

Position du point $(^{2,6})$

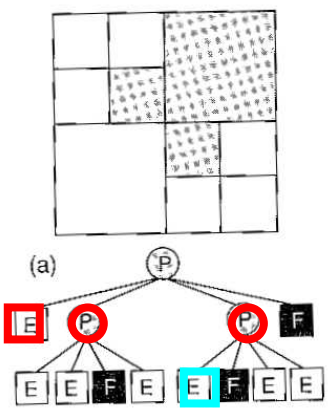
j= 6 =

i= 2 =

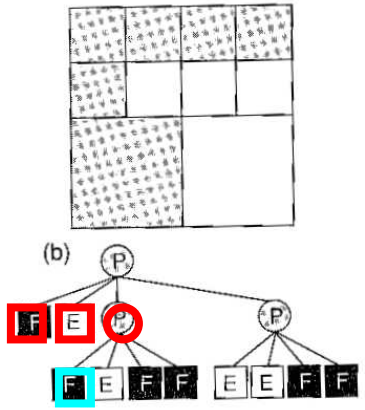


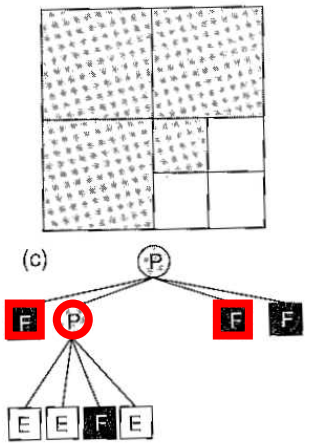
Opérations booléennes

A

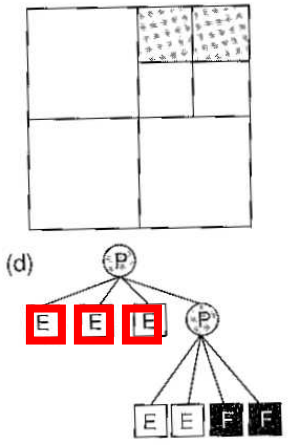


B





A u B

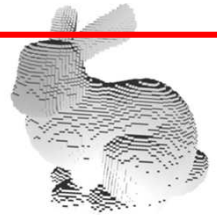


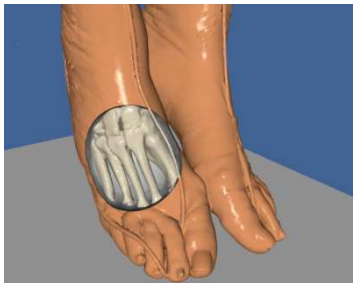
A n B

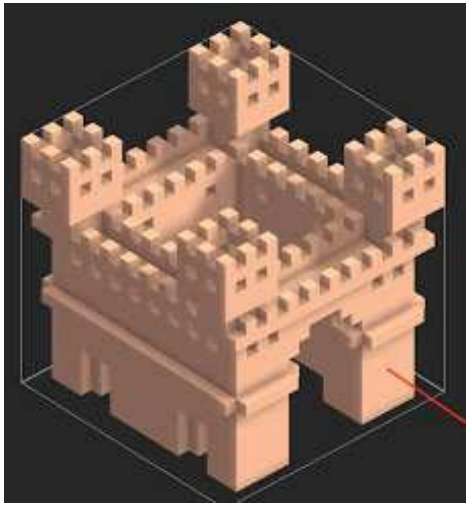
Propriétés du modèle

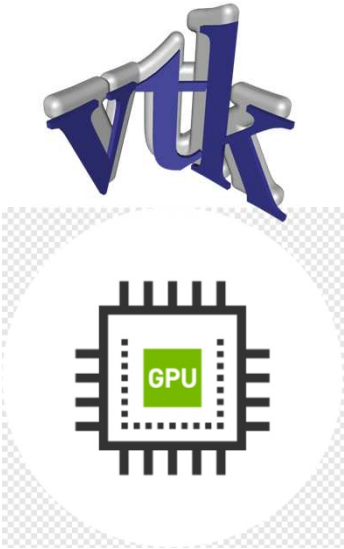
- Domaine
- Complétude
- Unicité
- Manipulation
- Performances

cuda, openCL





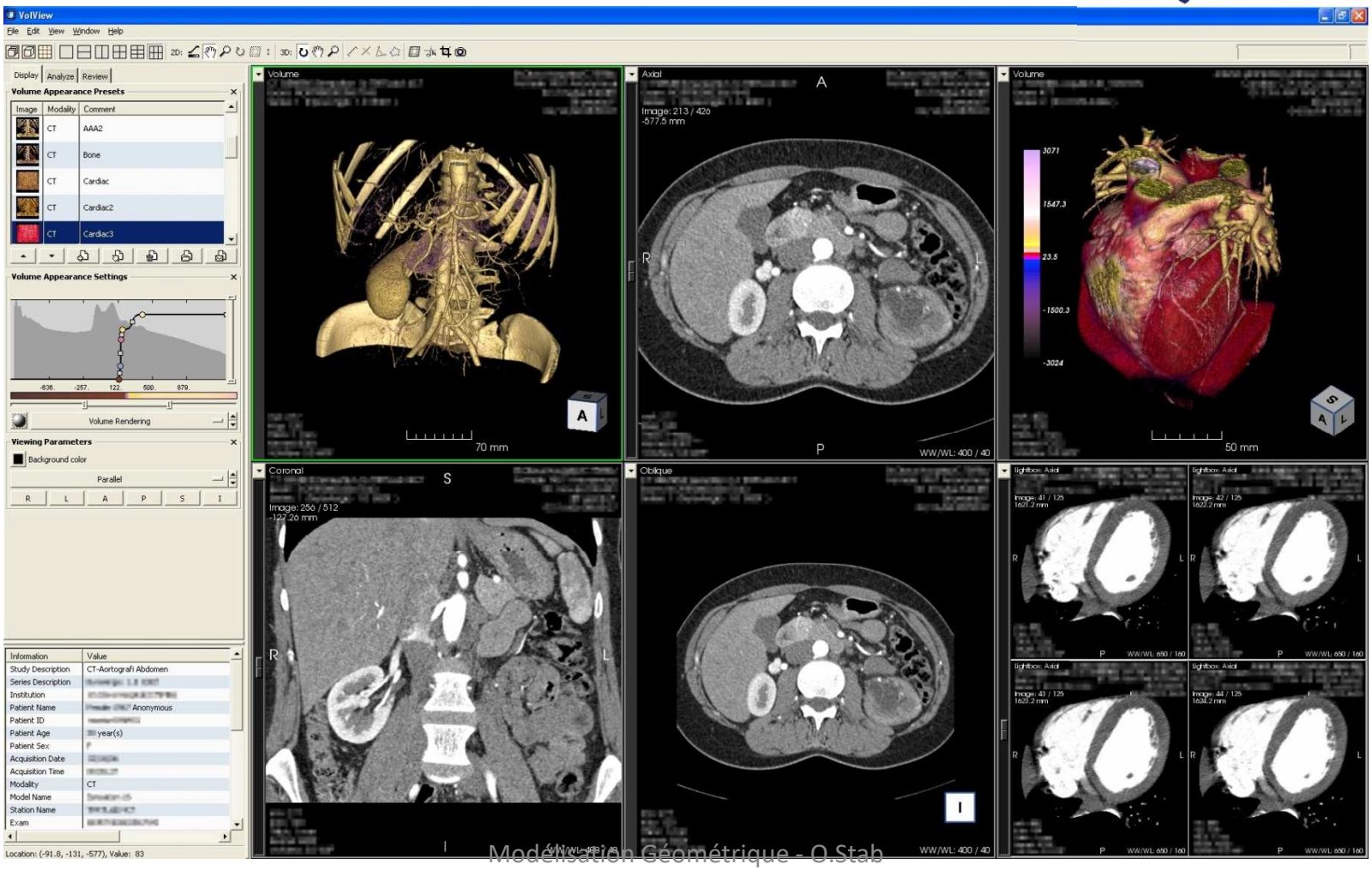




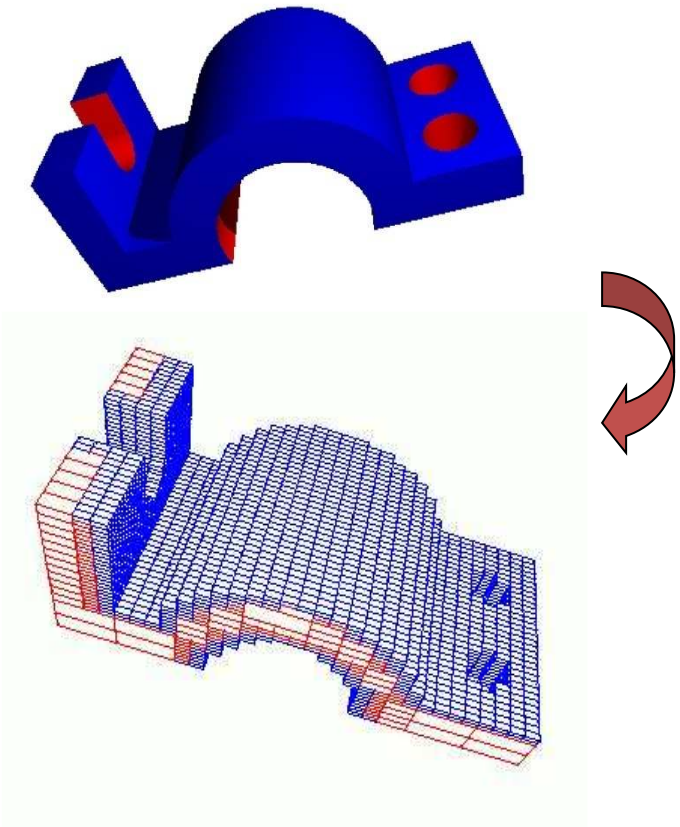
The Visualization ToolKit

www.vtk.org VTK (C++) open source, BSD

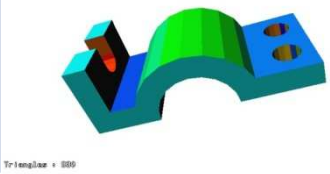
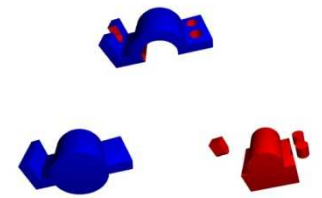
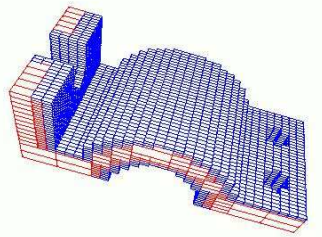
1993 Schroeder, Martin, Lorensen



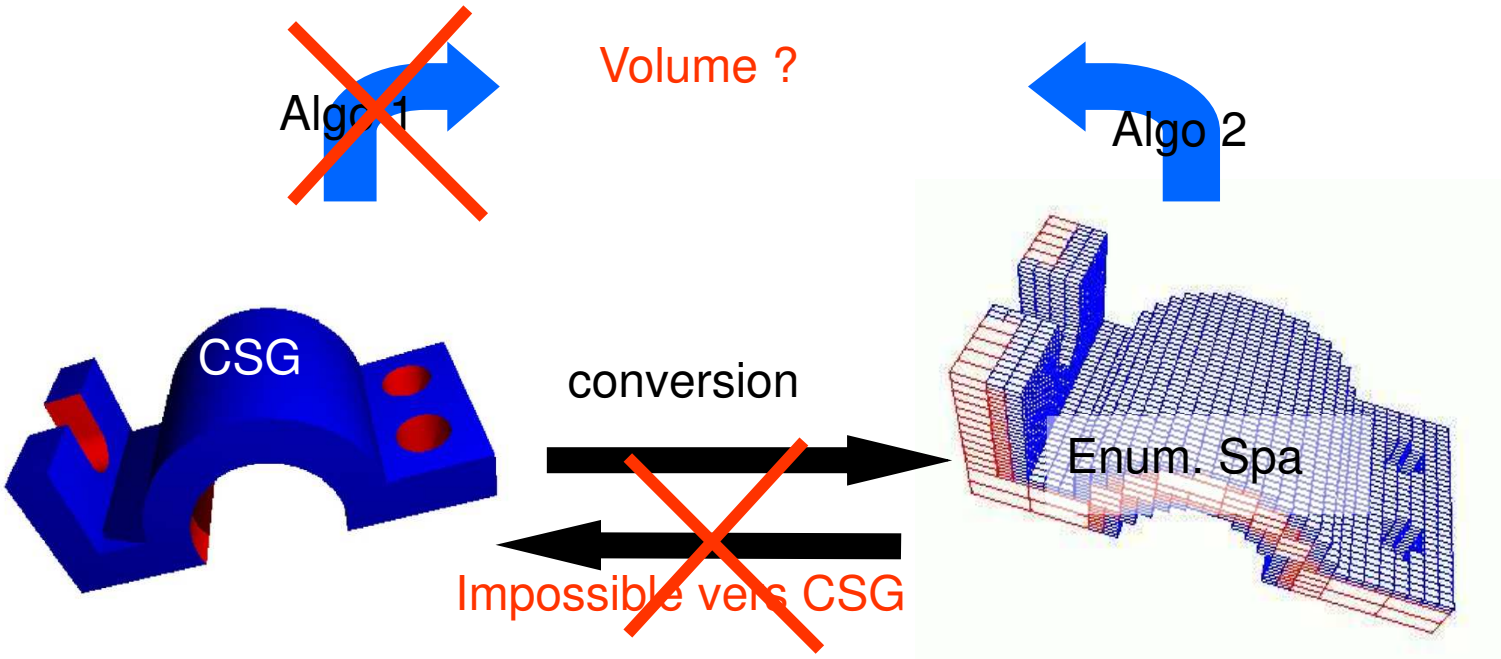
Conversion et comparaison des modèles



Comparaison des approches

	Domaine	Complétude	Interaction	Visual.
Brep	large	OK	difficile	
CSG	large	incomplet	naturelle	
Enum	large	précision	laborieuse	

Conversion des modèles



Hierarchie : CSG > Brep > grille, octree...

Algorithmes de conversion

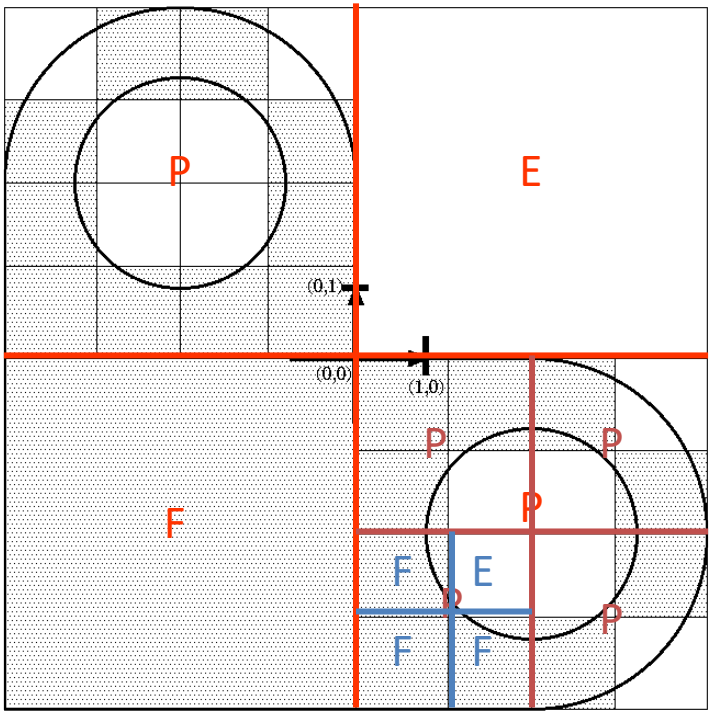
1. Modèle d ’énumération spatiale
    1. à partir d ’un autre modèle
    2. à partir d ’une Brep

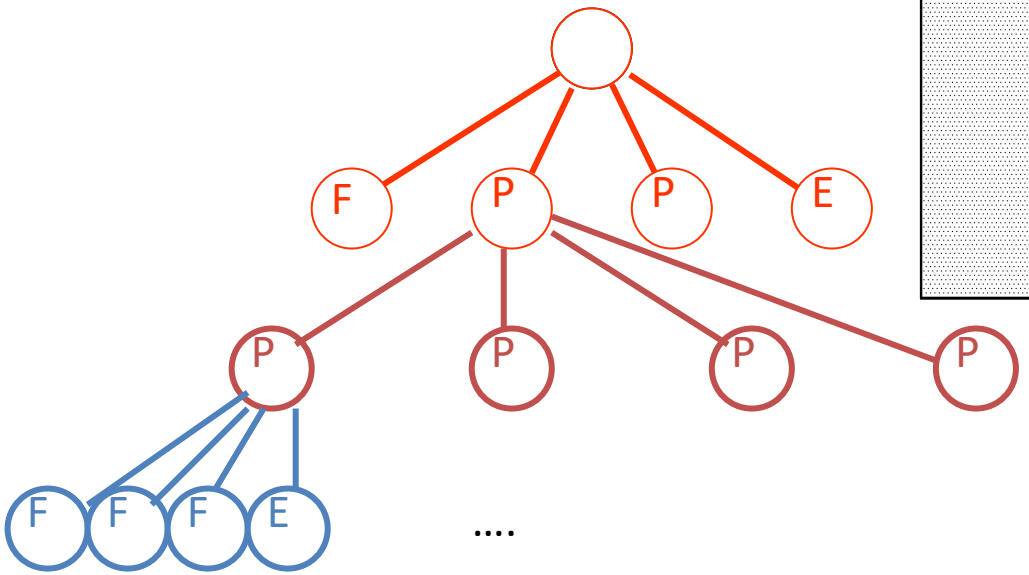
2. Représentation frontière
    1. à partir d ’un CSG
    2. à partir d ’une grille
    3. À partir d’un « champ de scalaires »

1.1.Construction d ’un Quadtree

Classification des cellules

- P : Partiel
- E : Empty
- F : Full





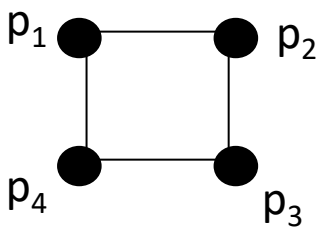
....

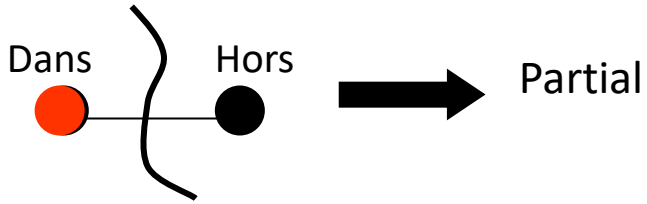
Quelque soit le modèle M on dispose de la fonction : PointDans(pt,M)

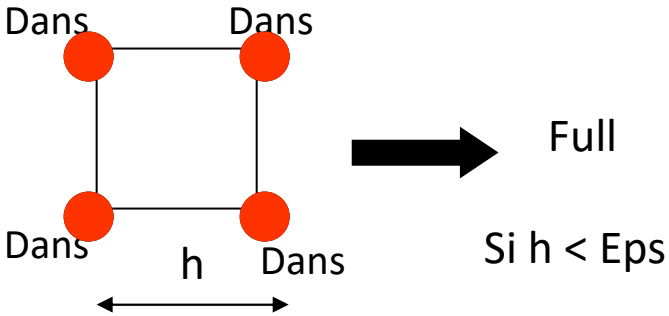
Classification des cellules

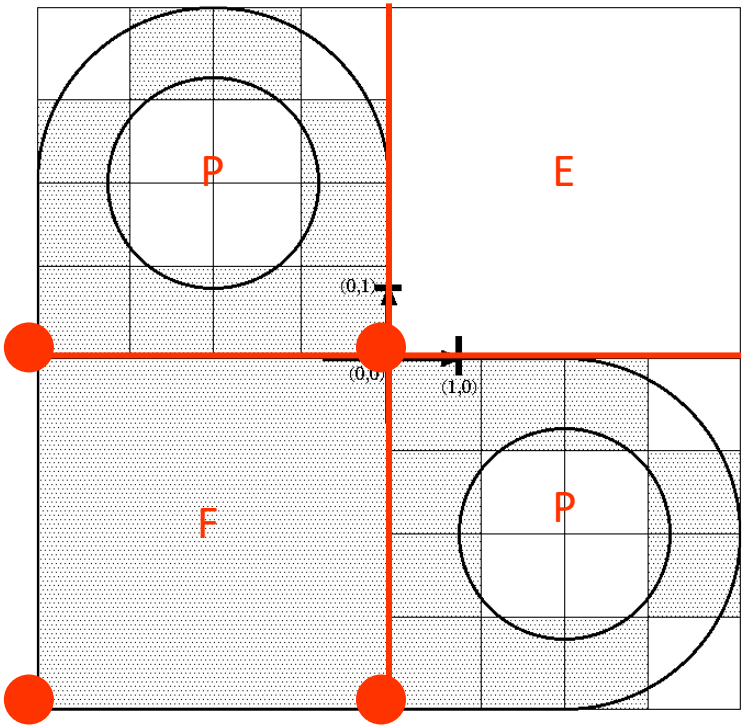
Utilisation de PointDans(pt,M)

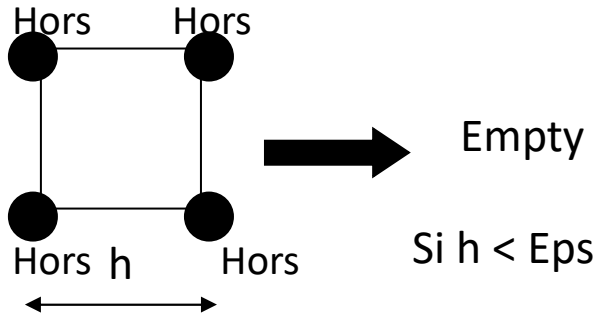
Localisation de points de la cellule : Dans, Hors







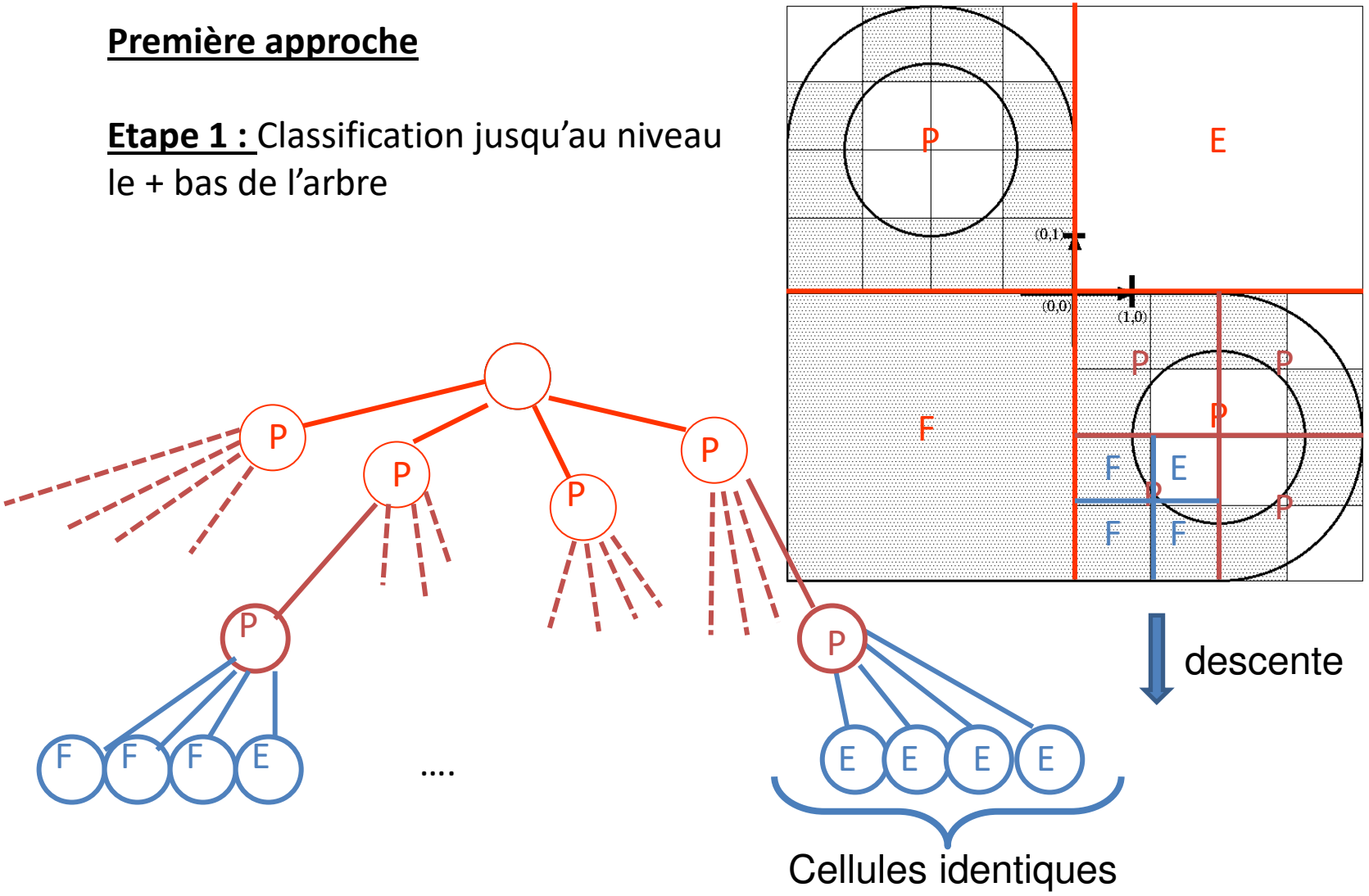




Approche 1: Construction du Quadtree « complet »

Première approche

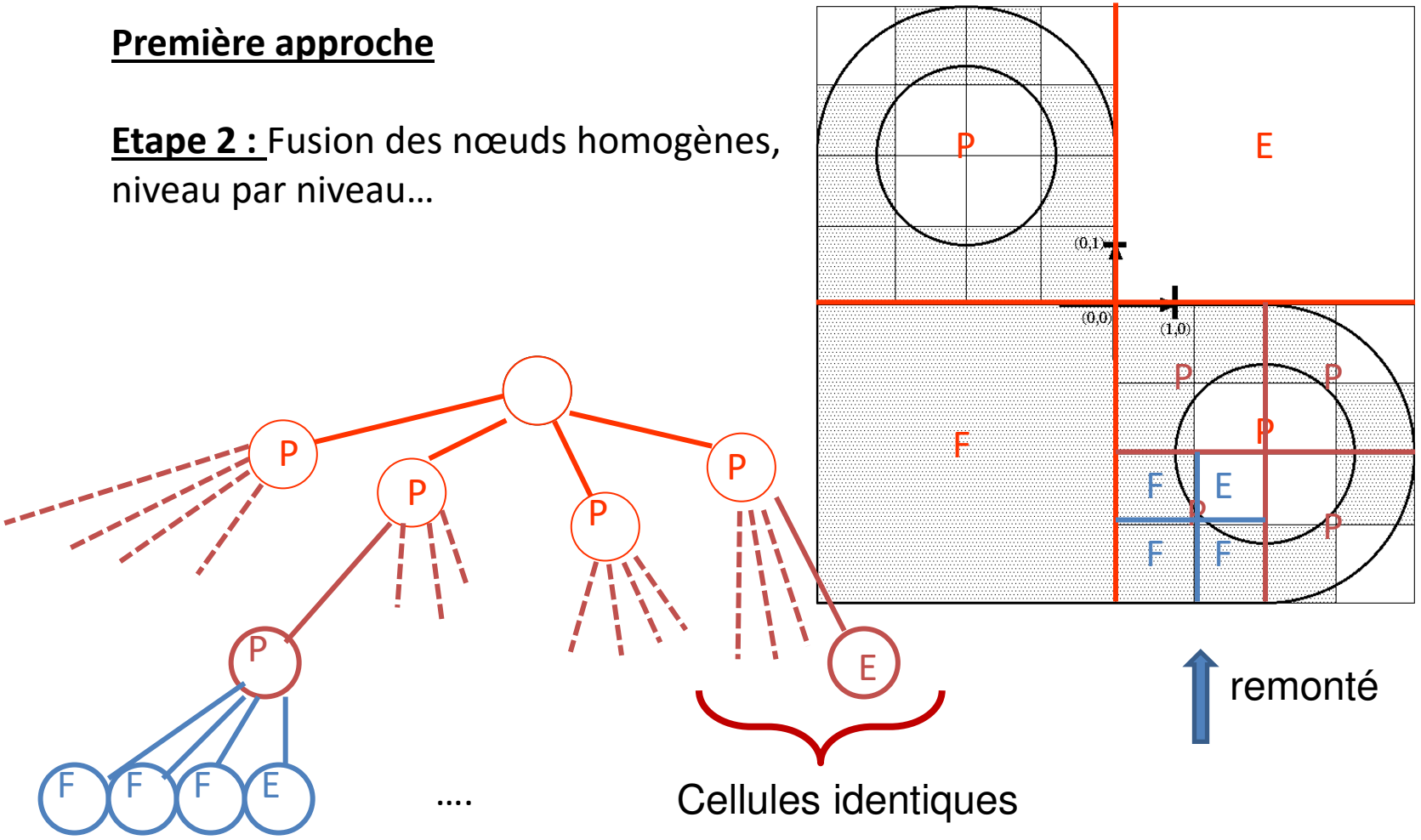
Etape 1 : Classification jusqu'au niveau le + bas de l'arbre



Approche 1: Réduction du Quadtree

Première approche

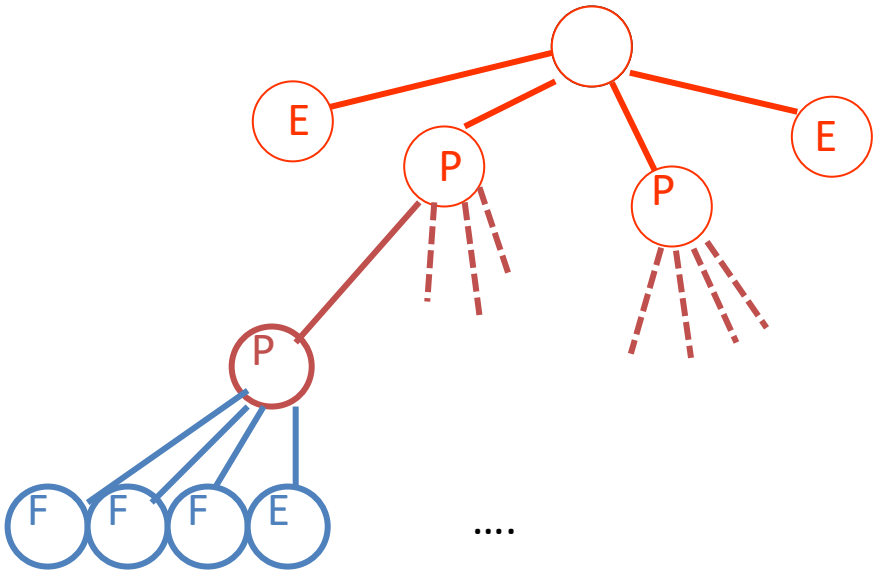
Etape 2 : Fusion des nœuds homogènes, niveau par niveau…

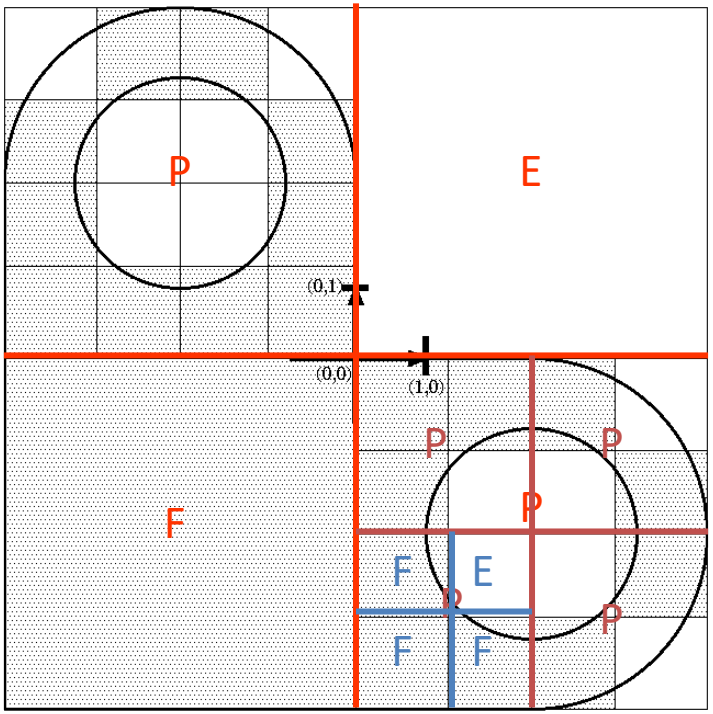


Approche 1: Réduction du Quadtree

Première approche

… Jusqu'à la fin.

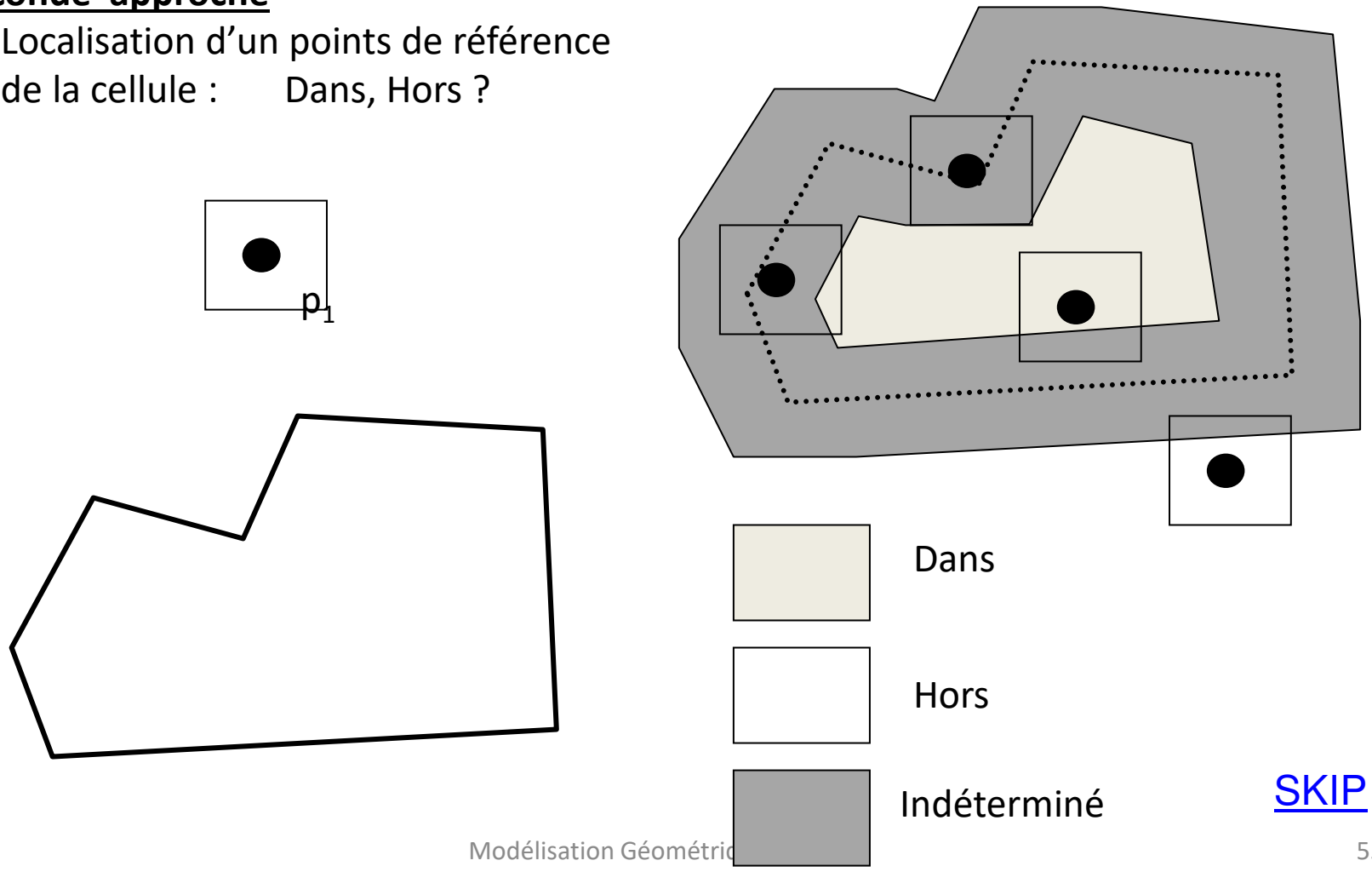




Approche 2 : Dilatation de la géométrie

Seconde approche

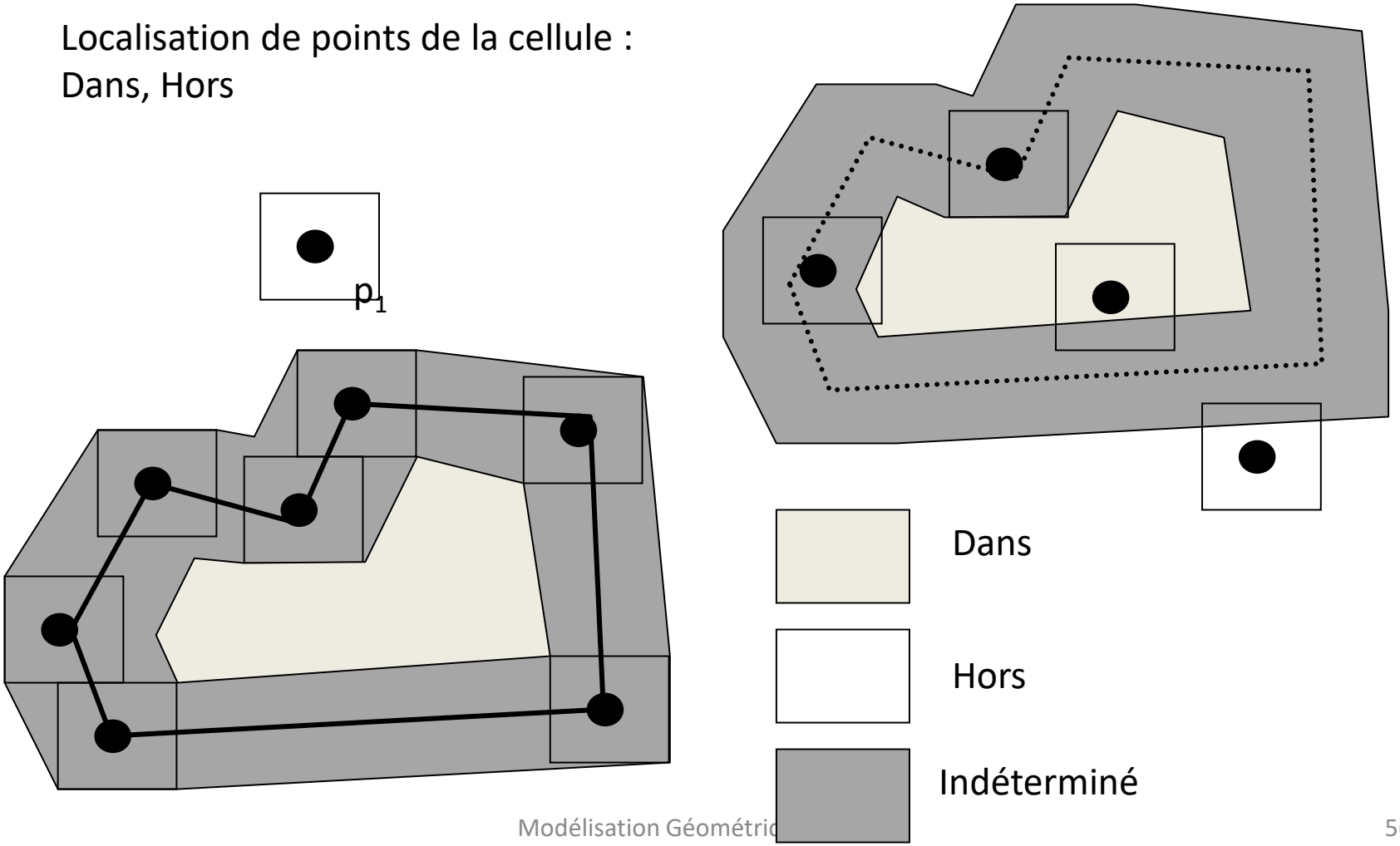
Localisation d'un points de référence de la cellule : Dans, Hors ?



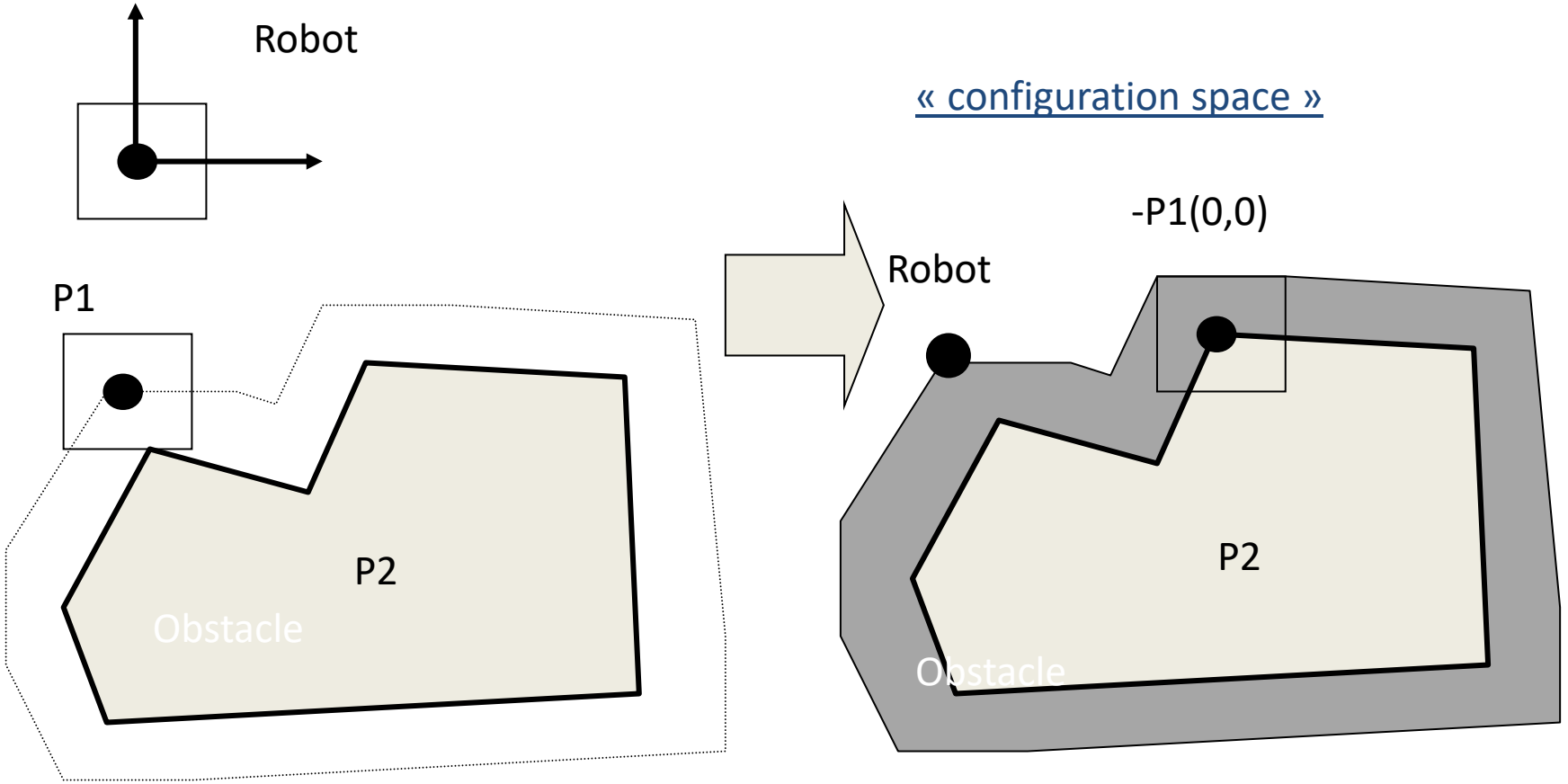
SKIP

Approche 2: Dilatation de la géométrie

Localisation de points de la cellule :
Dans, Hors



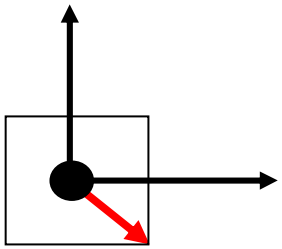
Analogie : évitement d'obstacles

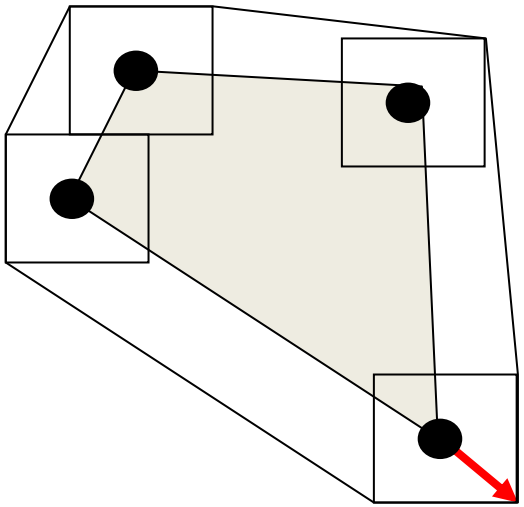


FS(P2,P1) = P2 ***+*** -P1(0,0)

Sommes de Minkowski

Polygones convexes

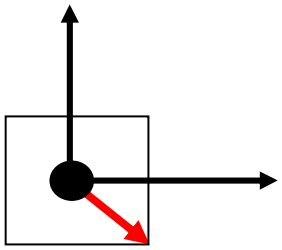




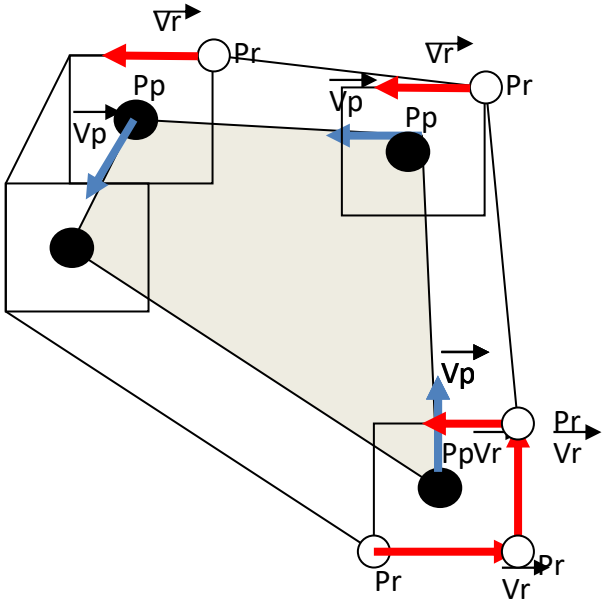
Les sommets de P1+P2 = sommets de P2 + VSi

Sommes de Minkowski

Polygones convexes



VSi



$\overrightarrow{Vr}$= Pr SUIV[Pr]

$\overrightarrow{Vp}$=Pp SUIV[Pp]

Si(PrV(Vr, Vp)> 0)Alors

Pr = SUIV[Pr]

FS = FS u { Pp + Pr }

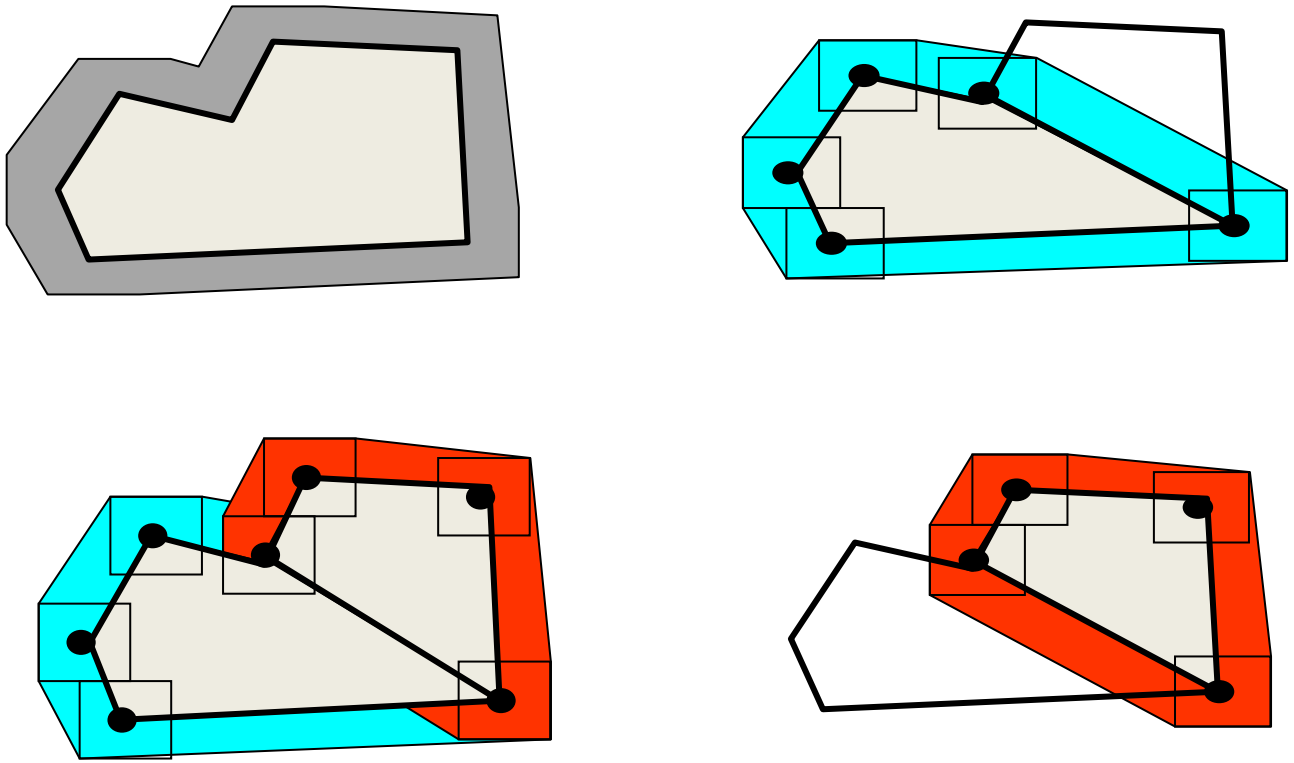
Sinon Pp = SUIV[Pp]

FS = FS u { Pp + Pr }

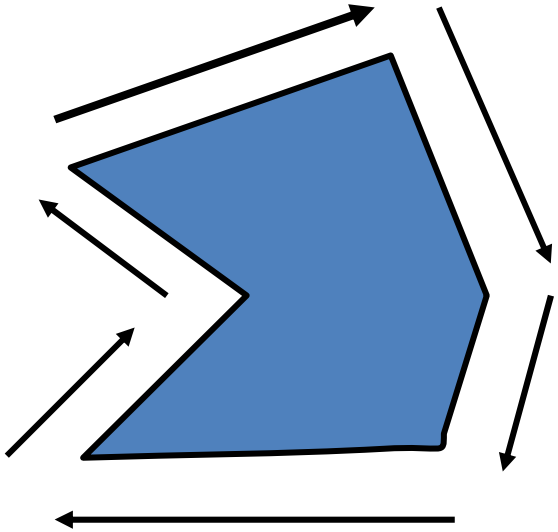
Les sommets du polygone P1+P2 = sommets de P2 + VSi

Sommes de Minkowski

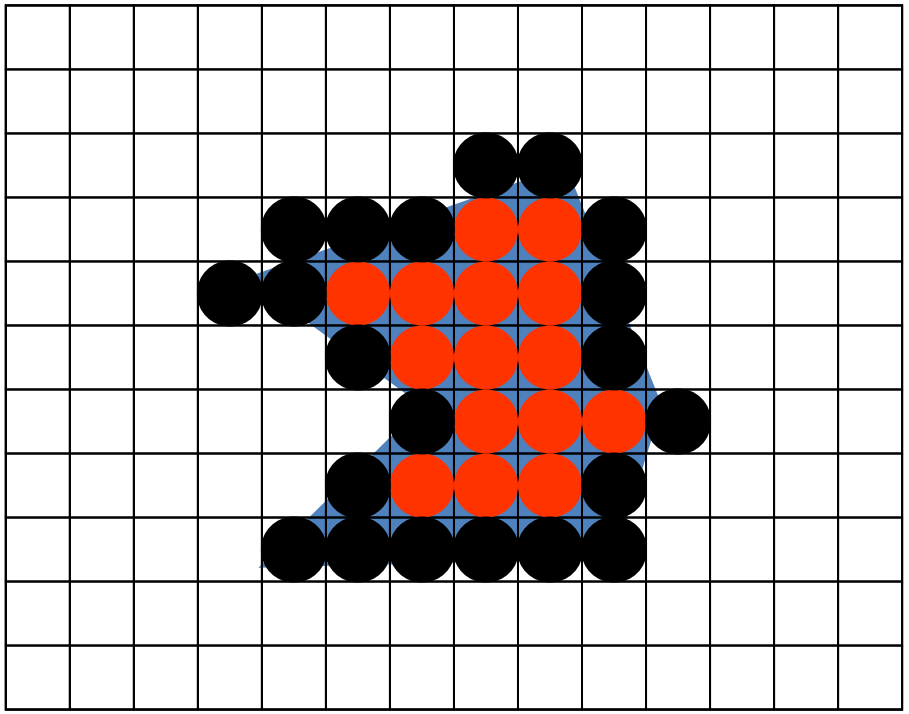
Polygones non convexes



1.2.Enumération à partir d ’une Brep



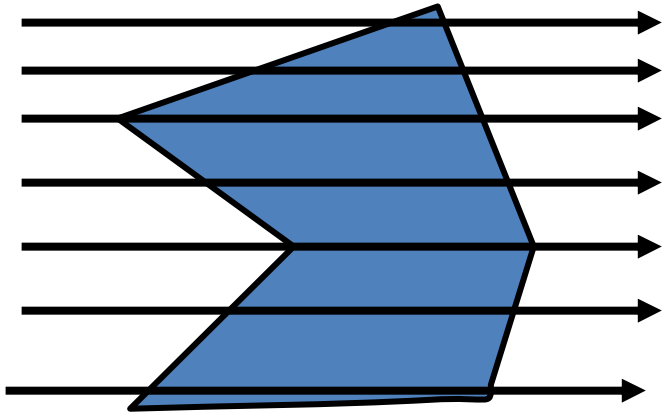
Algorithme de tracé : mid-point algorithme



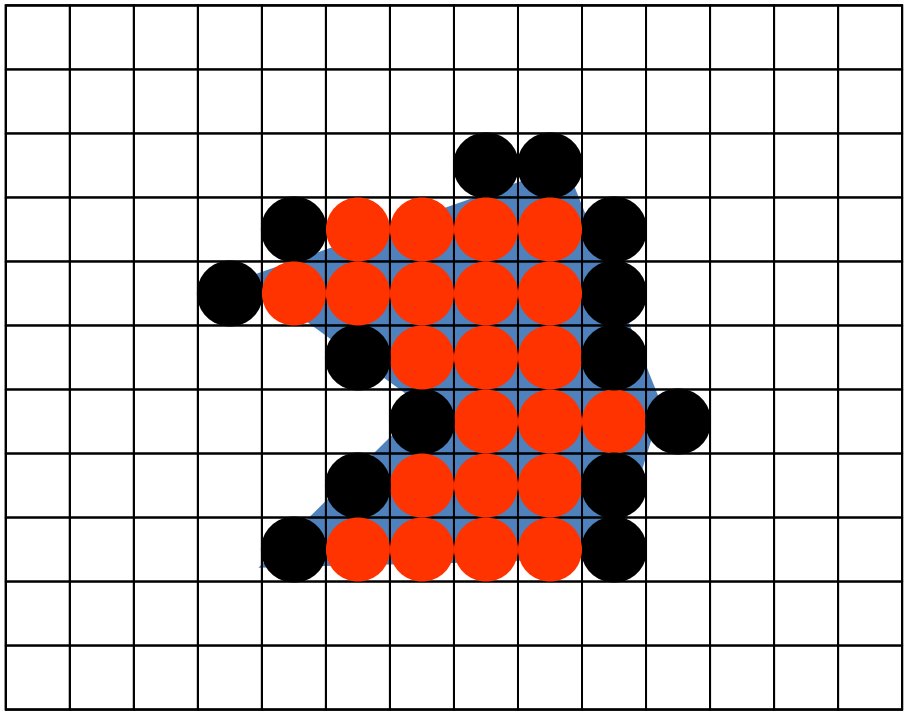
Remplissage : voisinage 4-connexité

Enumération à partir d ’une Brep (2)

Scan-line Algorithm



Edge coherence

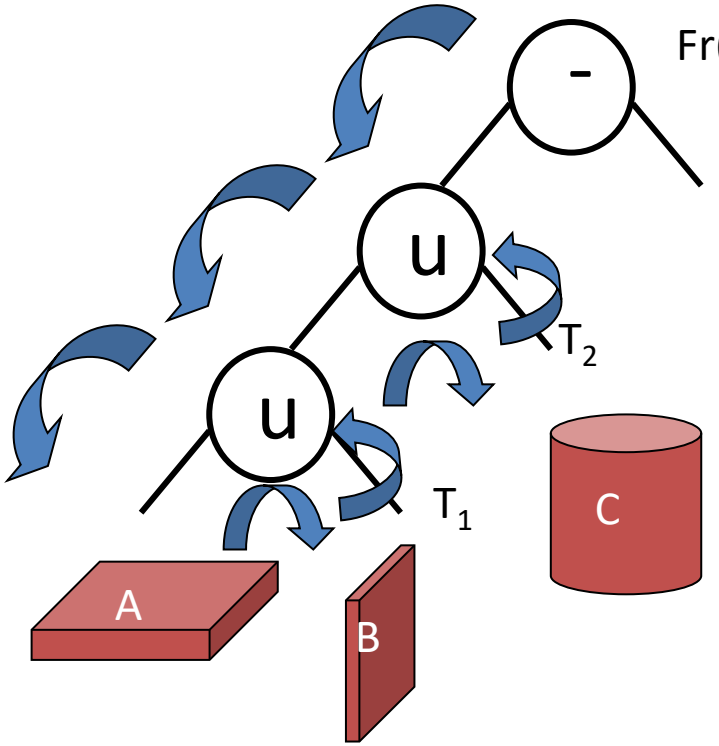


Algorithmes de conversion

- Modèle d ’énumération spatiale
  - à partir d ’un autre modèle
  - à partir d ’une Brep

- Représentation frontière
  - à partir d ’un CSG
  - à partir d ’une grille
  - À partir d’un « champ de scalaires »

2.1.Calcul de « la » Brep d ’un CSG



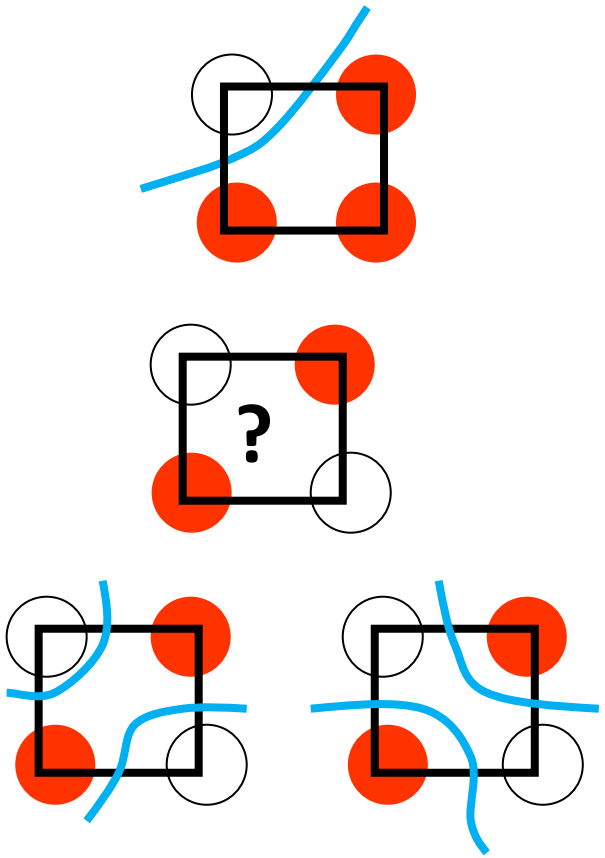
$Fr(O) = (Fr(A)u(T_1Fr(B)))u(T_2Fr(C)) - ...$

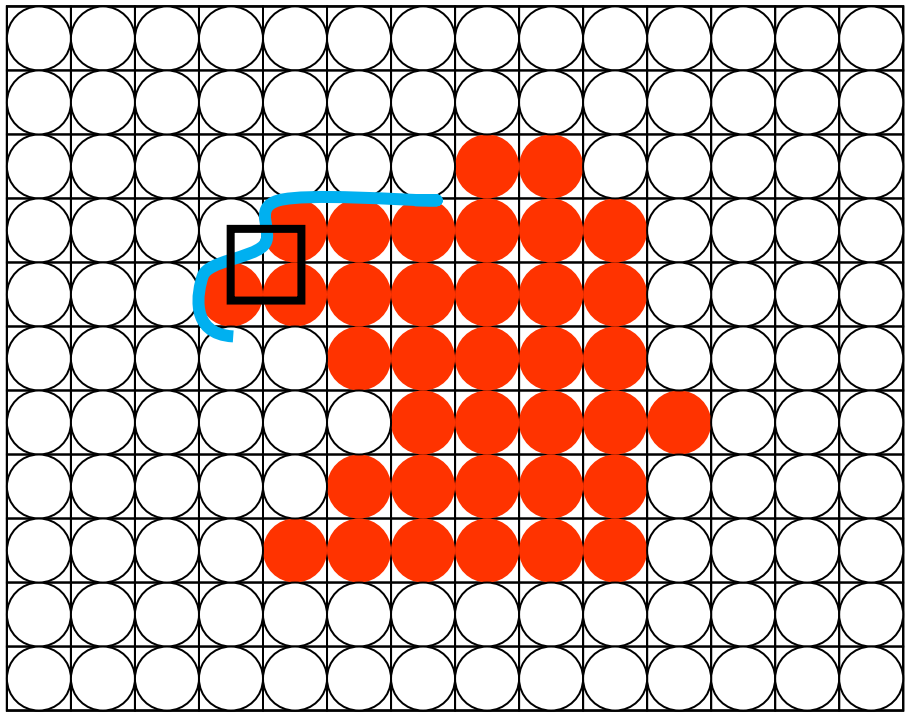
```
Union {
    union {
      box { <0,0,0> <107,48,11> }
      box { <0,0,0> <11,48,25> } translate <96,0,11>
      }
     cylinder { <0,0,0> <0,48,0> 30 translate <53.5,0,5.5> }
}
```

$Fr(A)$ u_{brep} $T_1(Fr(B))$ u_{brep} $T_2(Fr(C))$

2.2. Le Marching Cube (MC)

Extraction d ’une Brep à partir d ’une image (~ ligne iso-valeur)

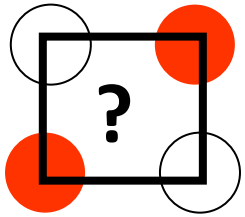




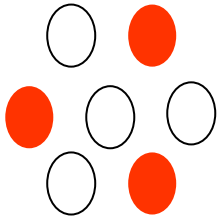
3 « solutions » empiriques :

- Privilégier le vide ou le plein
- Limiter la courbure ➔ analyse cellules voisines
- Accéder aux valeurs non seuillées

Un problème topologique

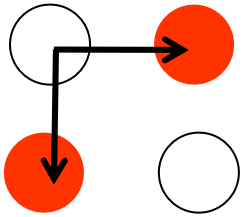


6-connexité

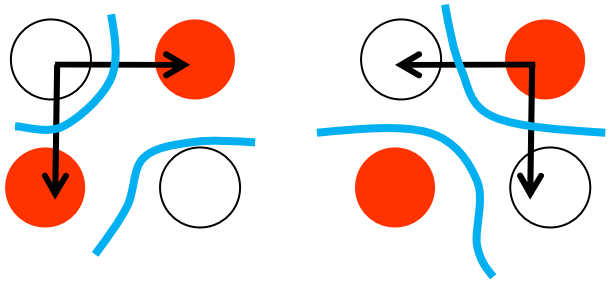


Voisinage :

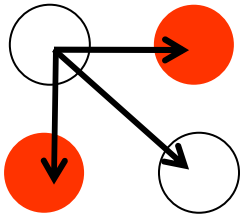
– 4-connexité

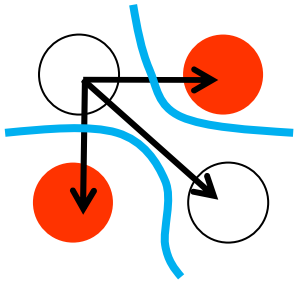


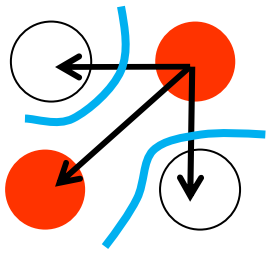
Fr(A) != Fr(Compl(A))



– 8-connexité

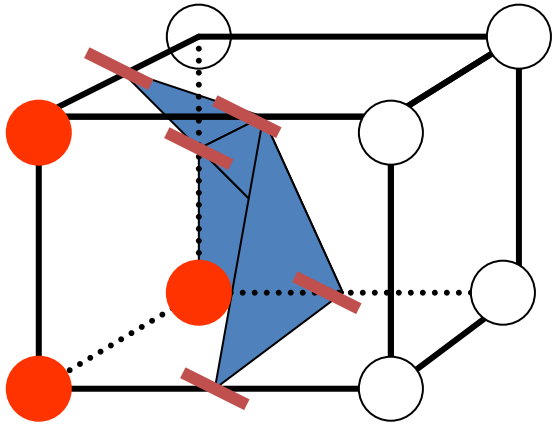


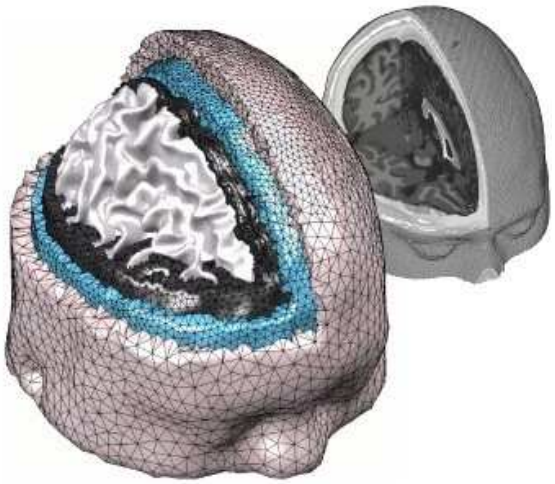




Le Marching Cube (MC)

Extraction d ’une Brep à partir d ’un voxet (ou d’une série d’images)



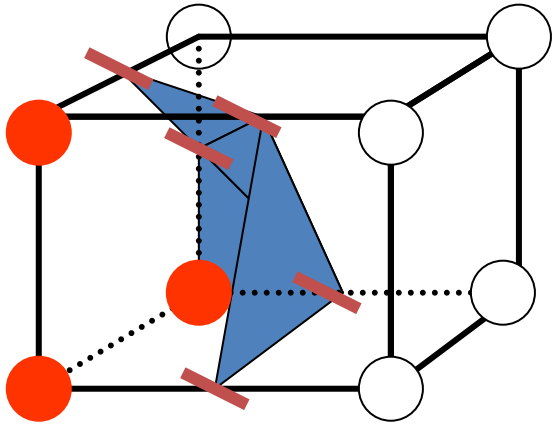


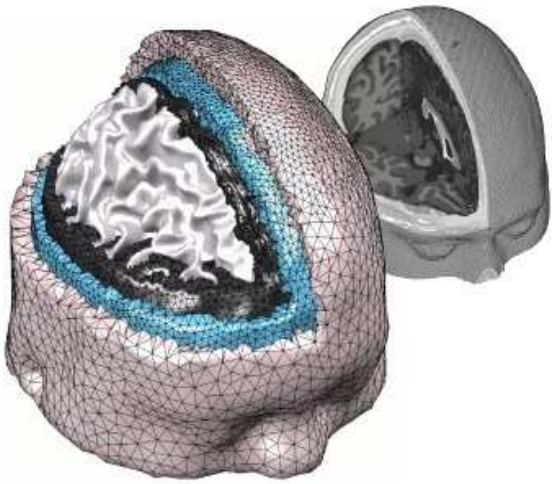
Symmetry exploited	
None	$2^8 = 256$
Reflection	128
Rotation	23
Rotation + mirror	22
Rotation + reflection	15
Rotation + reflection + mirror	14

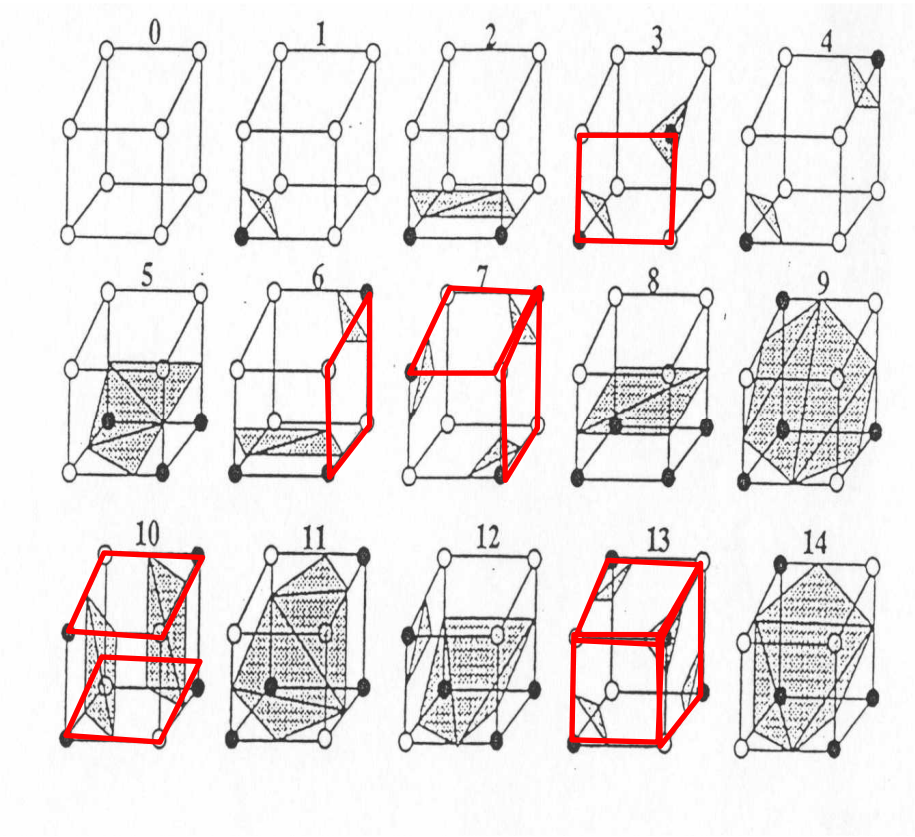
Marching cubes algorithms
Lorenson & Cline 1987

Le Marching Cube (MC)

Extraction d ’une Brep à partir d ’un voxet (ou d’une série d’images)







Marching cubes algorithms
Lorenson & Cline 1987

Marching Cube (MC)

A survey of marching cubes algorithms
T.S. Newman & H. Yi Computers & Graphics 30 (2006) 854-879

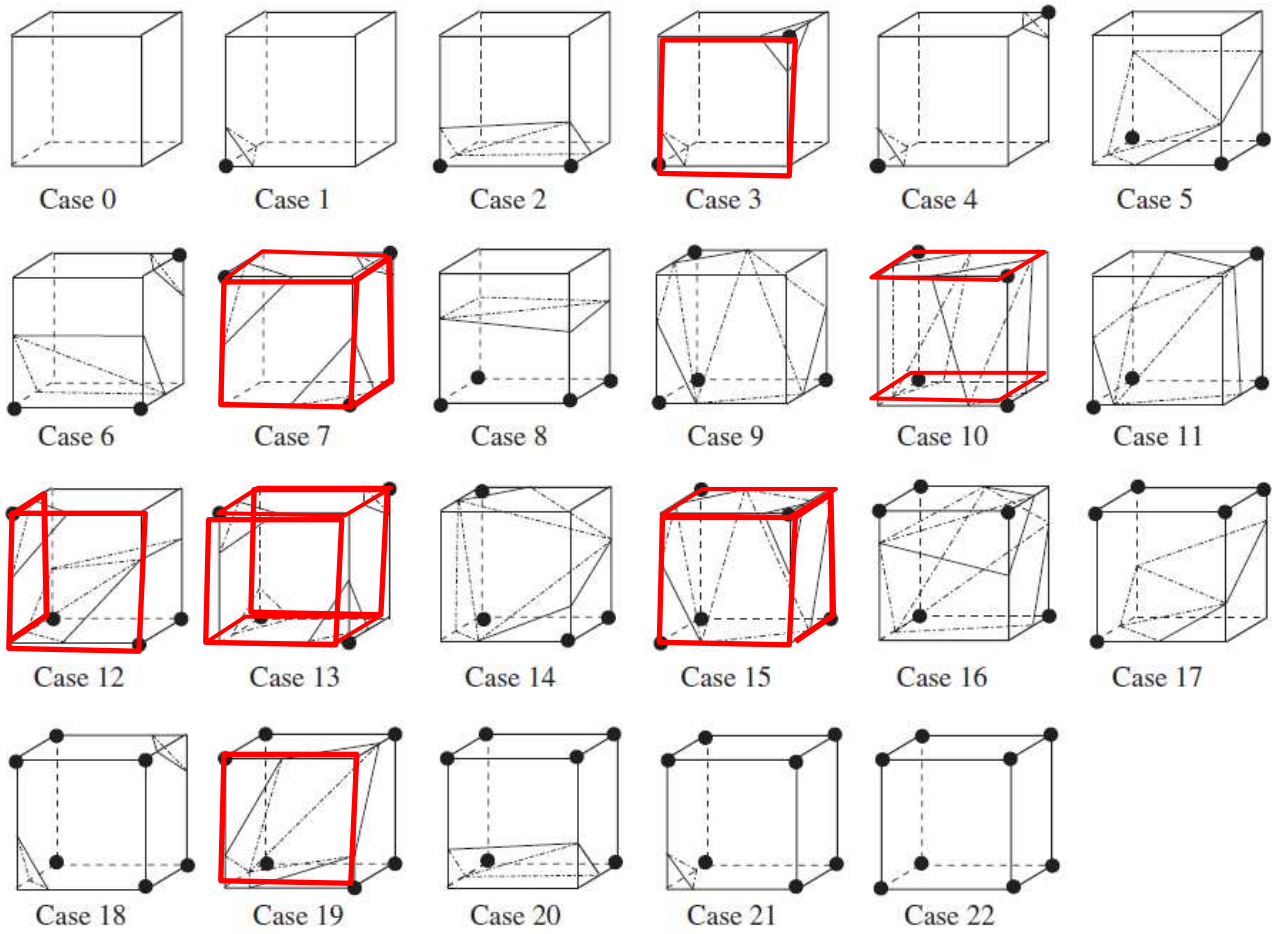
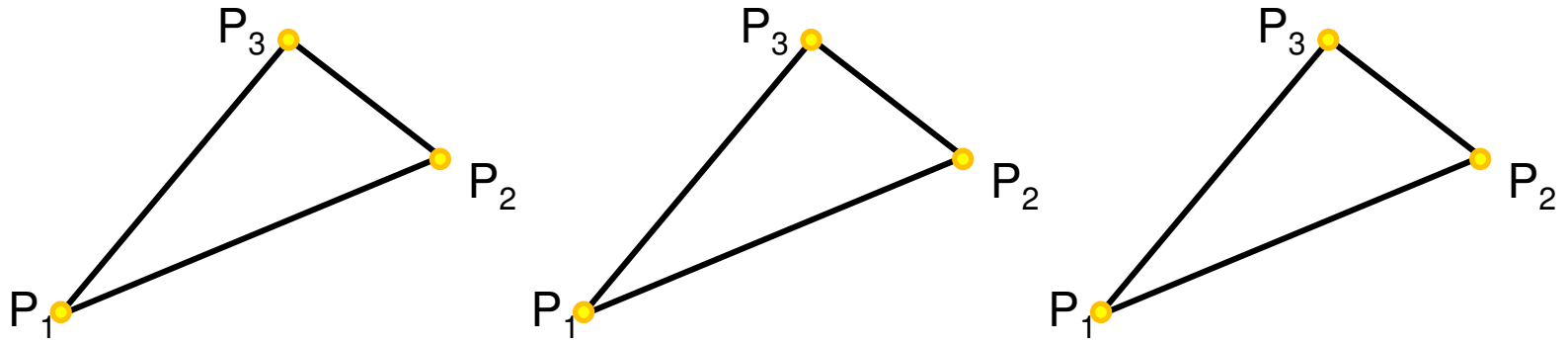


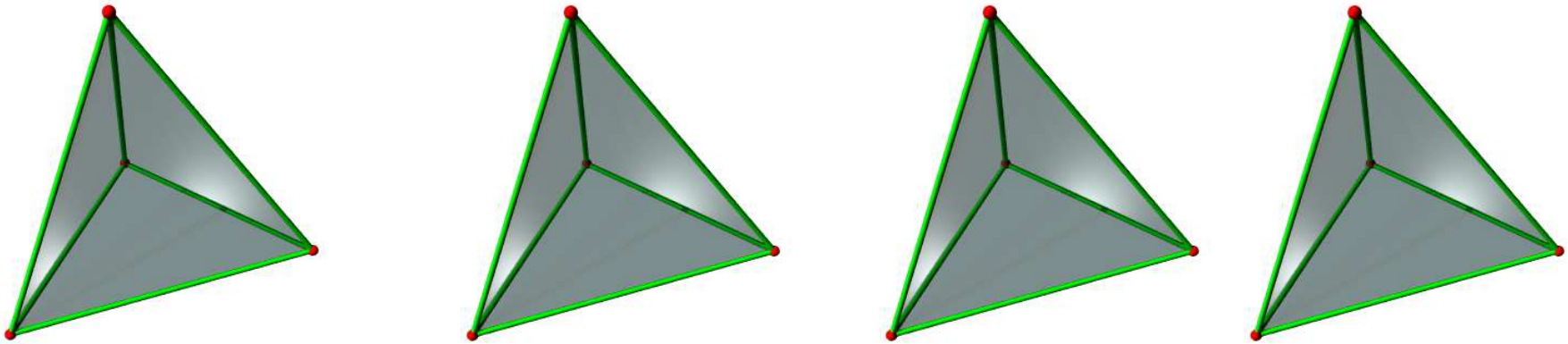
Fig. 6. The 23 intersection topologies (using the numbering of [27–29]) that result when only rotation is exploited.

Exercice : Marching Tetrahedra

- Extraction d'une iso-courbe sur une triangulation

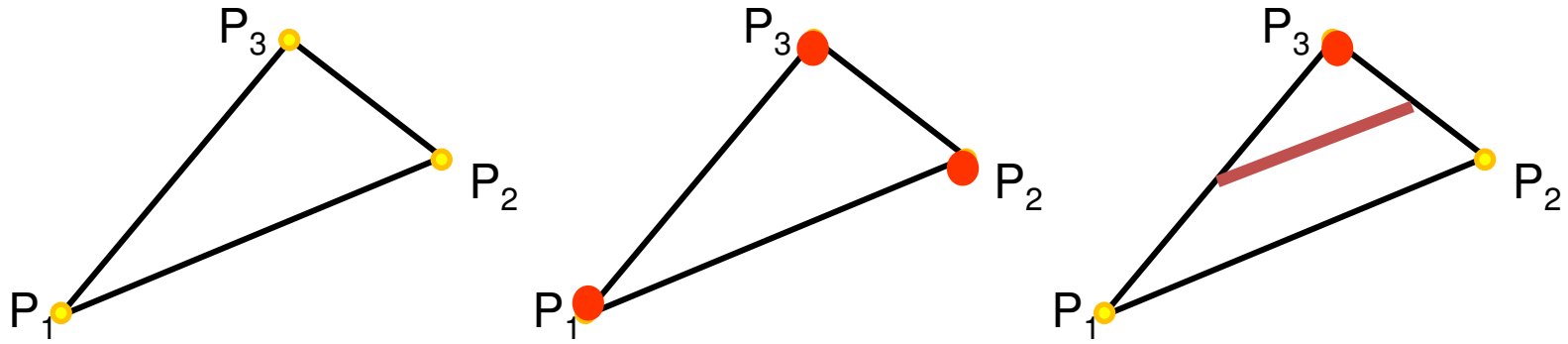


- Extraction d'une iso-surface dans une tetraédrisation

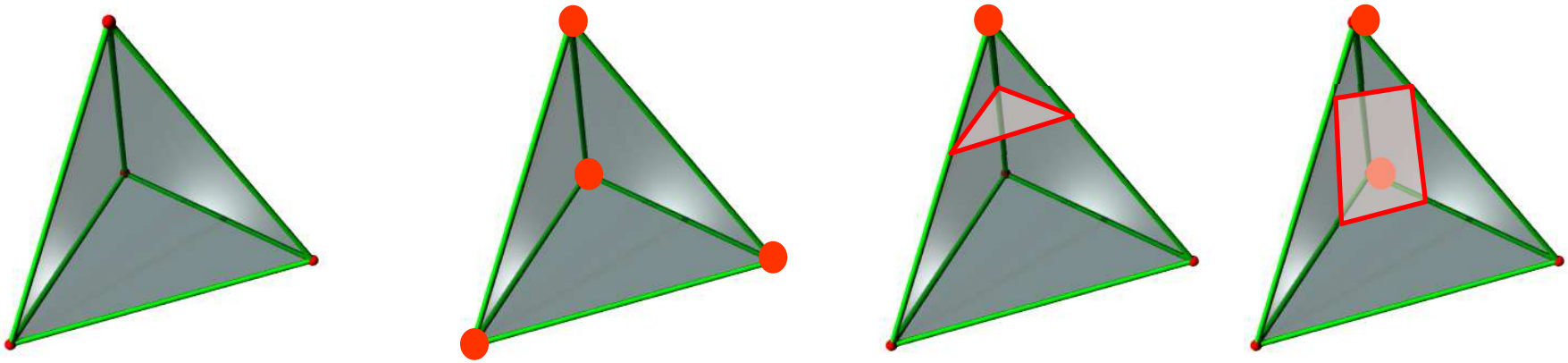


Exercice : Marching Tetrahedra

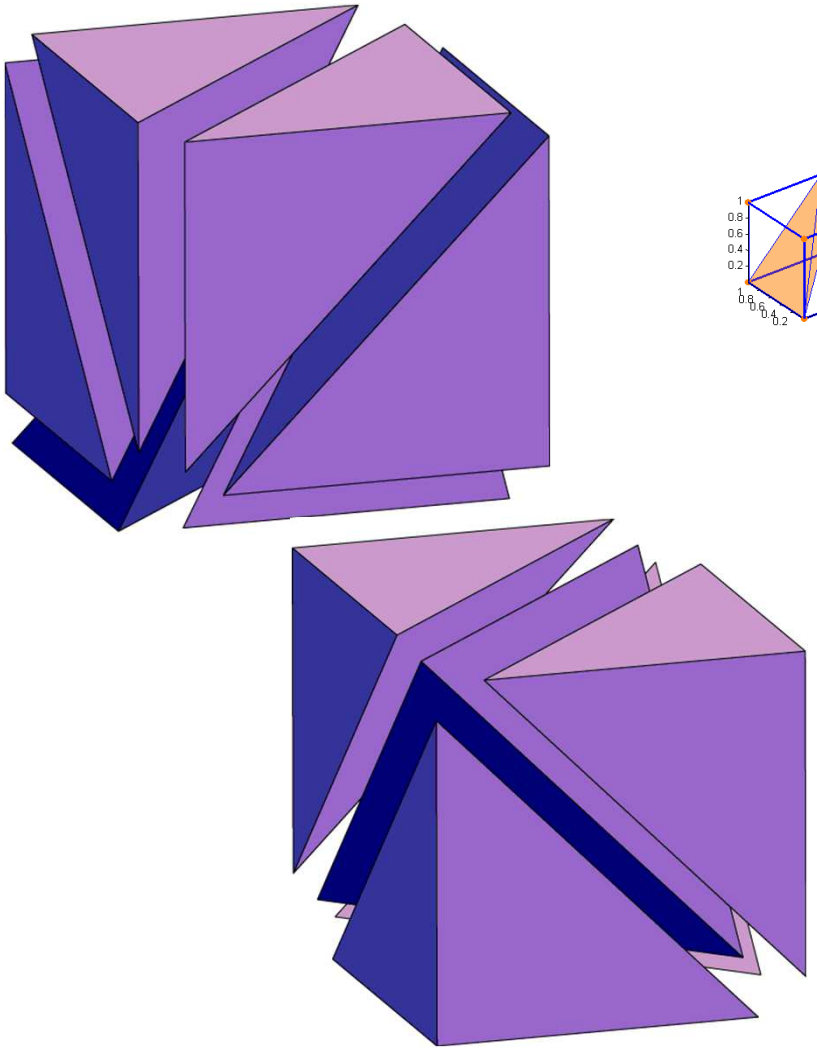
- Extraction d'une iso-courbe sur une triangulation

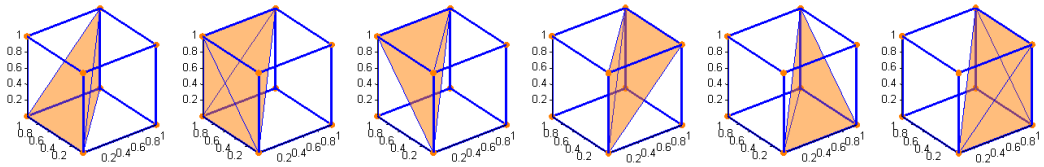


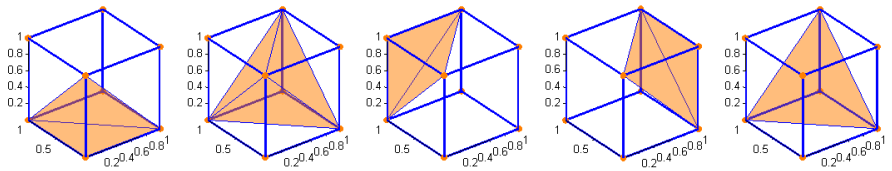
- Extraction d'une iso-surface dans une tetraédrisation



Un cube = 5 ou 6 tétraèdres







Pour aller plus loin

- CODING ADVENTURE/Marching cubes : https://www.youtube.com/watch?v=M3iI2l0ltbE

- GPU
  - sur UNREAL Engine 5
  - Sur UNITY : https://www.youtube.com/watch?v=vogqQ2v9Oxw
  - En Cuda.

2.3. Conversion : F(x,y,z) ➔ Brep

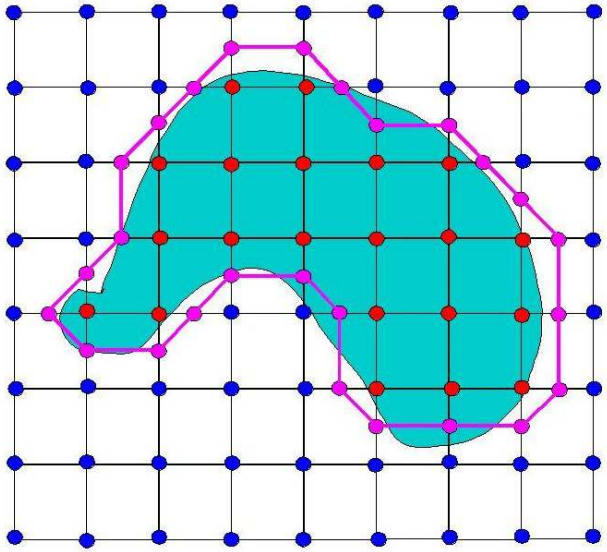
Quand il n'existe une représentation paramétrique de la frontière de F(x,y) alors le cas devient complexe…

*f_noise3d(x*2, y*2, z*2)-0.3*
contained_by{box{-2,2}}

F(x,y,z) =0 ➔ ?

?

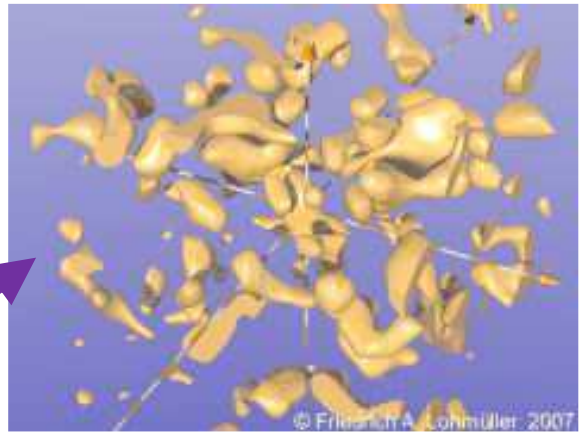
Enumération



Brep : Polygones
(ou polyèdres en 3D)

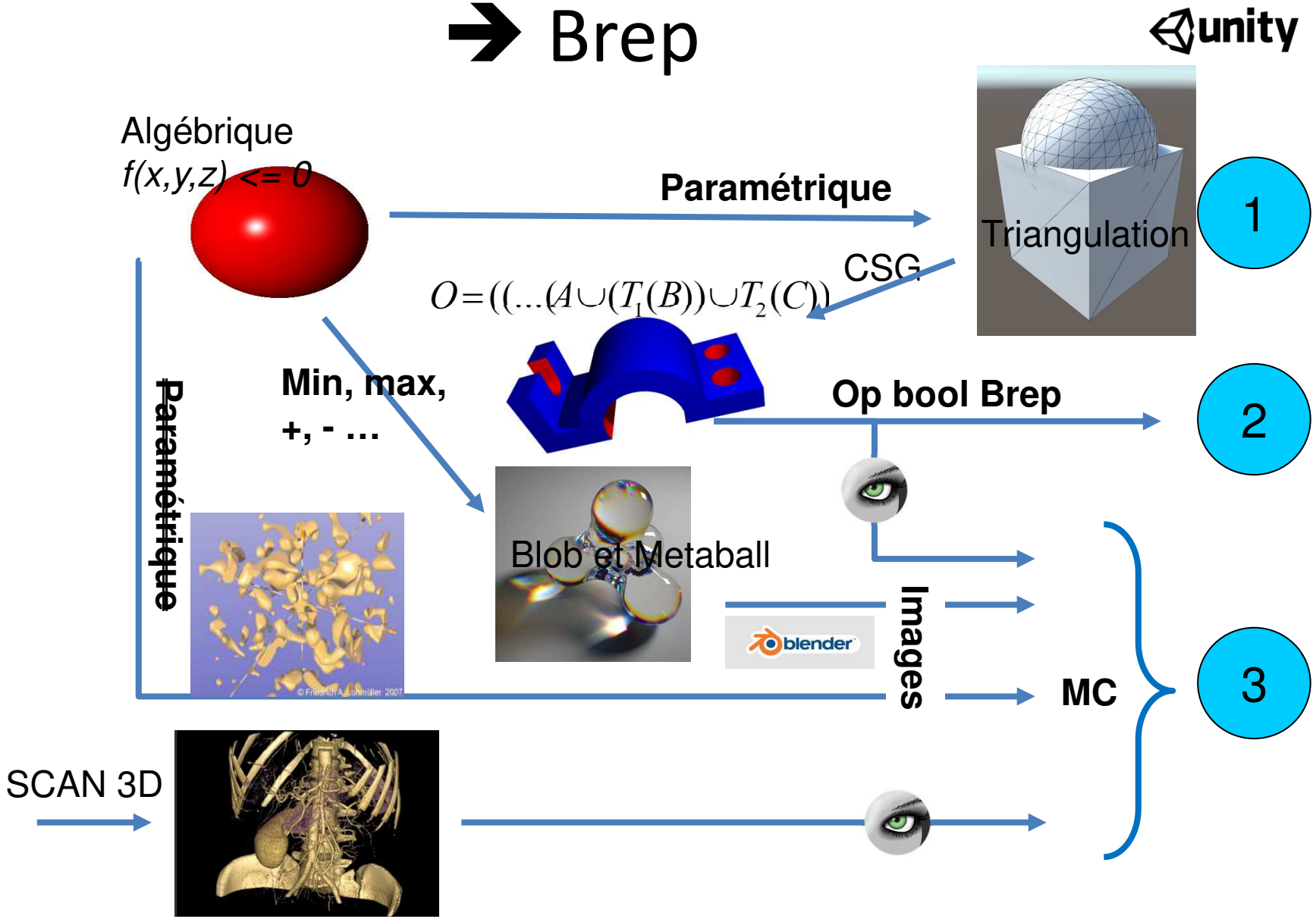
MC

Brep



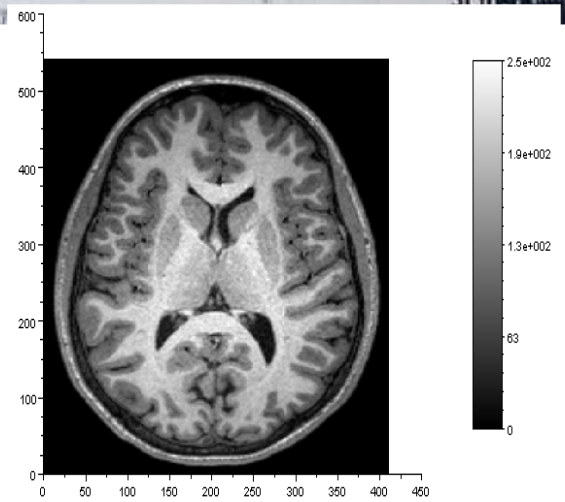
Algorithmes de conversion

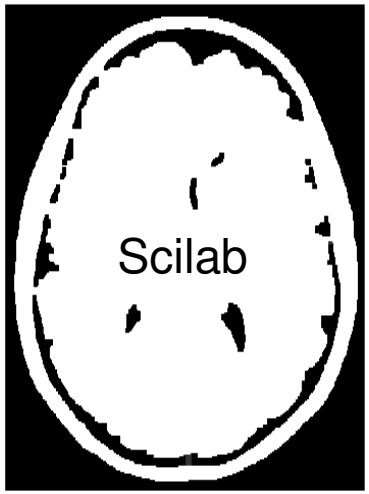
➔ Brep



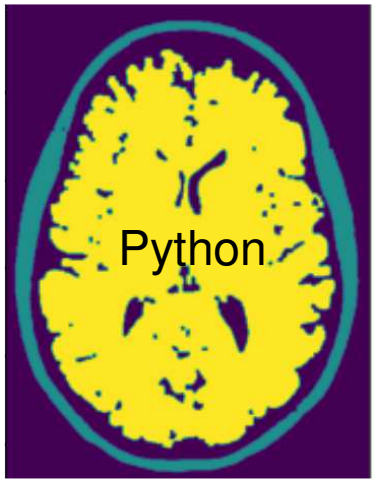
TP n°6 : Imagerie

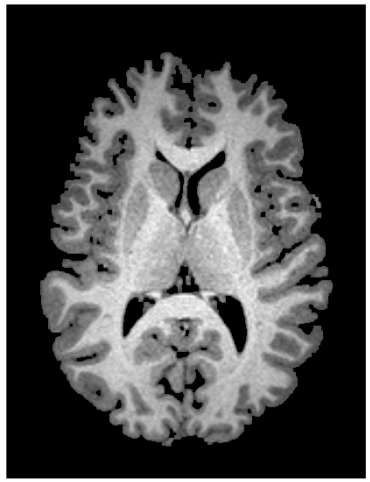
- Passage XY ← I,J
- Seuillage
- Erosion + dilatation
- Extraction des CC …





ou



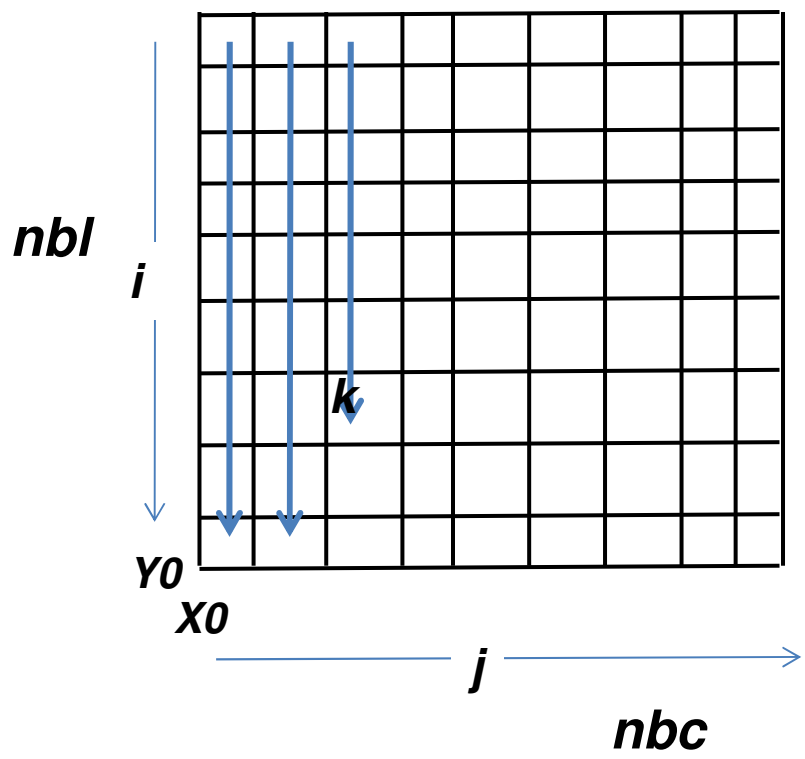


Grille = matrice dans Scilab

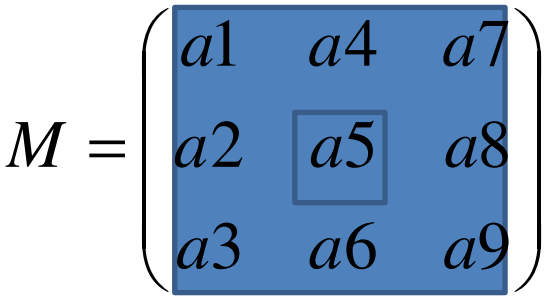
$$M = \begin{pmatrix} a1 & a4 & a7 \\ a2 & a5 & a8 \\ a3 & a6 & a9 \end{pmatrix}$$

$k=i+j*(nbl)$;

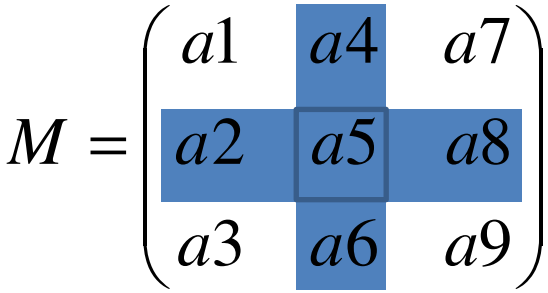
$M(i,j)==M(k)$



Voisinage et adressage



Voisinage en 8-connexité



Voisinage en 4-connexité

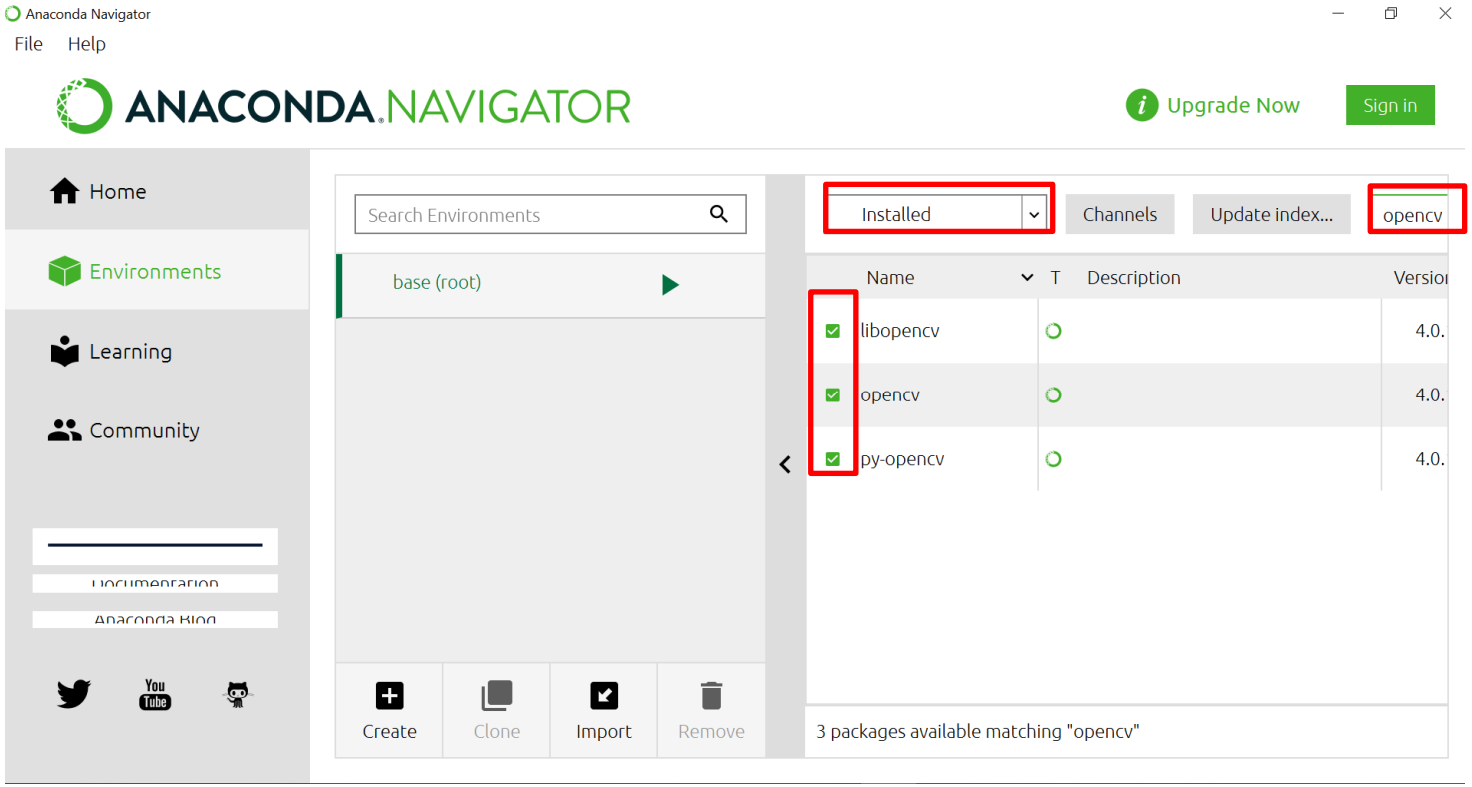
```
// Le voisinage de la cellule 5,8
Dij=[-1,0,1]
G2D(5+Dij,8+Dij)
```

```
Di=[-1,-1,-1; 0,0,0; 1,1,1]
Dj=[-1, 0, 1;-1,0,1;-1,0,1]
K=Di+5+(nbl)*(8-1+Dj)
G2D(K)
```

exercice

TP n°6 : imagerie

- Installer les packages et importer (import cv2)



TP n°6 : imagerie


```
img = cv2.imread(« file.jpg »,cv2.IMREAD_GRAYSCALE)

plt.imshow(img)

ret,img_thresh1 = cv2.threshold(img,threshold,…)

image_erosion = cv2.erode(img_thresh1,kernel_H,iterations = 2)
image_dilation = cv2.dilate(image_erosion,kernel_H,iterations = 3)

ret, markers = cv2.connectedComponents(image_erosion)
```