HMIN318M

Imagerie (médicale) 3D

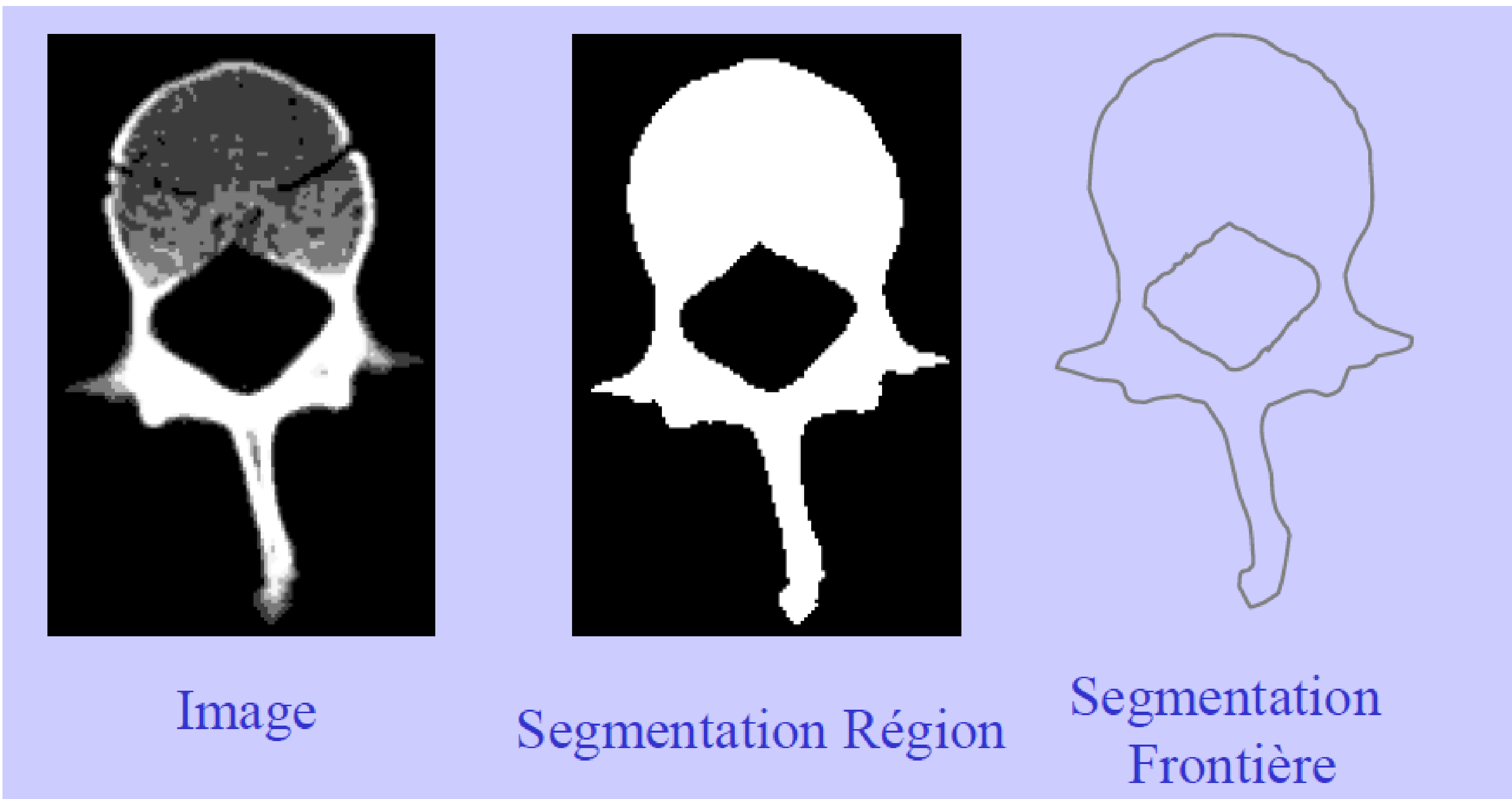
Segmentation (2/2)

Noura Faraj
noura.faraj@umontpellier.fr

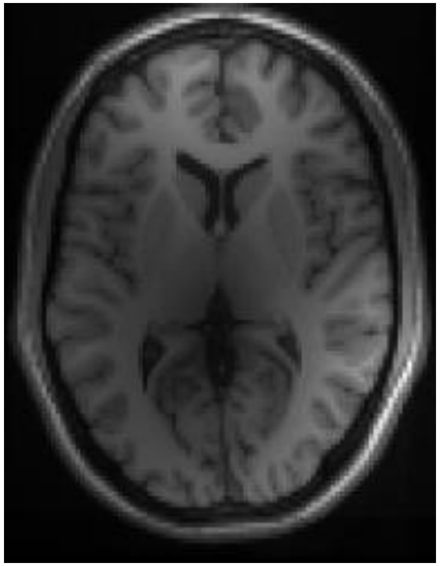
Source Gérard Subsol

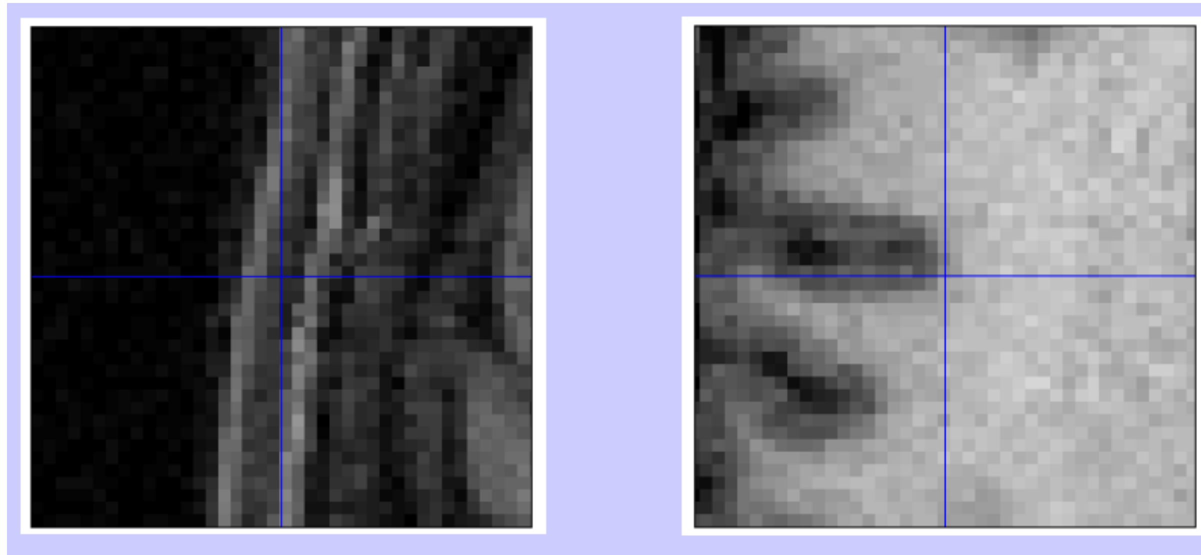
Le problème de la segmentation (3)

- Détermination des **voxels** appartenant à la ROI.
  - Distribution de probabilités d'appartenance à chaque voxel
- Définition de la surface **frontière**



Limites des méthodes région ?





- Les artefacts d’acquisition (inhomogénéité du champ en IRM, volume partiel en CT et IRM…) font varier l’intensité des structures dans l’image.

→ Une notion plus stable peut être celle de la **discontinuité** d’intensité (= frontière, **contour**…)

Quelques méthodes de segmentation 3D «frontière »

Généralités

- Fondées sur la notion de discontinuité d'intensité

  → Calcul du **gradient 3D** (par filtrage linéaire, par exemple Sobel)

$$\nabla f = grad\ f = < \frac{\partial f}{\partial x}(x,y,z), \frac{\partial f}{\partial y}(x,y,z), \frac{\partial f}{\partial z}(x,y,z) >$$

$\boldsymbol{G_z}$=

-1	-2	-1
-2	-4	-2
-1	-2	-1

$(\boldsymbol{x, y, z-1})$

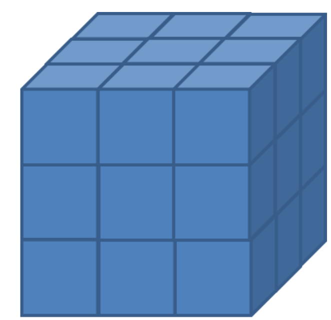
0	0	0
0	**0**	0
0	0	0

$(\boldsymbol{x, y, z})$

+1	+2	+1
+2	+4	+2
+1	+2	+1

$(\boldsymbol{x, y, z+1})$

$*$



Amplitude du gradient

$$\boldsymbol{G = \sqrt{G_x^2 + G_y^2 + G_z^2}}$$

→ valeur élevée mais quelle valeur ?

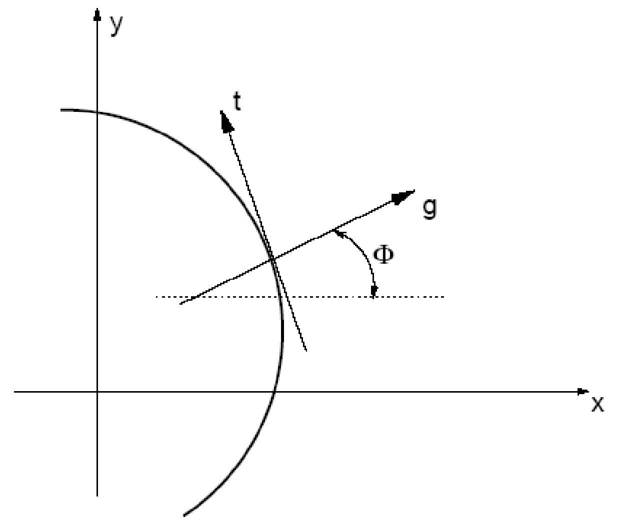
Orientation du gradient

$$\boldsymbol{\theta = arctan(\frac{G_y}{G_x})} \qquad \boldsymbol{\phi = arctan(\frac{G_x}{G_z})}$$

Généralités

- Fondées sur la notion de discontinuité d'intensité
  - → Dérivée de la norme du gradient dans la direction du gradient = 0
  - → En général approximée par l'opérateur **Laplacien**

$$\Delta i = \frac{\partial^2 i}{\partial g^2} + \frac{\partial^2 i}{\partial t^2} = \frac{\partial^2 i}{\partial x^2} + \frac{\partial^2 i}{\partial y^2} + \frac{\partial^2 i}{\partial z^2}$$



$$\frac{\partial^2 i}{\partial t^2} = 0 \qquad\qquad \Delta i = 0 \Leftrightarrow \frac{\partial^2 i}{\partial g^2} = 0$$

(peu de variation tangentielle)

→ Filtres linéaires qui approximent le Laplacien discret

Example 2D

Approximations du laplacien par différences finies

$$\begin{aligned}\Delta f &\approx [f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)]-4f(x,y)\\ &\approx f\star\Delta_{\text{dis}}\end{aligned}$$

ou

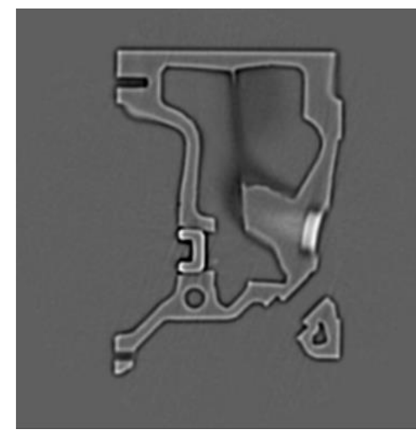
$$\begin{aligned}\Delta f \approx\ & [f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)+\\ & f(x+1,y+1)+f(x-1,y-1)+f(x-1,y+1)+\\ & f(x+1,y-1)]-8f(x,y)\approx f\star\Delta'_{\text{dis}}\end{aligned}$$

avec :

$$\Delta_{\text{dis}}=\begin{array}{|c|c|c|}\hline 0 & 1 & 0\\ \hline 1 & -4 & 1\\ \hline 0 & 1 & 0\\ \hline\end{array}\qquad \Delta'_{\text{dis}}=\begin{array}{|c|c|c|}\hline 1 & 1 & 1\\ \hline 1 & -8 & 1\\ \hline 1 & 1 & 1\\ \hline\end{array}$$

Généralités

- Mais comment définir le passage par 0 dans une image discrétisée et bruitée?
  - → Zones d'intensité constante (norme gradient)
  - → Trous dans les frontières
  - → Frontières épaisses



Les algorithmes qui suivent essayent de résoudre ces problèmes.

Algorithme du «livewire»

E.N. Mortensen, W.A. Barett. "Intelligent scissorsfor image composition. In: SIGGRAPH '95.

- Soit une image 2D $\mathrm{I}(i,j)$

Définir une fonction de coût pour passer d'un pixel P à un pixel voisin P'. Par ex. :

$$cost(P,P') = (g_{max} - ||g_{P'}||) + angle(g_P, g_{P'})$$

favorise les pixels avec un gradient élevé (= contour)

favorise les liens entre pixels qui ont des gradients de même direction

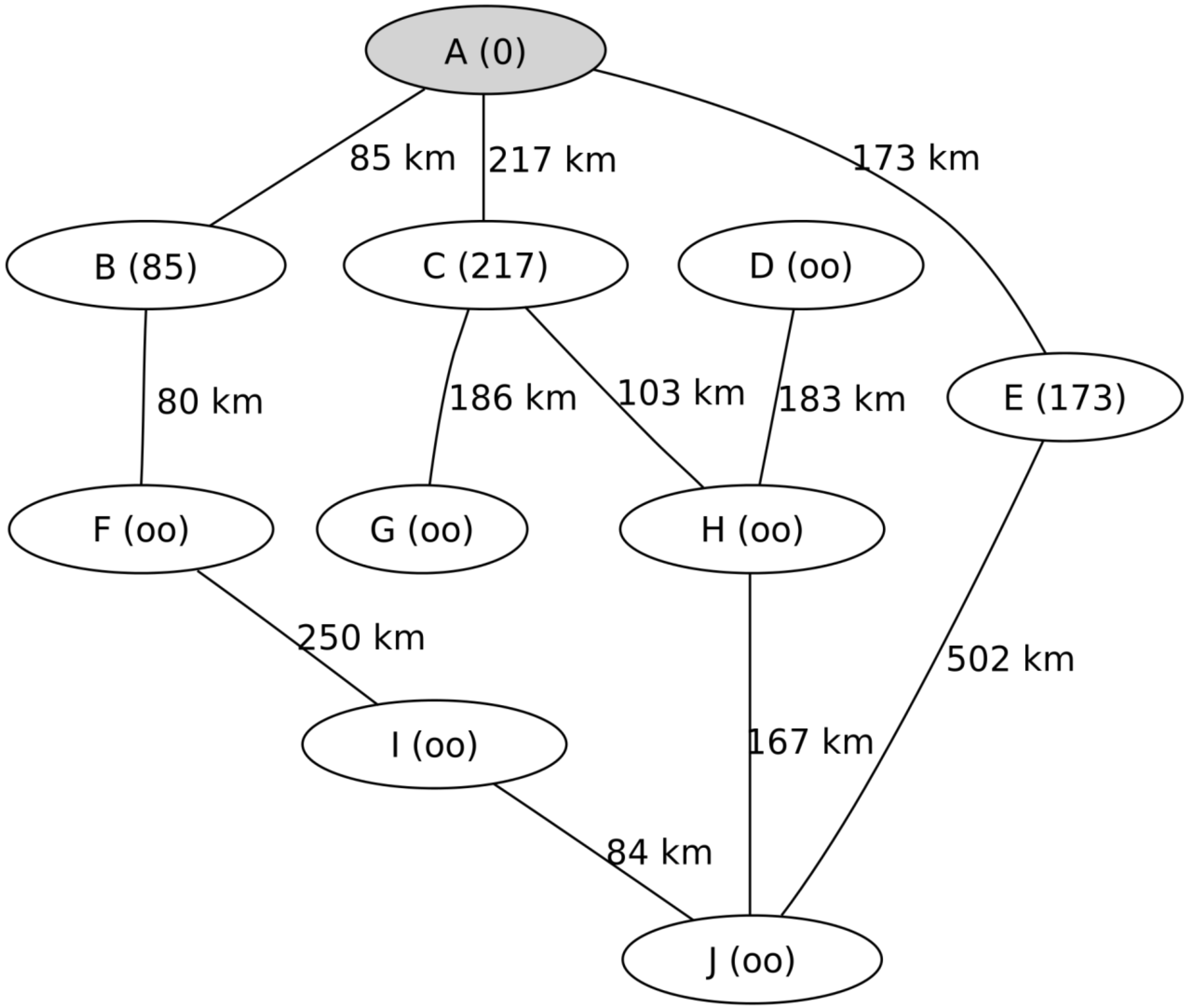
Coût faible si les deux points sont sur un contour (gradients élevés + angle faible)

- Fixer un pixel de départ O,
- Calculer, pour tous les pixels P de l'image, le chemin (O,P) qui minimise le coût par un algorithme de recherche du plus court chemin (voir algorithme de Dijkstra) dans un graphe

→ **frontière**

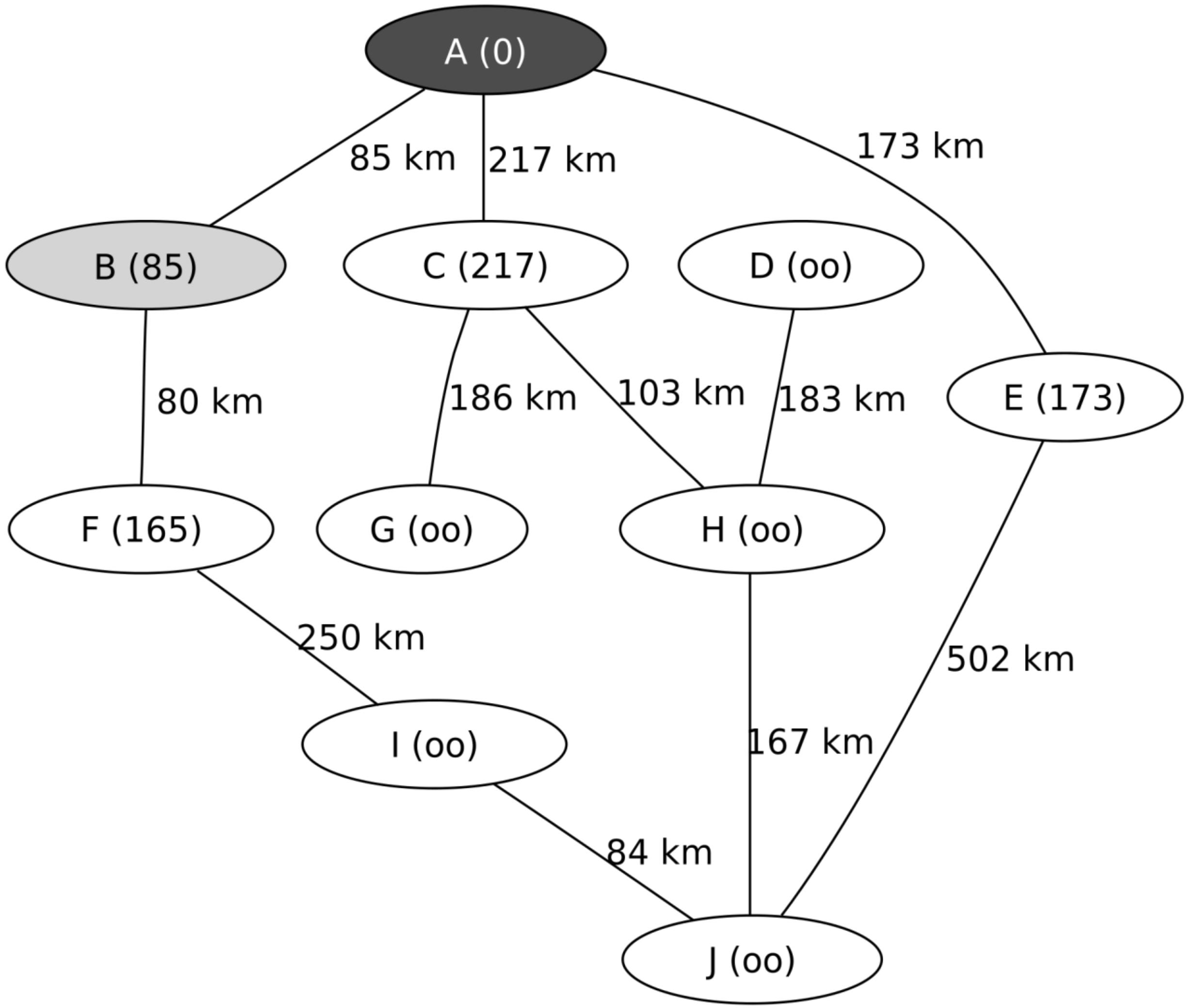
http://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra

Algorithme du «livewire»



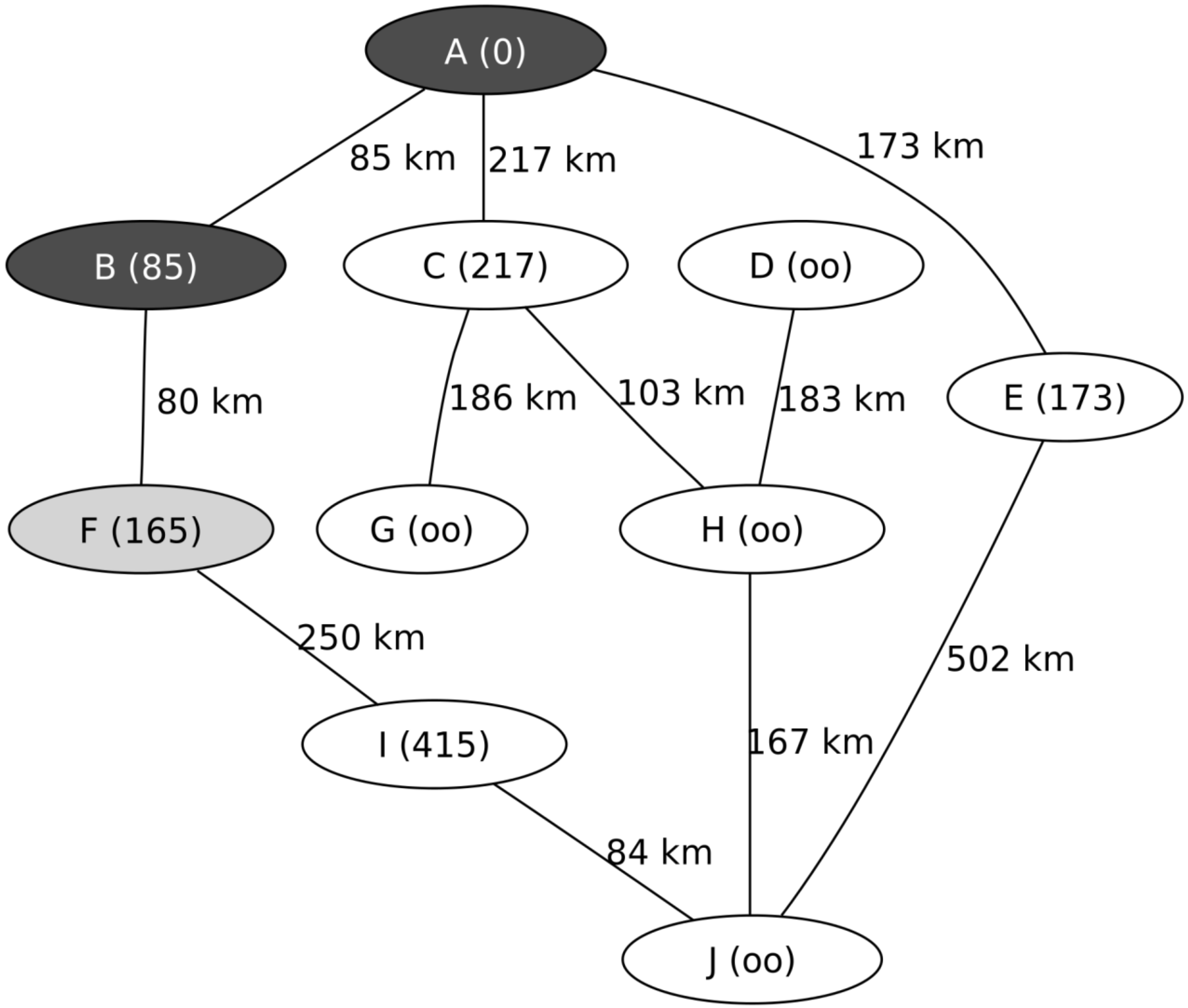
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



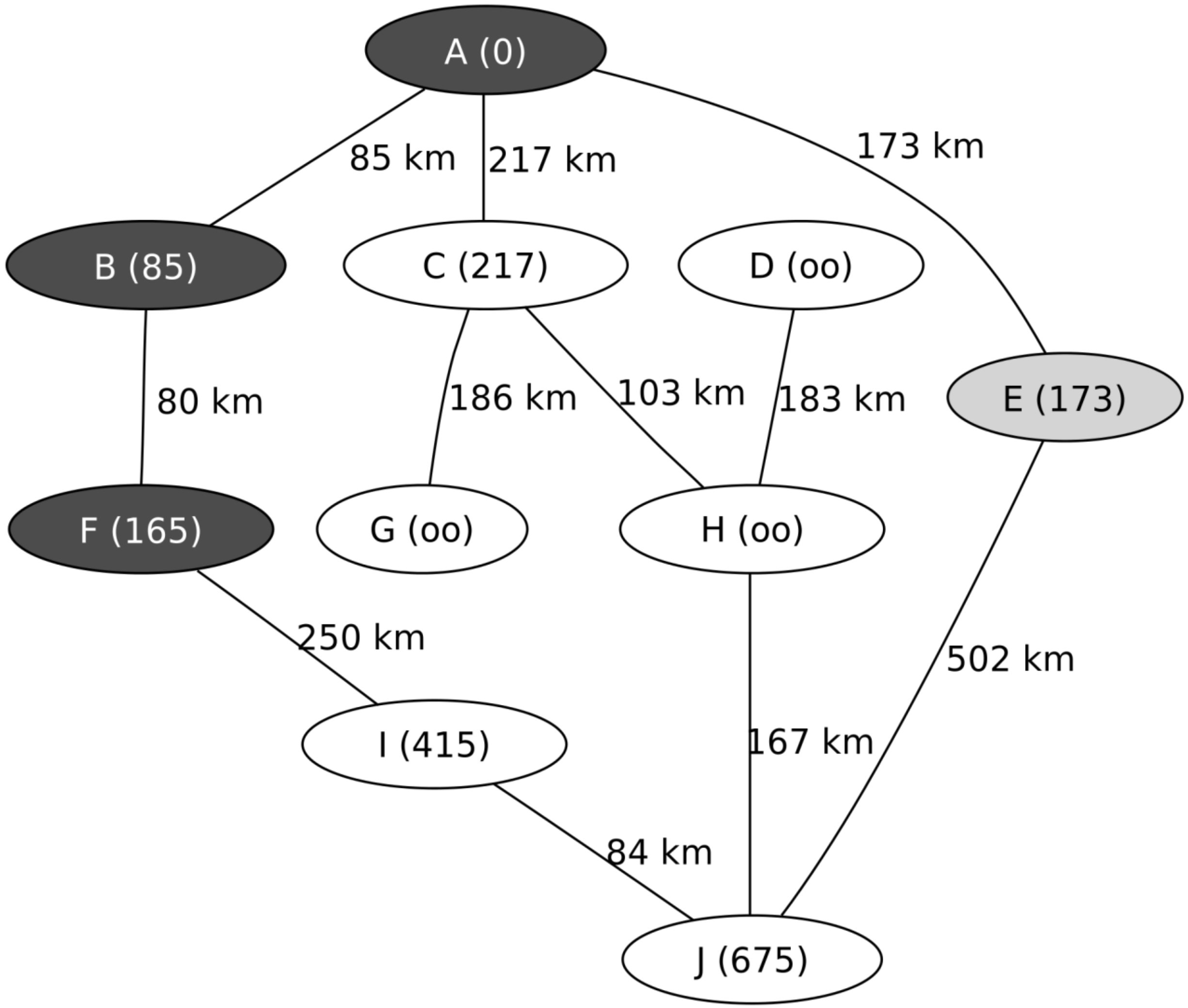
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



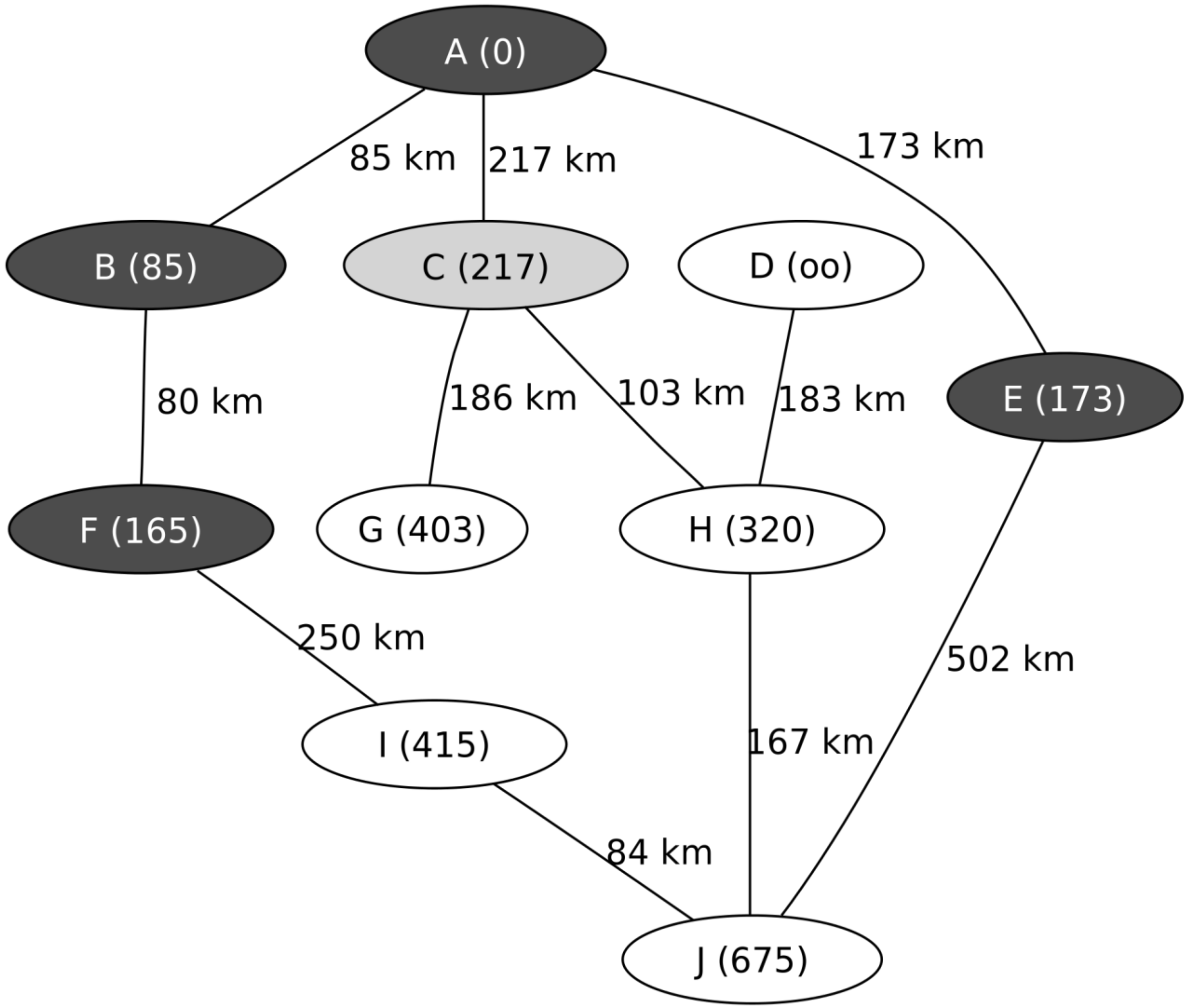
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



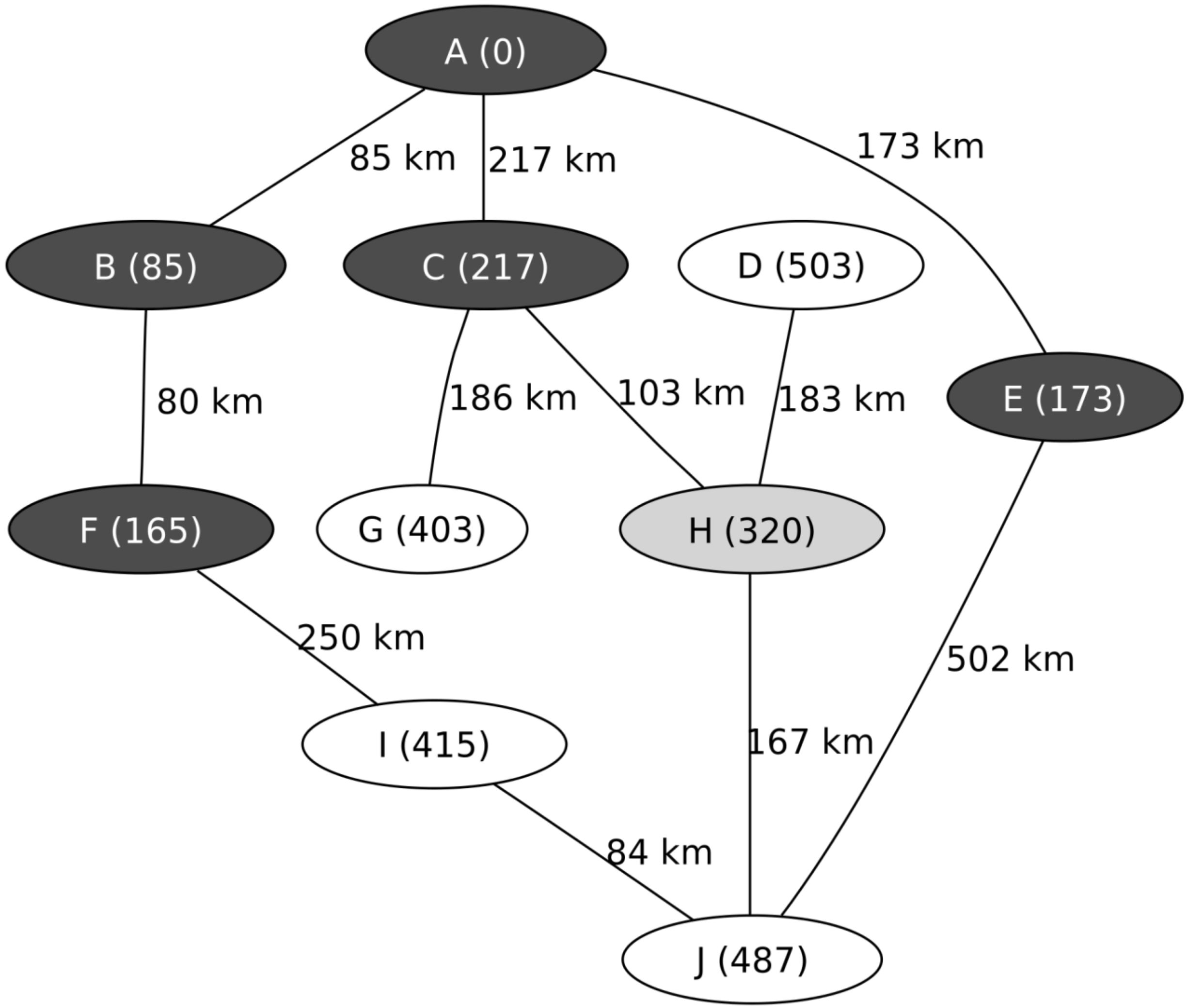
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



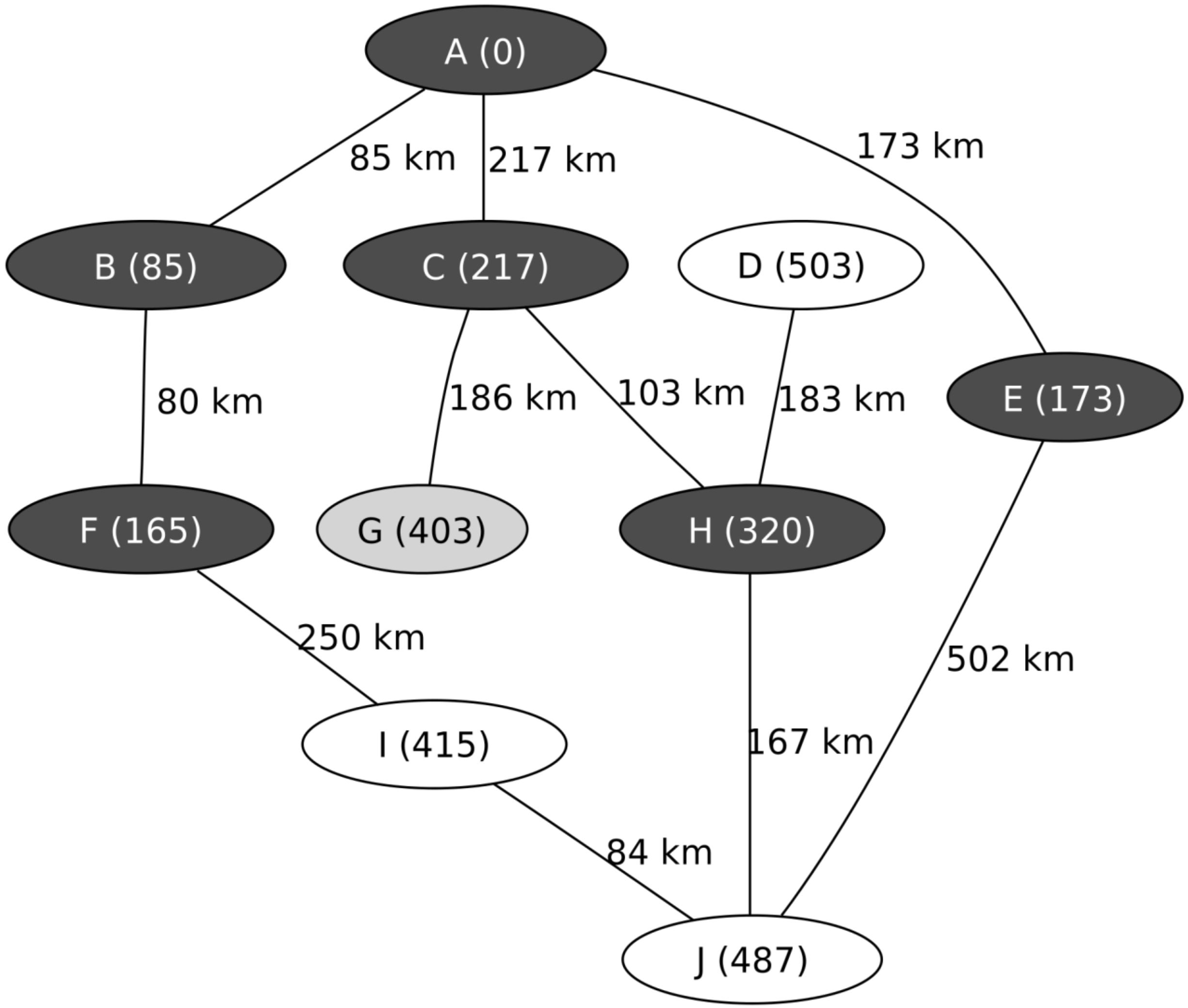
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



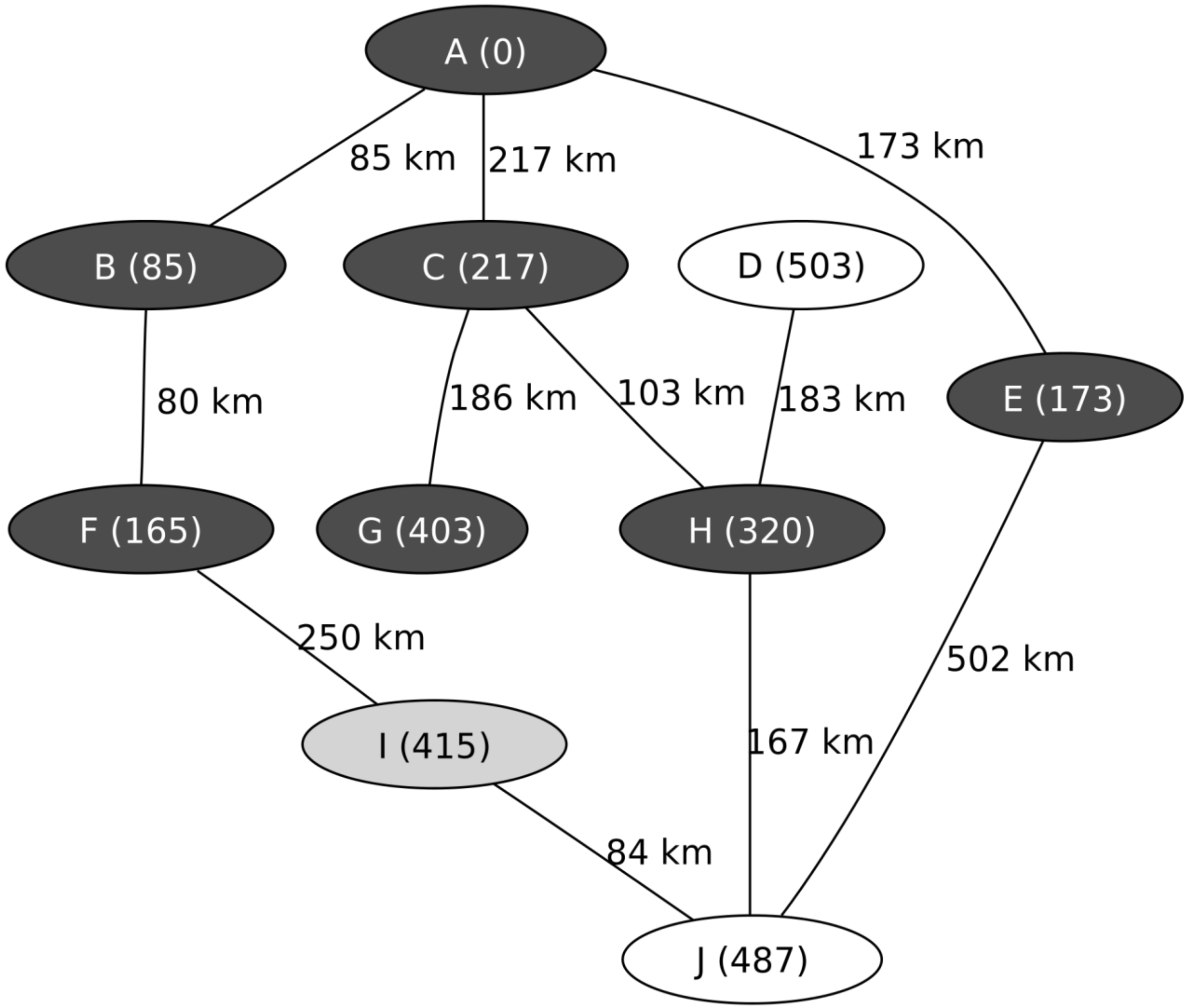
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



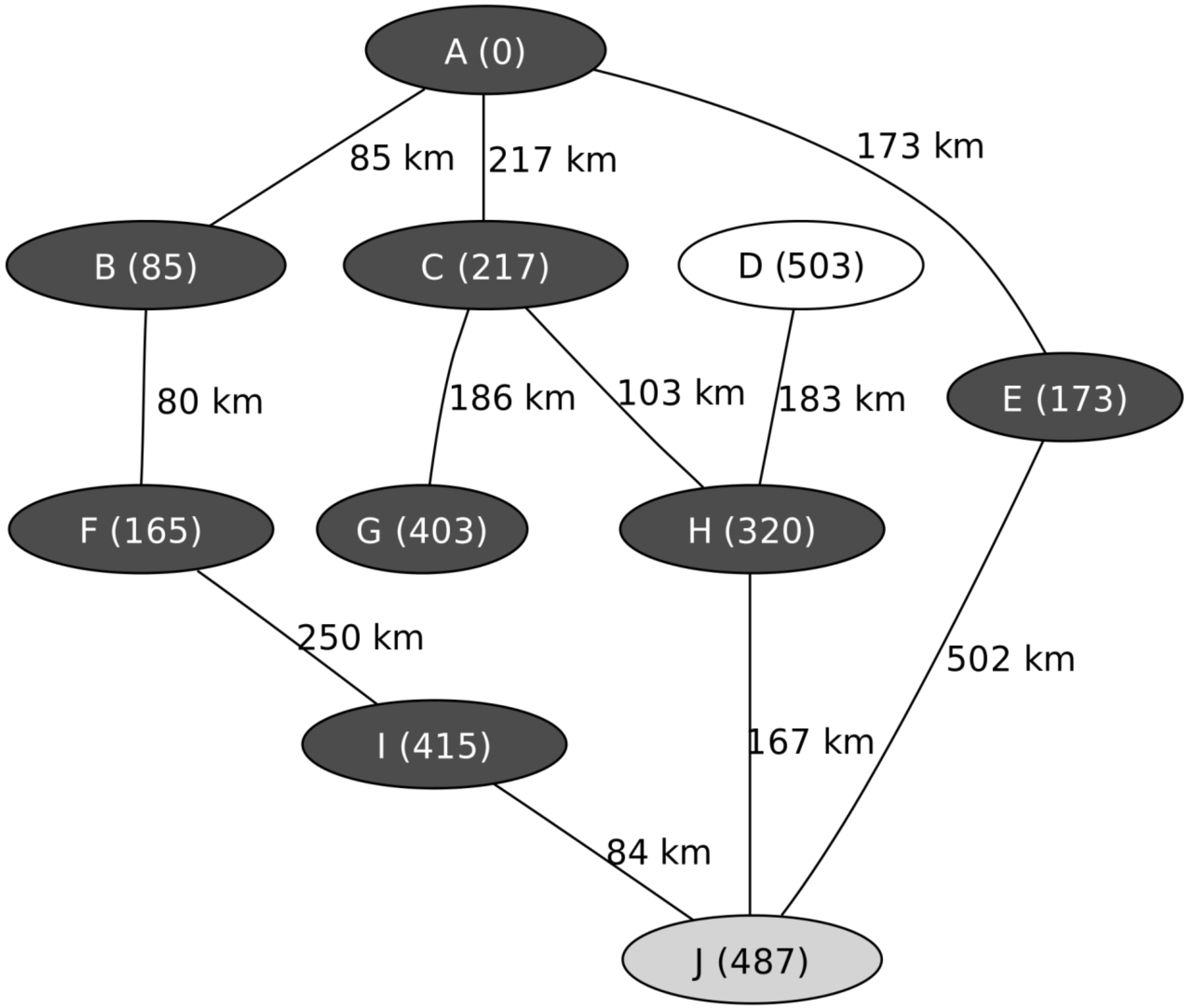
Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»



Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

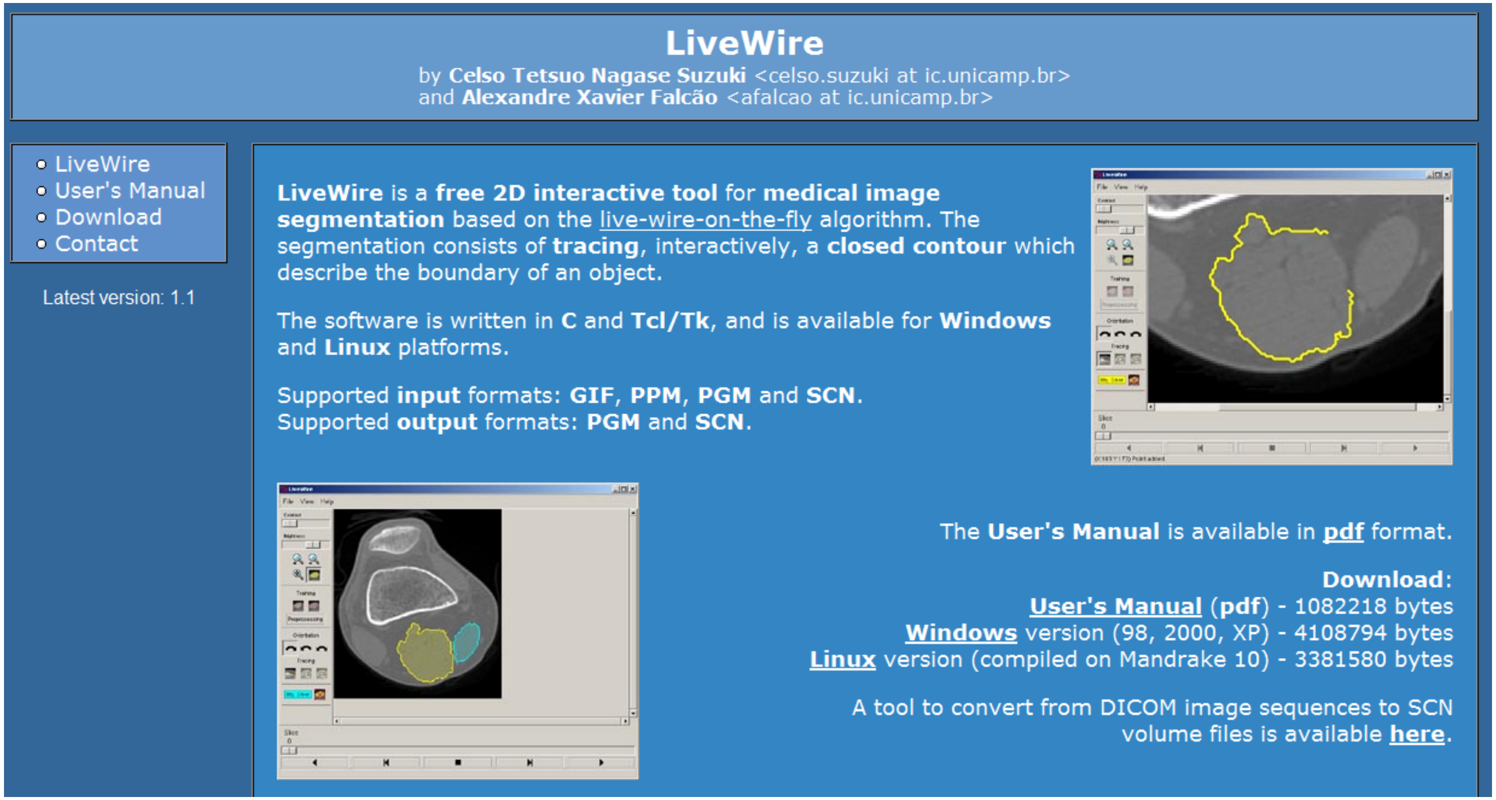
Algorithme du «livewire»



Algorithme du plus court chemin dans un graphe de Dijkstra(1959), Wikipedia

Algorithme du «livewire»

- L'algorithme Livewire est une méthode de segmentation interactive intuitive.
- Sensible au bruit, ce qui peut requérir un post-traitement (courbe déformable).
- **Difficilement généralisable en 3D** car on perd vite le côté intuitif (livewire orthogonaux ?).



Algorithme du « watershed »

- On a une image 3D I dont les intensités varient de imin à imax.
- On «inonde» l'image I.

S. Beucher, C. Lantuéjoul. Use of watershedsin contour detection. InInternational workshop on image processing, real-time edgeand motion detection(1979).

Initialisation

```
•i=imin
```

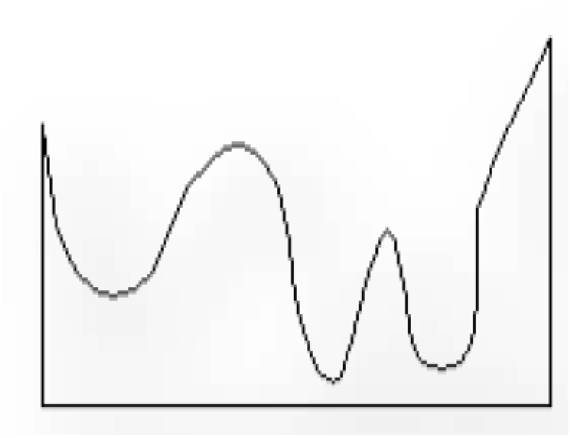
Procédure

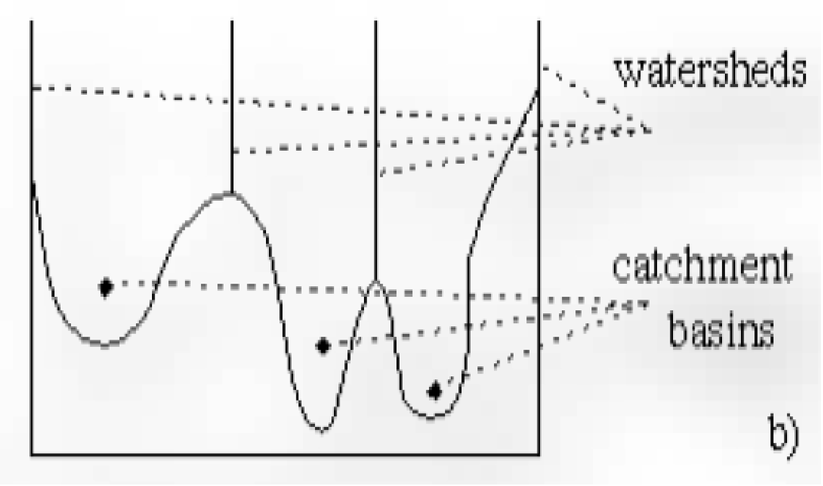
```
Tant que i < imax
 • Pour tous les voxels V d'intensité i qui ne sont pas labellisés
   • Si V connecté à des voisins tous labellisés Lj, V prend le label Lj
   • Si aucun voisin de V n'est labellisé, V crée un nouveau label Lk
   • Si V est connecté à des voisins de labels différents, V est labellisé
     W (watershed= ligne de partage des eaux)

 • i=i+1
```

- A la fin, on a partitionné l'image I en «bassins» définis par leurs labels Lk et leurs **frontières fermées** déterminées par tous les voxels environnants labellisés W.

Algorithme du « watershed »

- Intensité de départ i=imin
- On remonte le niveau de l'intensité i
- Il y a remplissage de zones bassins de l'image
- Quand deux bassins se rejoignent, on crée une frontière.

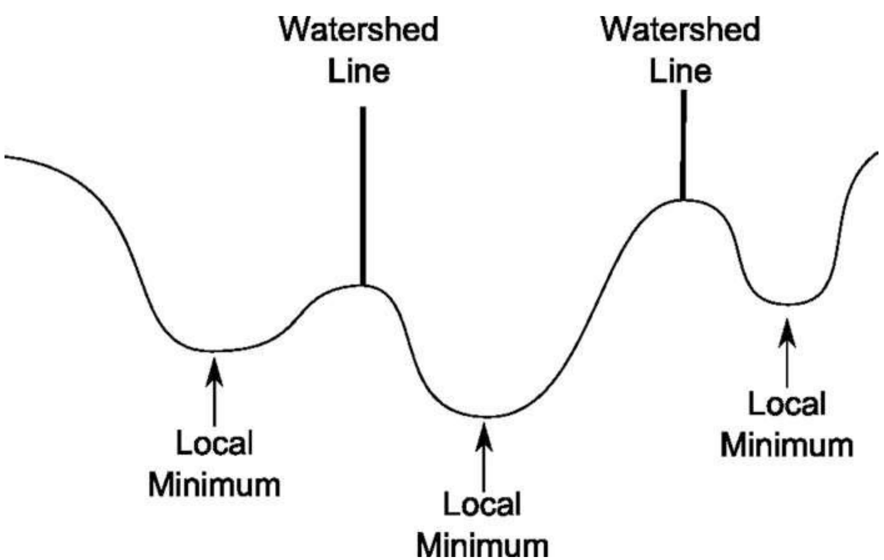




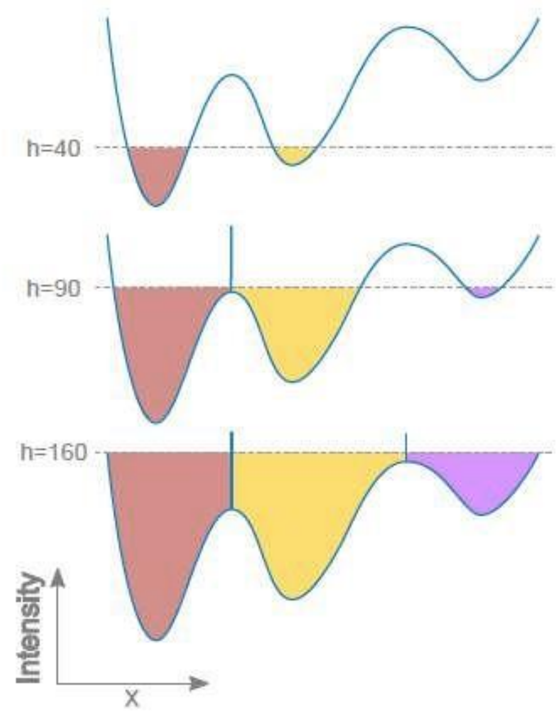
http://imagej.net/MorphoLibJ

Algorithme du « watershed »

- Considérer les niveaux de gris comme des altitudes
- Identifier les minima locaux
- Inonder les bassins à partir des minima
- Séparer les bassins par des barrages → le watershed

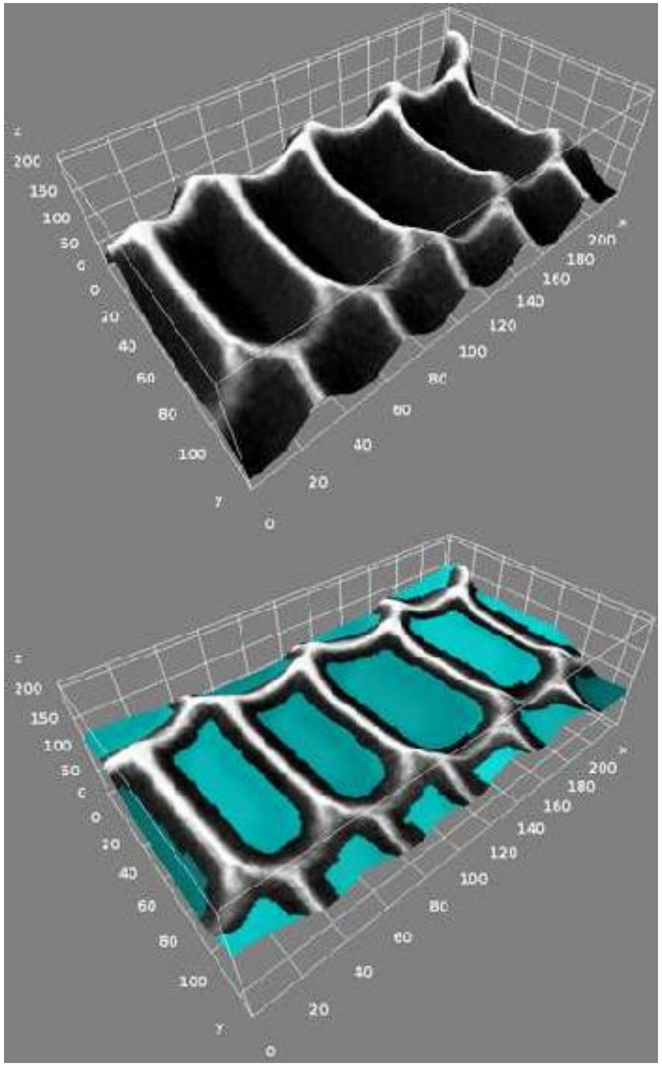


http://imagej.net/MorphoLibJ



Algorithme du « watershed »

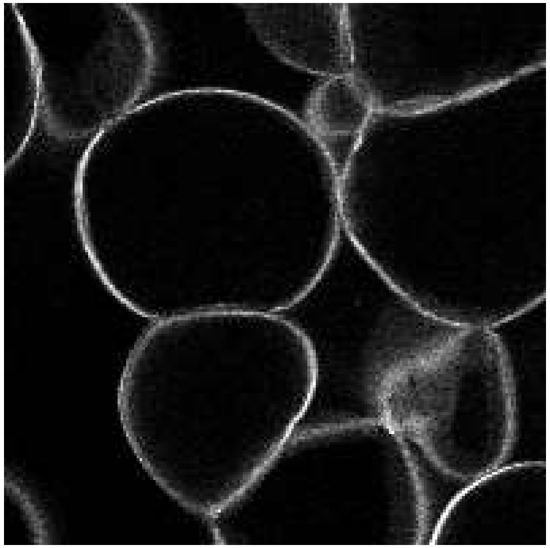
- Considérer les niveaux de gris comme des altitudes
- Identifier les minima locaux
- Inonder les bassins à partir des minima
- Séparer les bassins par des barrages → le watershed



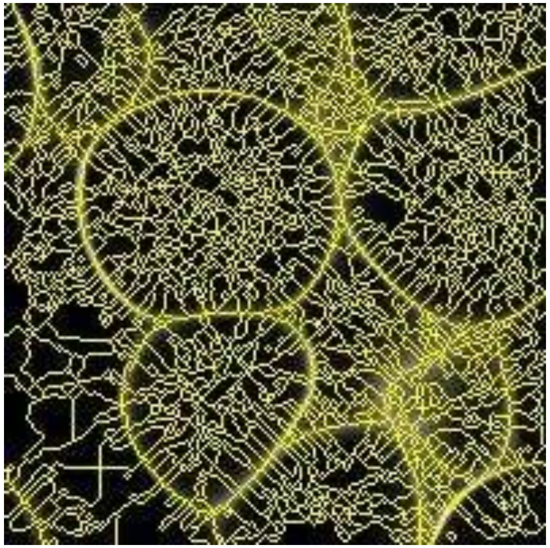
http://imagej.net/MorphoLibJ

Algorithme du « watershed »

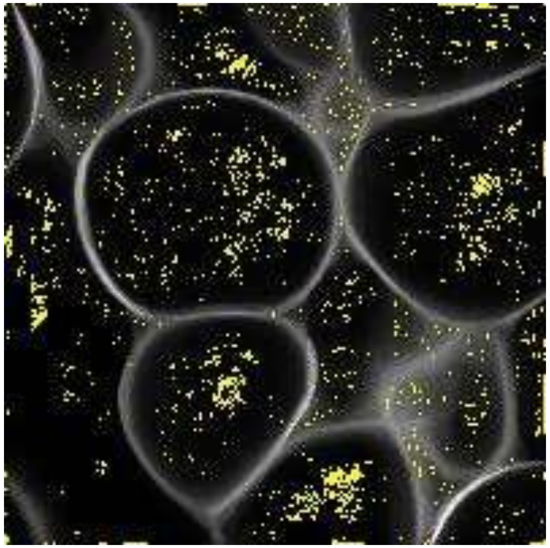
- Algorithme très utilisé en 2D, en particulier en imagerie biologique (structures contrastées).
- Se généralise facilement en 3D.
- Problème : la **sur-segmentation**, en particulier dès qu'il y a du bruit dans l'image…



original image



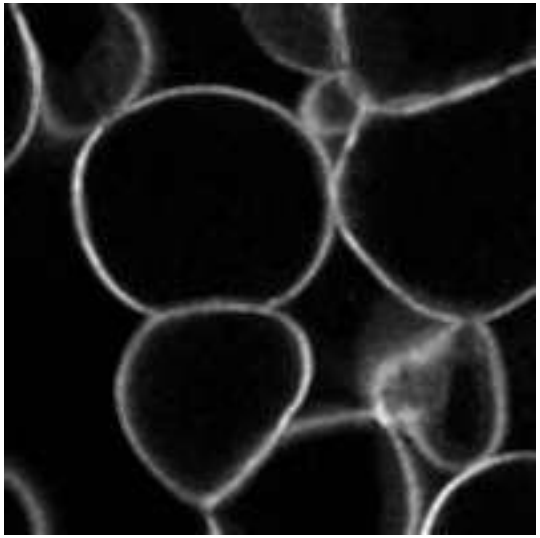
watershed segmentation

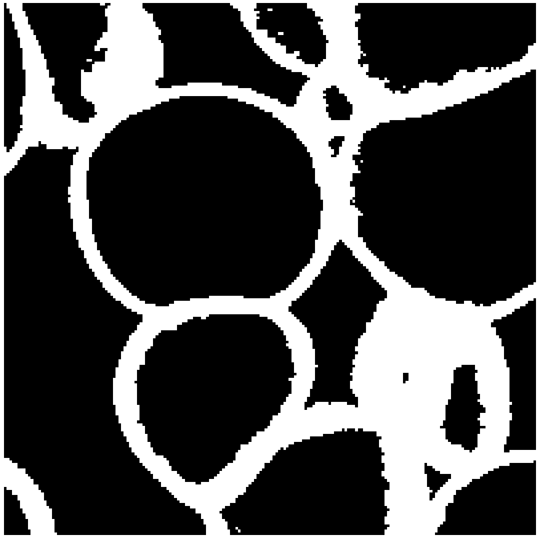


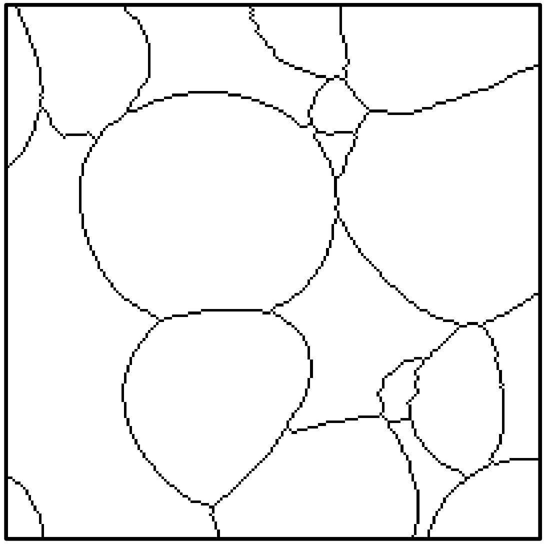
local minima

Solutions pour la sur-segmentation

- Idée : supprimer les minima non voulus
  - Filtrer l’image d’entrée (Gaussien, médian…)
  - Détecter automatiquement les minima
  - Utiliser des minima étendus

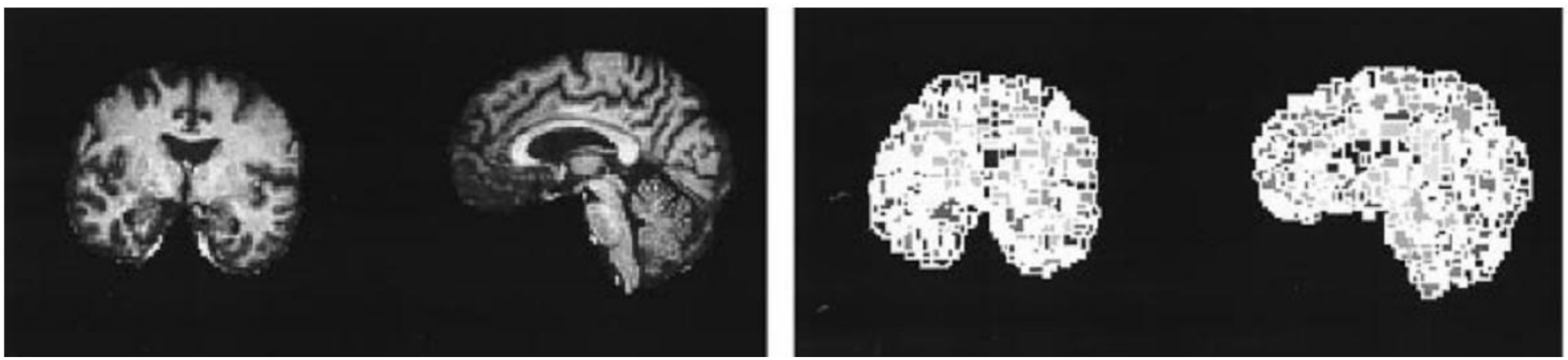






Algorithme du « watershed »

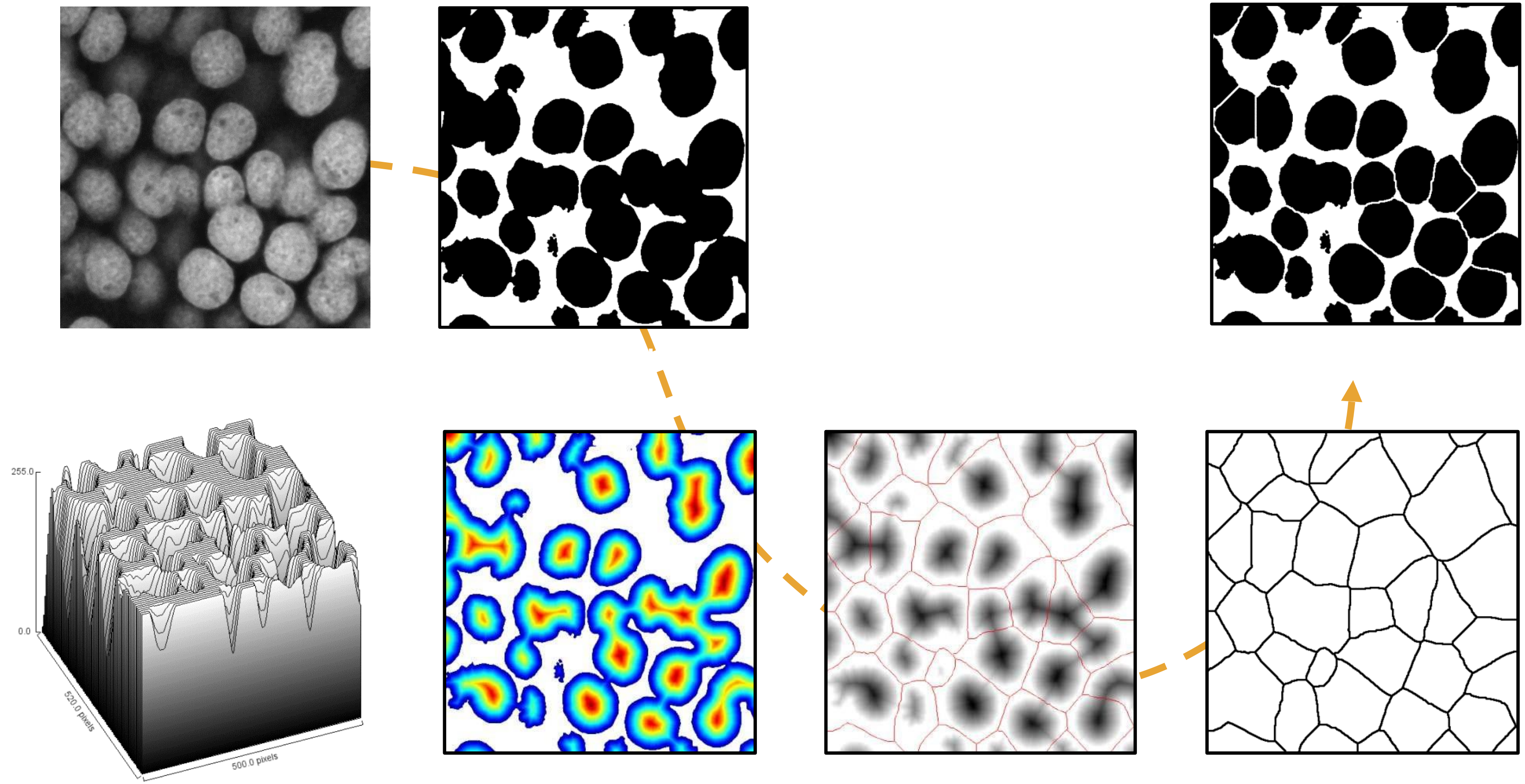
- Algorithme très utilisé en 2D, en particulier en imagerie biologique (structures contrastées).
- Se généralise facilement en 3D.
- Problème : la **sur-segmentation**, en particulier dès qu'il y a du bruit dans l'image…



- Fusion des bassins segmentés en fonction de critères de similarité.
- On peut aussi travailler sur **l'image gradient** (faible valeur : zones homogènes, forte valeur : contours).

→ assez peu utilisé ou alors en première étape d'un algorithme de segmentation.

Examples



http://imagej.net/Distance_Transform_Watershed

Algorithme du «level set»

Soit S une surface fermée plongée dans I, l'image 3D à segmenter.
Elle va former la frontière de la structure à segmenter.

Le processus de segmentation va être évolutif : S va se déformer au cours du temps :

$$\mathbf{P} \in S \rightarrow \mathbf{P}(t)$$

Supposons que S se déforme perpendiculairement :

$$\partial \mathbf{P}(t)/\partial t = v(\mathbf{P}(t))\, \mathbf{n}(\mathbf{P}(t))$$

Où :

- **n** est la normale à la surface en **P**(t) (orientée vers l'extérieur)
- v la fonction de déformation normale qui dépend de **P**(t)

On va définir S(t) par J(t) une image de même taille que I et dont les intensités vont évoluer dans le temps.

Soit **x** un voxel de l'image J(t) :

$$j(\mathbf{x},t) = \begin{cases} d(\mathbf{x},S(t)) \text{ si } \mathbf{x} \text{ est à l'extérieur de } S(t) \\ -d(\mathbf{x},S(t)) \text{ si } \mathbf{x} \text{ est à l'intérieur de } S(t) \end{cases}$$

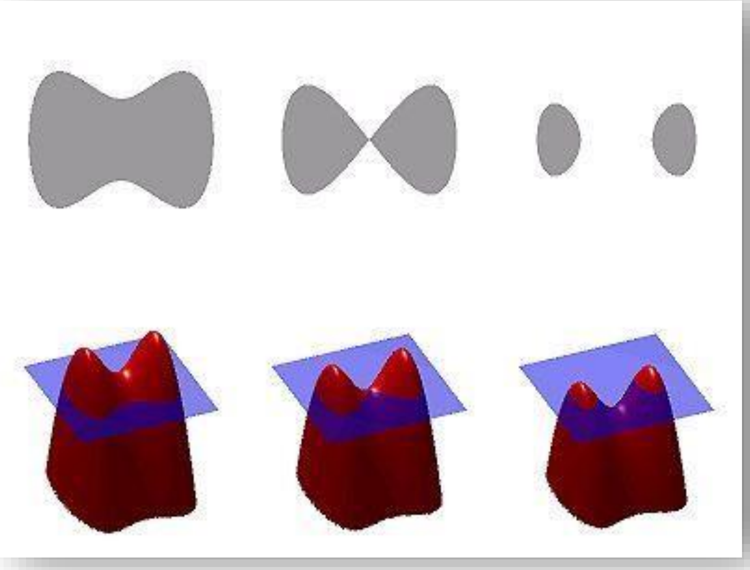
On a donc l'équivalence : $\{\text{ensemble des } \mathbf{x} \;/\; j(\mathbf{x},t)=0\} \leftrightarrow S(t)$

J(t) est une image qui représente S(t) suivant une **courbe/surface de niveau 0** (ou 0-*level set*).

Algorithme du «level set»

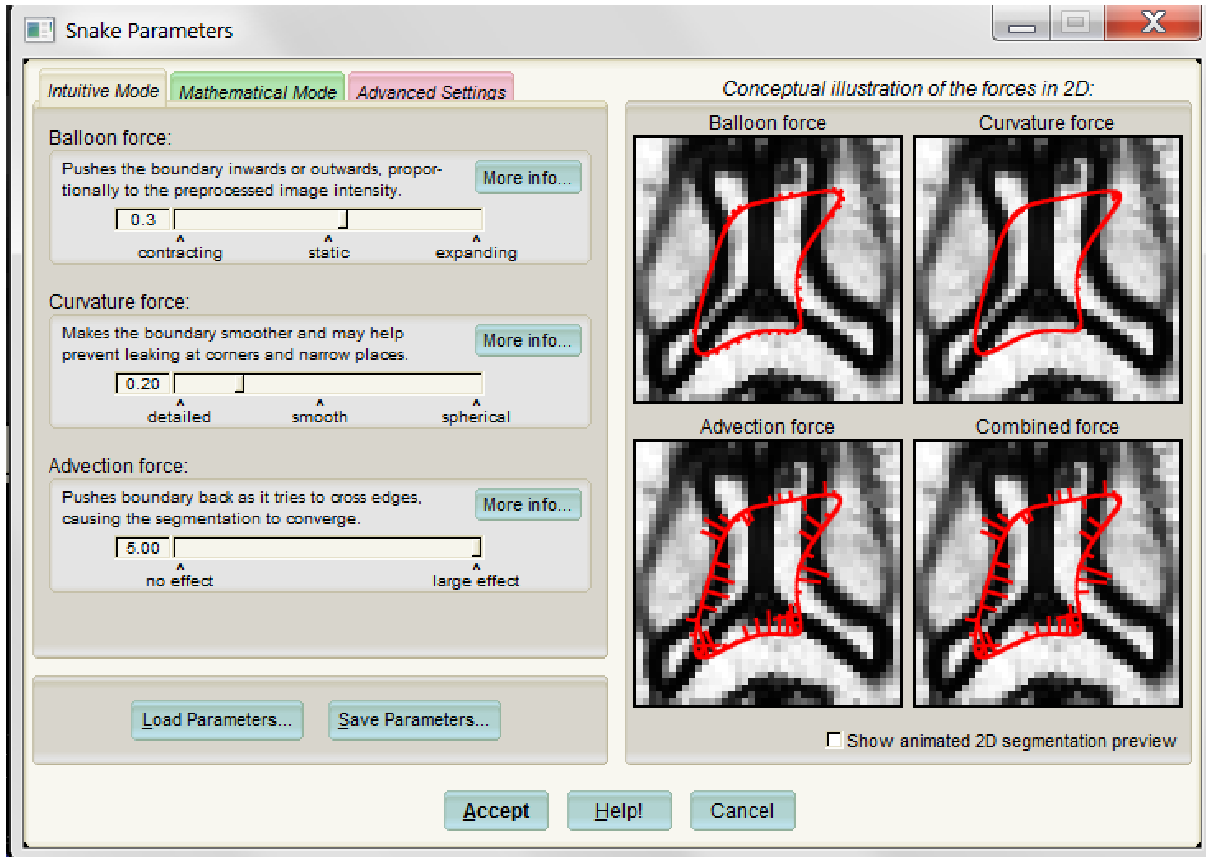
On peut donc définir un processus qui va faire varier les intensités j(**x**,t) autour du niveau 0 au cours du temps. Et à chaque instant, j(**x**,t)=0 définira la surface de segmentation S(t).

•Il s'agit d'une représentation dite **implicite**.
•La fonction v va dépendre des données (attraction), de la régularité locale que l'on souhaite pour S et du processus d'évolution (ballon).
•On peut même gérer en même temps plusieurs surfaces fermées ce qui peut être utile quand on ne connaît pas la topologie de la structure à segmenter.

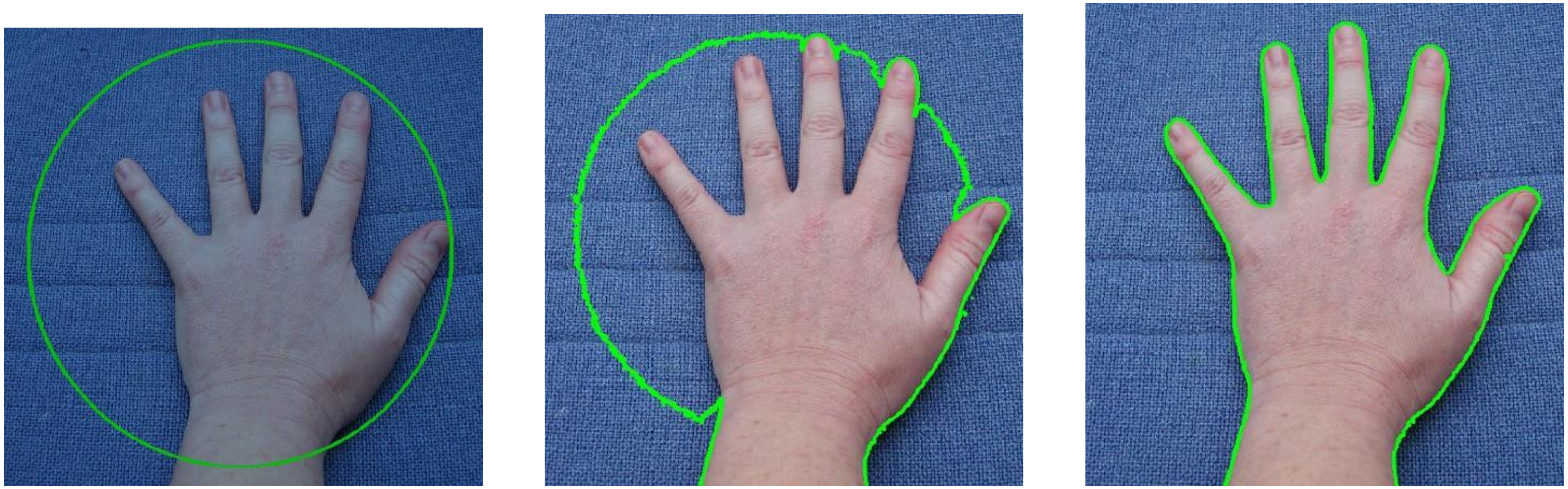


https://www.youtube.com/watch?v=75oVE86HeqU

Algorithme du «level set»



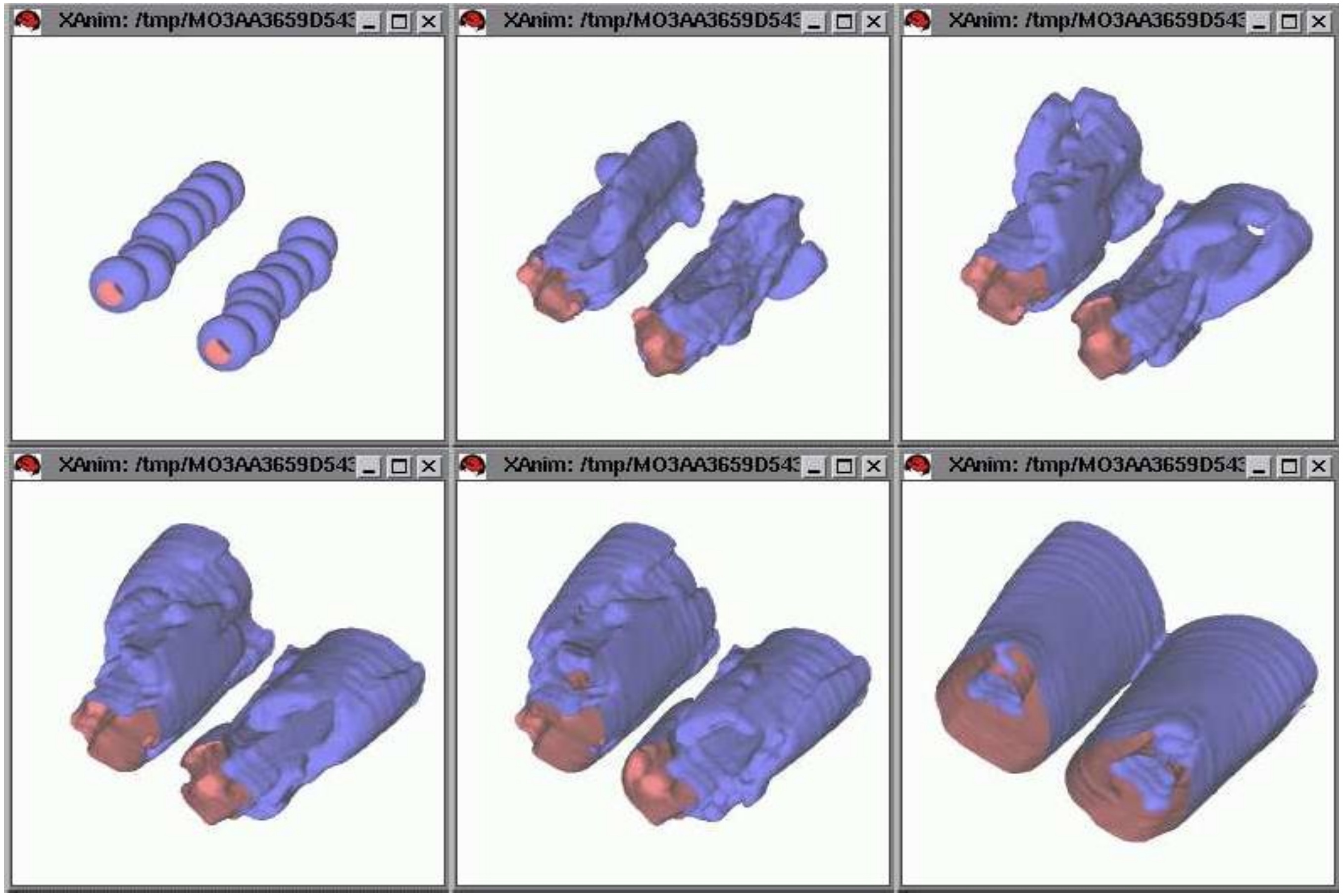
Exemple : segmentation 2D



images de D Lingrand (http ://www.polytech.unice.fr/ lingrand)

Exemple : segmentation 3D

- Segmentation des cuisses sur IRM (Malladi,Sethian)



Problèmes et limites

- Résolution numérique coûteuse (distance de tout l'espace à l'interface)
  - ⇒ fenêtre de résolution
  - ⇒ ou approximation polygonale
- Inflation constante ou déflation constante
- Arrêt du modèle non déterminé
- Difficile de rajouter de nouvelles contraintes (e.g., interaction utilisateur, autres attracteurs)

Méthode par «surface déformable » (ou snake en 2D)

Une surface continue ou discrète est directement plongée dans l'image et se déforme.

- Dans le cas continu, la surface est définie paramétriquement. On minimise une énergie qui combine un terme d'attraction vers les données (fort gradient, intensité fixée) et un terme de régularisation.
- Dans le cas discret (c.a.d. un maillage 3D), on fait bouger tous les sommets en fonction des données tout en gardant localement une régularité.

Méthode par «surface déformable » (ou snake en 2D)

La complexité est relative au nombre de paramètres (et du schéma de discrétisation) ou au nombre de sommets.

On peut raffiner localement en introduisant de nouveaux paramètres ou sommets.

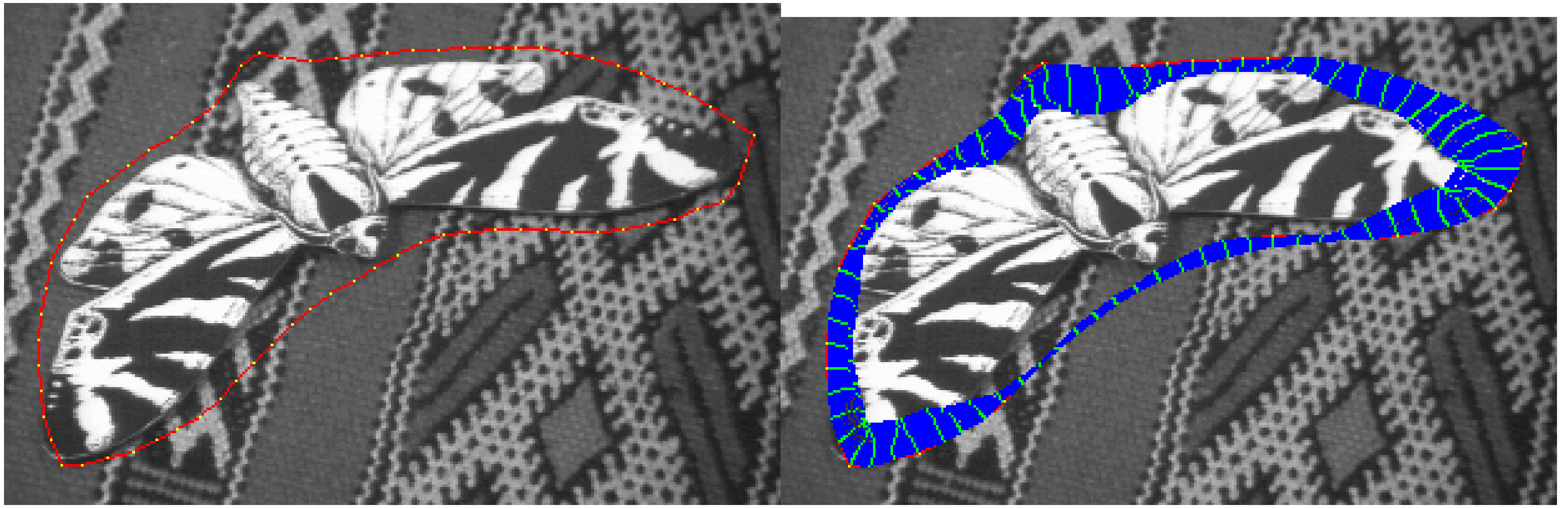
Plus difficile de gérer les changements de topologie.

Algorithme connu sous le nom de «snake» en 2D.

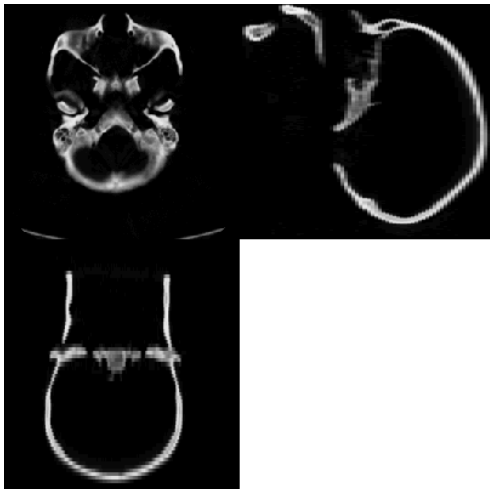
Contours actifs : exemple 2D

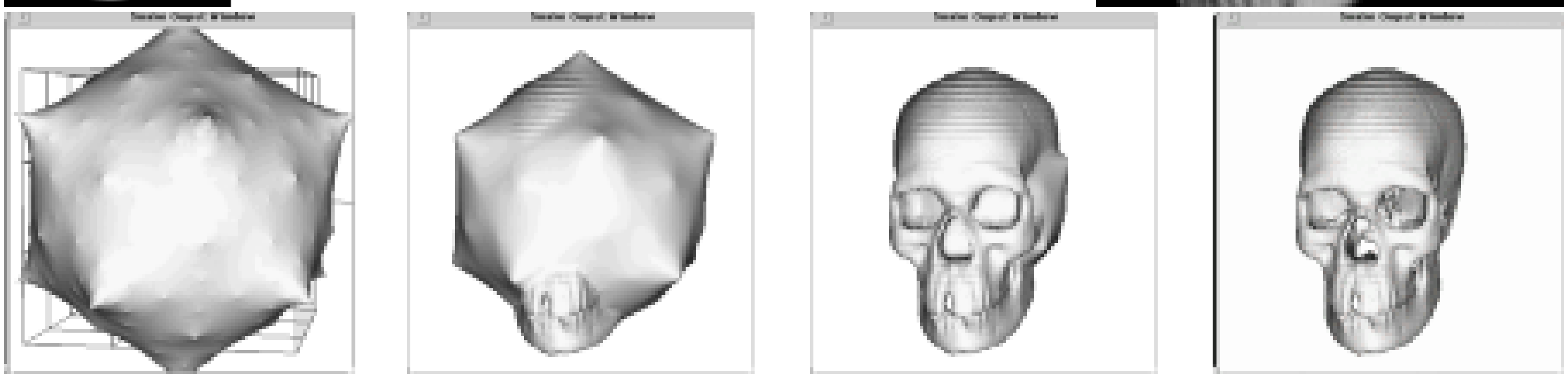
Contour actif ou Snake : optimisation itérative

- initialisation : courbe assez proche du contour à extraire
- itérations : déformations du contour actif de façon à ce qu’il atteigne une position d’énergie minimum.



Contours actifs : exemple 3D





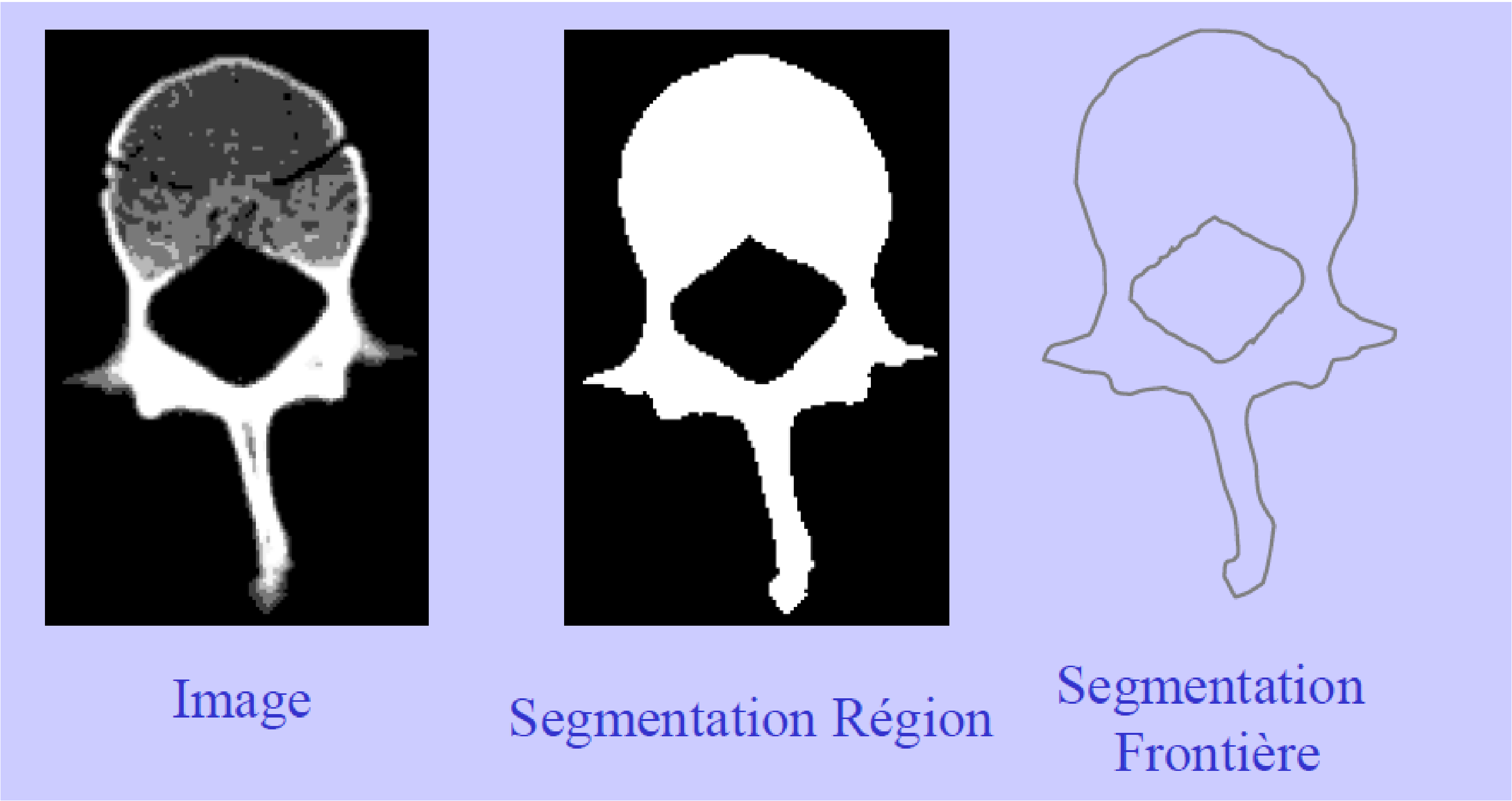
(a) (b) (c) (d)

Visualisation surfacique

Mais comment visualiser la ROI ?

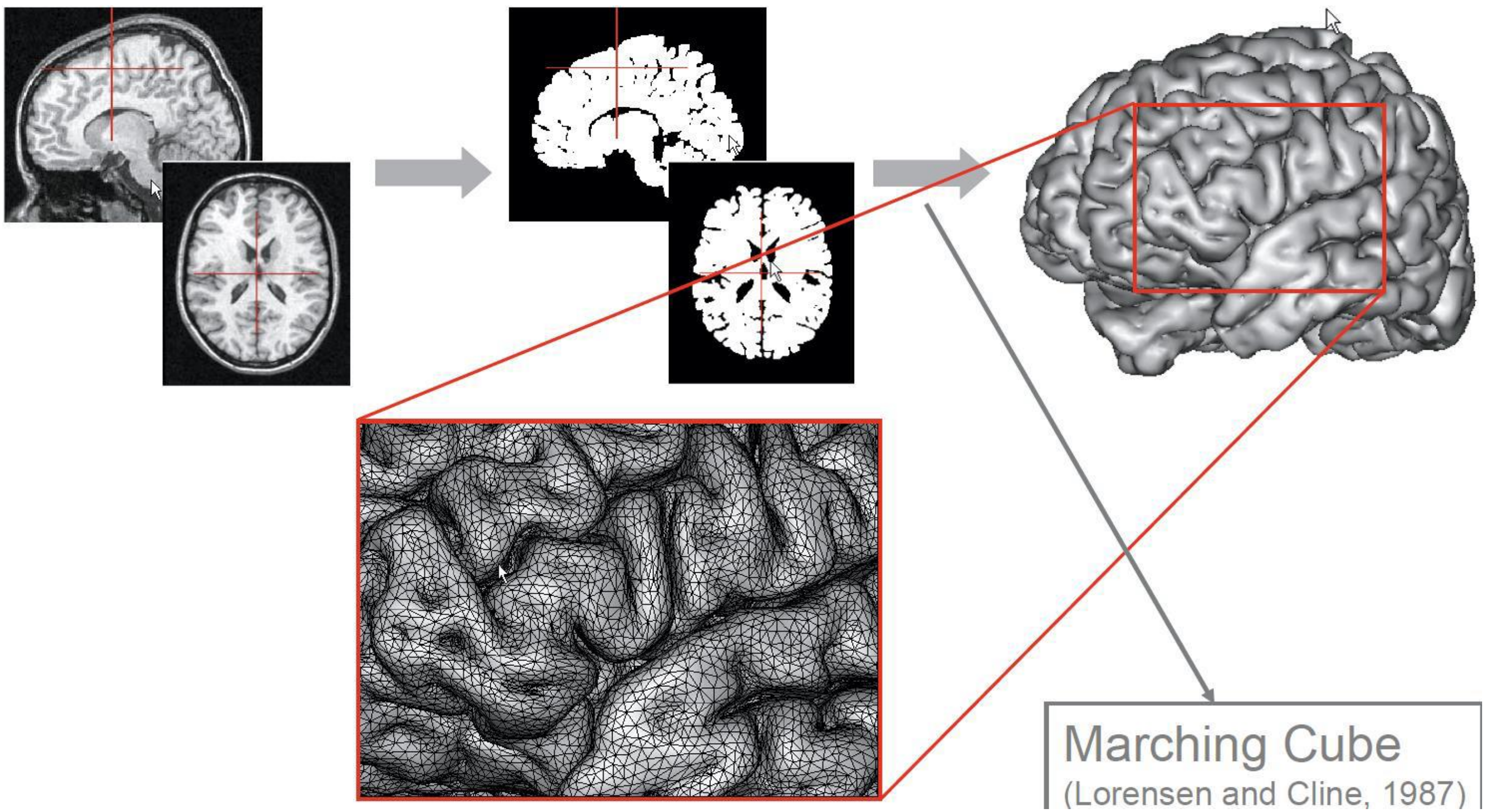
→ On a une ROI définie dans l'image par une région ou une frontière

→ Maillage 3D composé d'un ensemble de faces (triangles)



Mais comment visualiser la ROI ?

→ **Visualisation surfacique**



Obtenir un **maillage 3D** représentant la frontière de la ROI dans l'image segmentée

Extraction d’une iso-surface

- L’**iso-surface**(resp. iso-contour en 2D) est la surface (resp. le contour) qui «passe» par les voxels(resp. pixels) d’une intensité donnée.

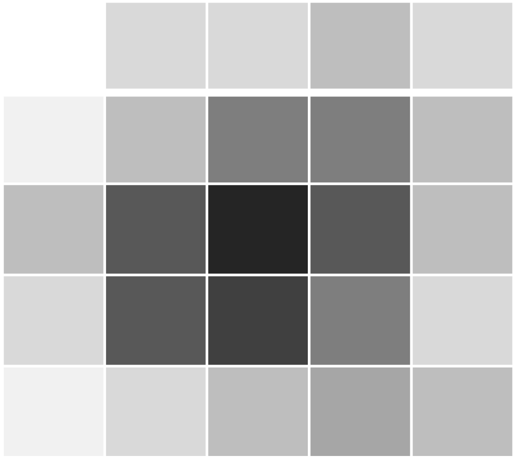
- Représentation implicite : I(x,y,z)=I0

- Surface (resp. contour) fermée

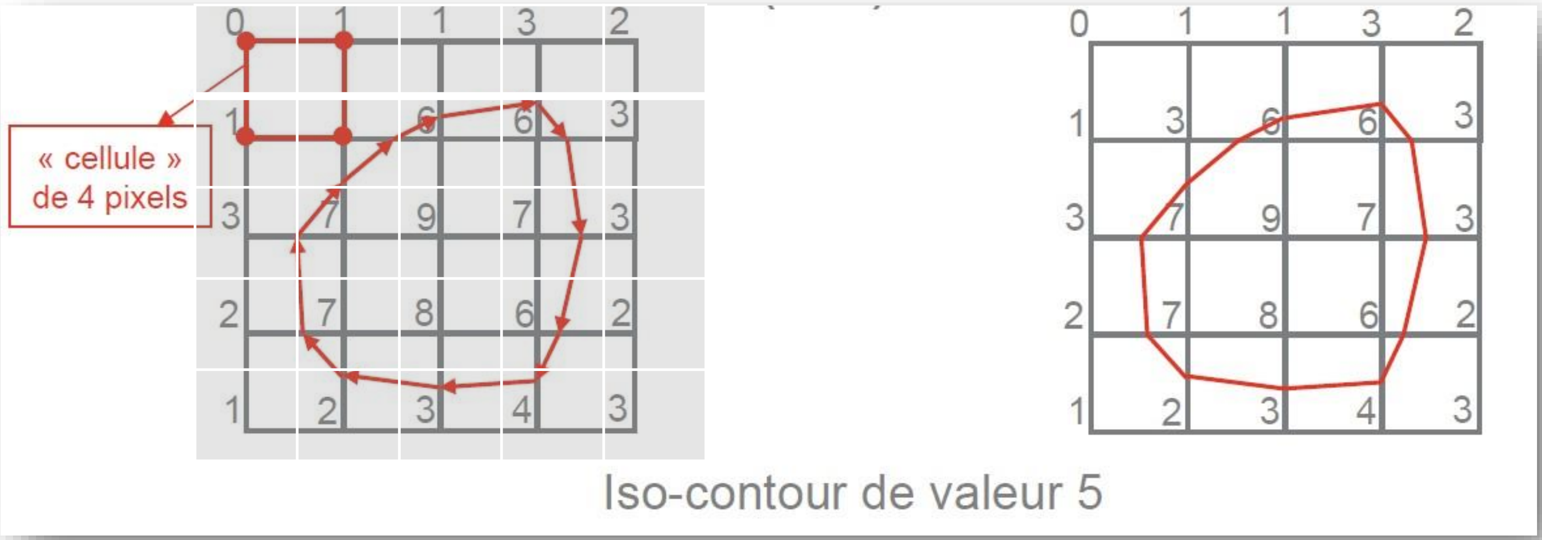
- Image segmentée (0/1 ou 0/255), I0 = 0.5 ou I0 = 128

  → maillage surfacique de la structure segmentée

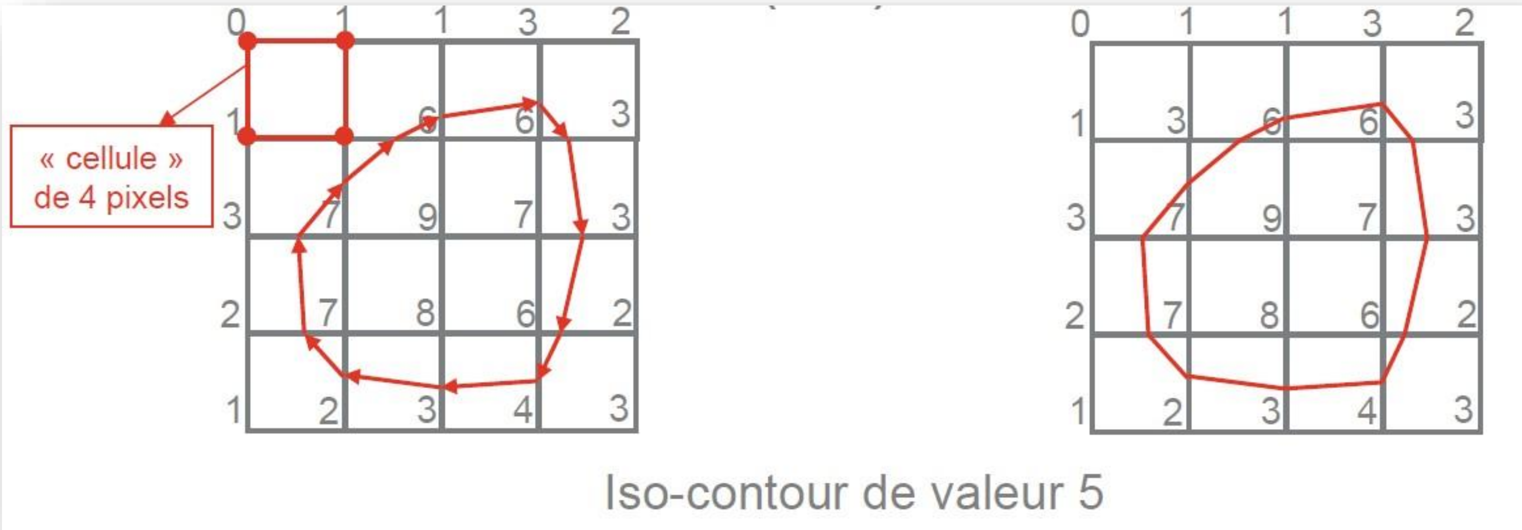
Etudions d’abord le problème en 2D !



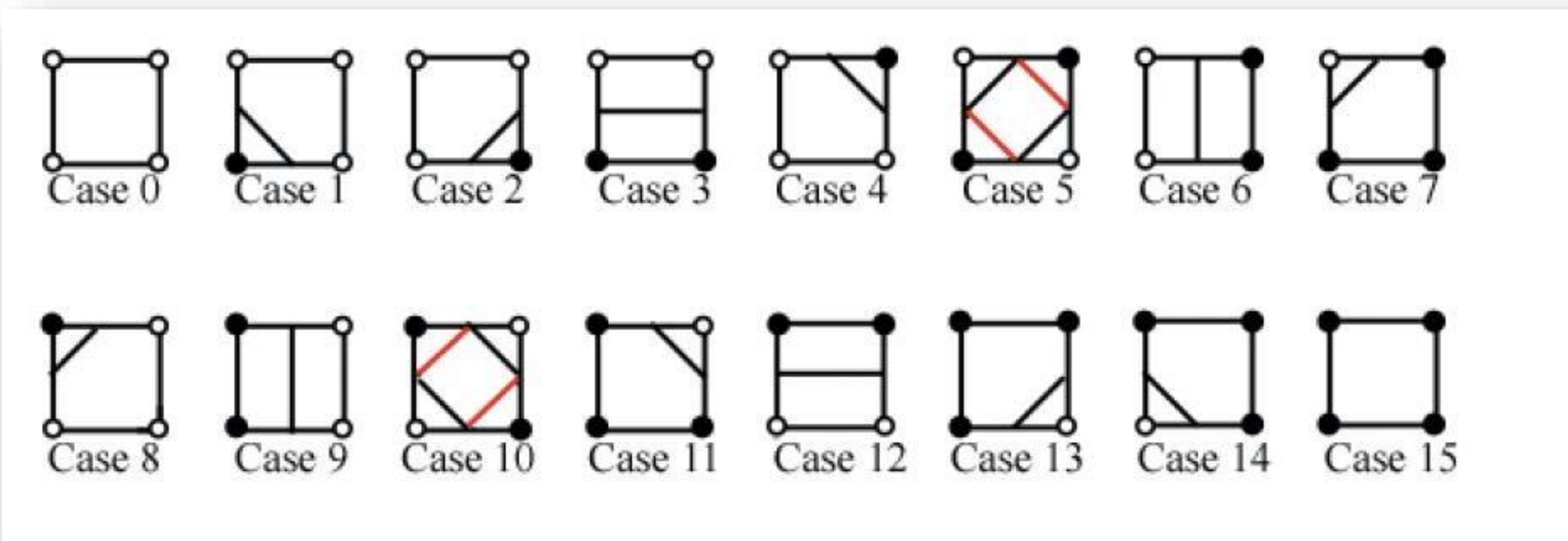
- Création de la grille 2D/3D reliant les centres des pixels/voxels
- Examen de toutes les cellules (= 4 pixels ou 8 voxels).
- Un sommet de la cellule est en dehors [$I(x,y) < I_0$] ou à l'intérieur [$I(x,y) > I_0$] du contour.

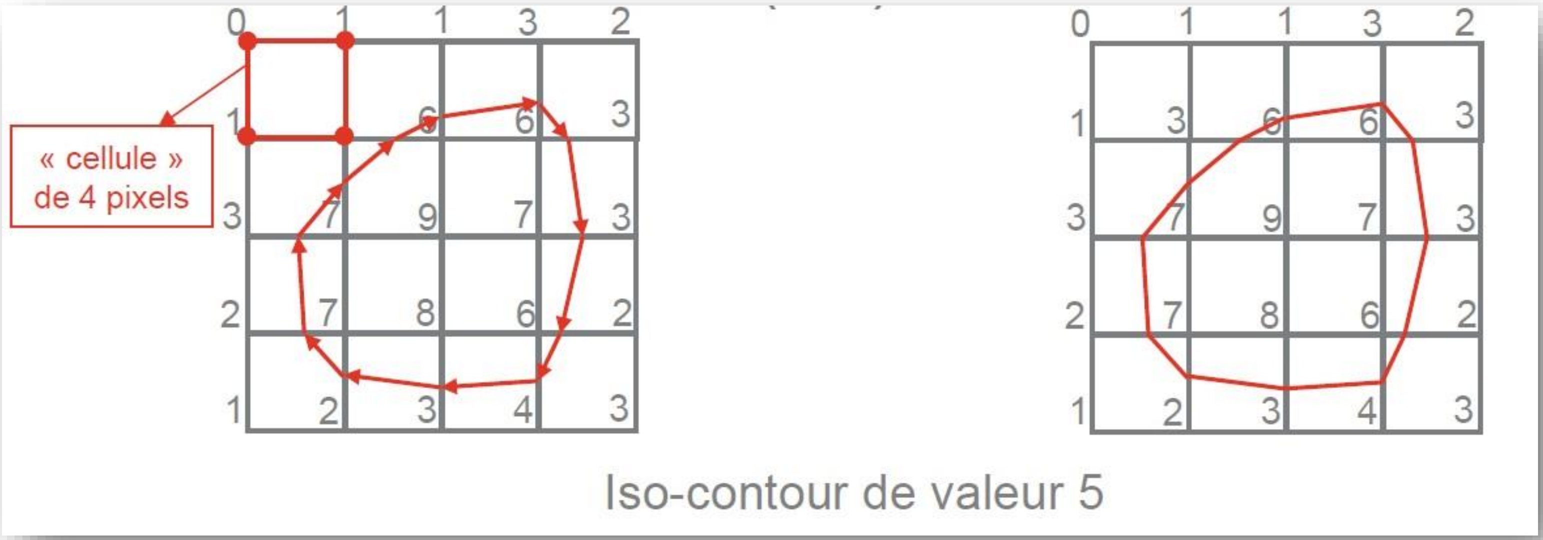


• Création de la grille 2D/3D reliant les centres des pixels/voxels
• Examen de toutes les cellules (= 4 pixels ou 8 voxels).
• Un sommet de la cellule est en dehors [$I(x,y) < I_0$] ou à l'intérieur [$I(x,y) > I_0$] du contour.

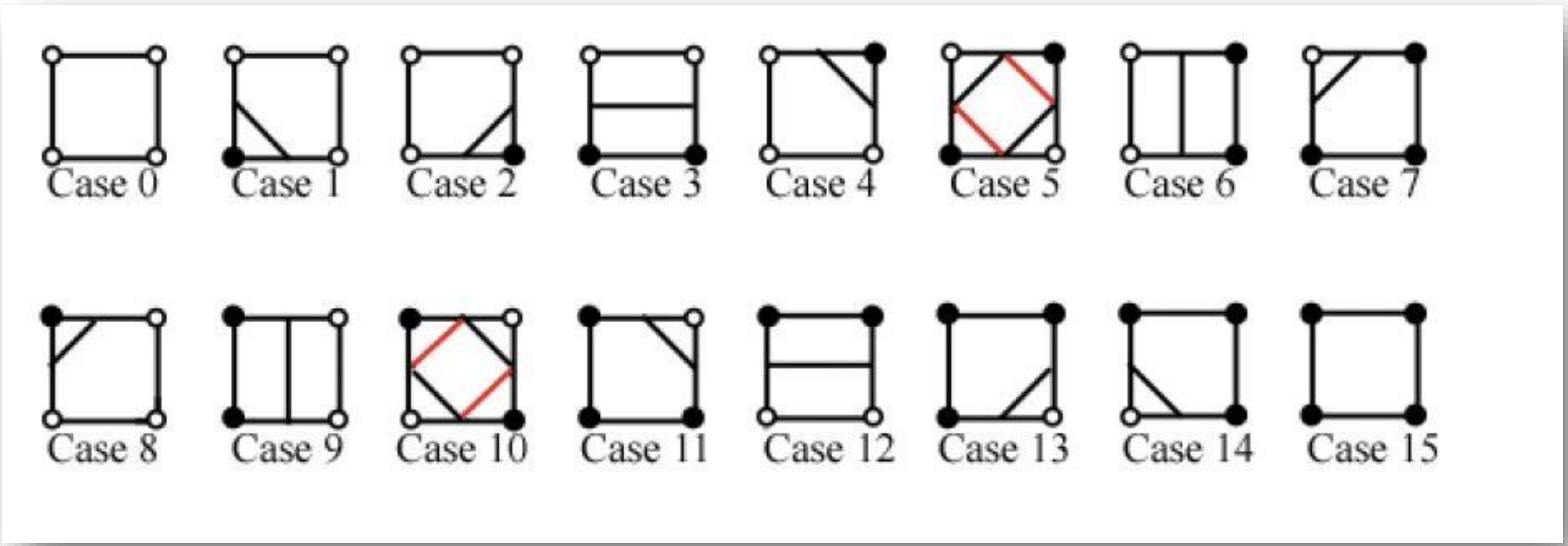


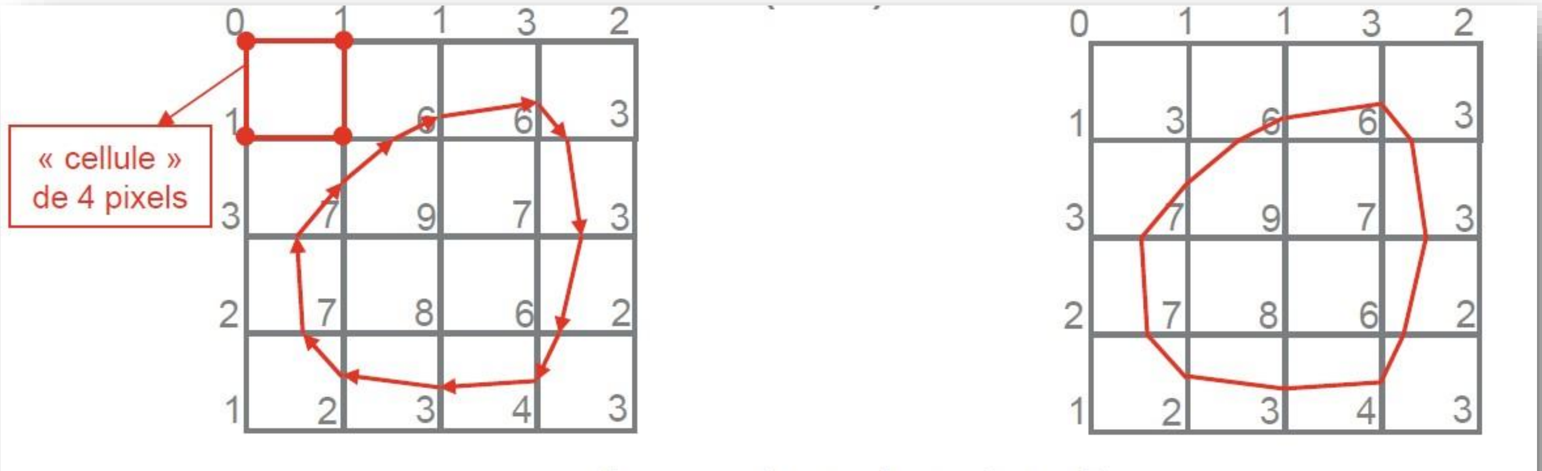
- Création de la grille 2D/3D reliant les centres des pixels/voxels
- Examen de toutes les cellules (= 4 pixels ou 8 voxels).
- Un sommet de la cellule est en dehors [$I(x,y) < I_0$] ou à l'intérieur [$I(x,y) > I_0$] du contour.





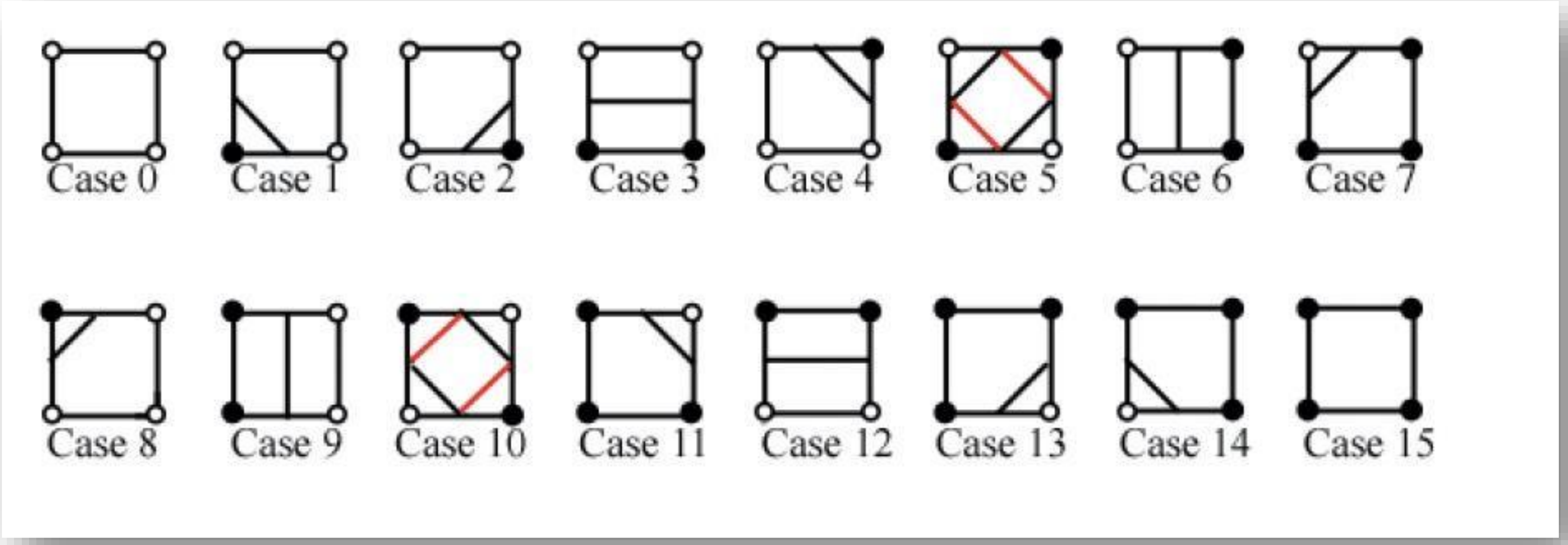
- Examen de toutes les cellules (= 4 pixels ou 8 voxels).
- Un sommet de la cellule est en dehors [$I(x,y)< I_0$] ou à l'intérieur [$I(x,y)> I_0$] du contour.
- Pour une cellule, un nombre fini (2^4=16) de configurations est possible.

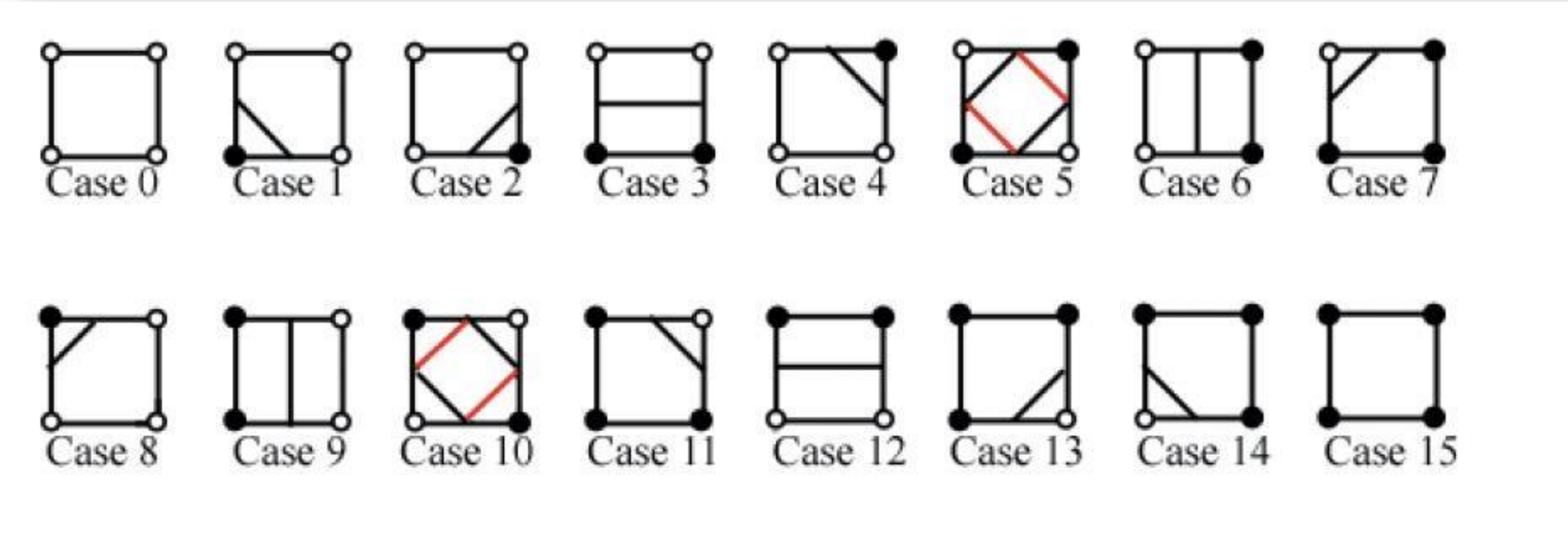




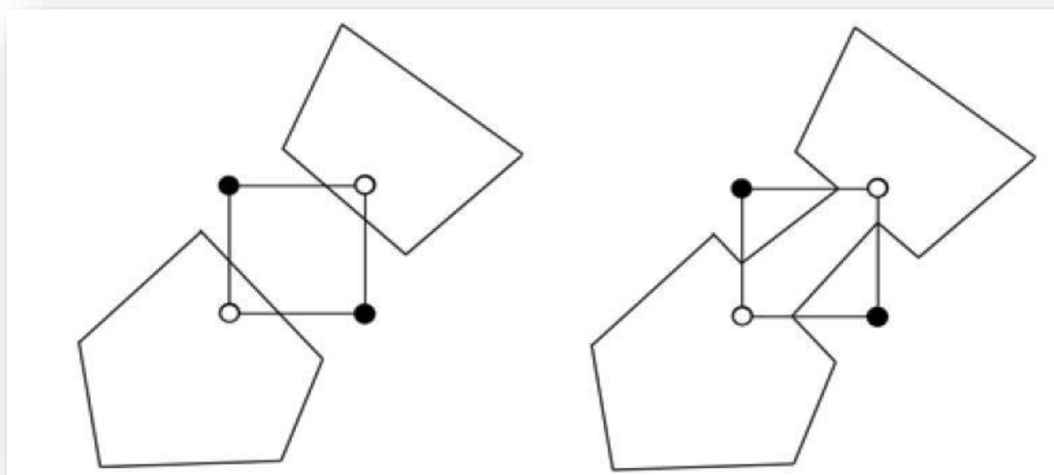
Iso-contour de valeur 5

- On peut alors définir un ou plusieurs segments correspondant au contour
- Qui vont se rabouter de cellule en cellule….
- Pour former une représentation discrète du contour

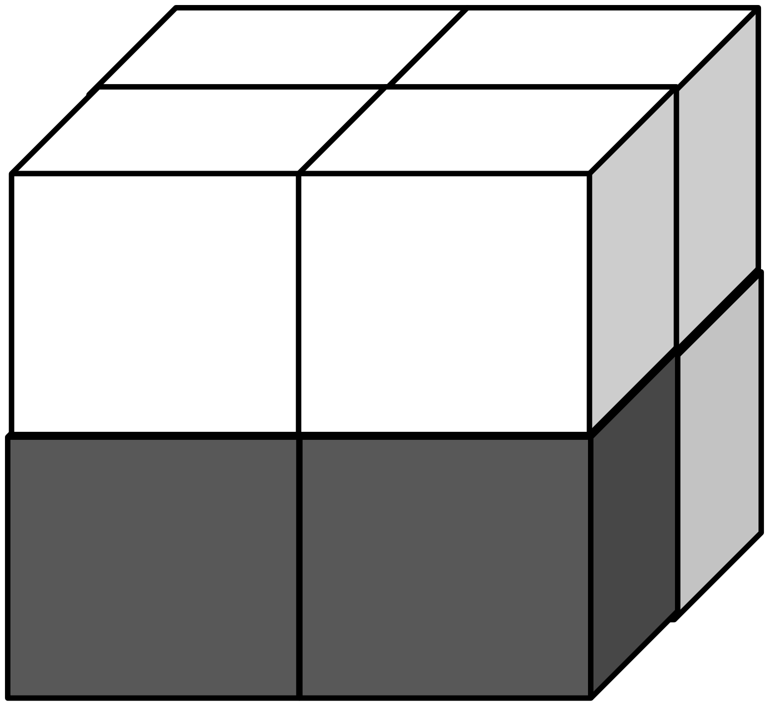




- D’un point de vue informatique, algorithme très facile à programmer.
- Il peut y avoir des ambiguïtés (case 5 / 10) mais le contour reste bien fermé.

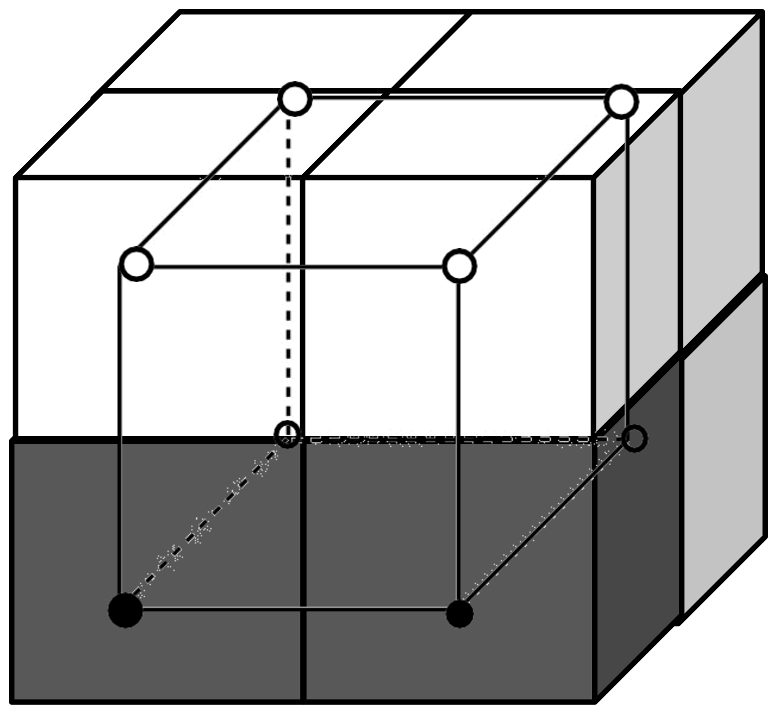


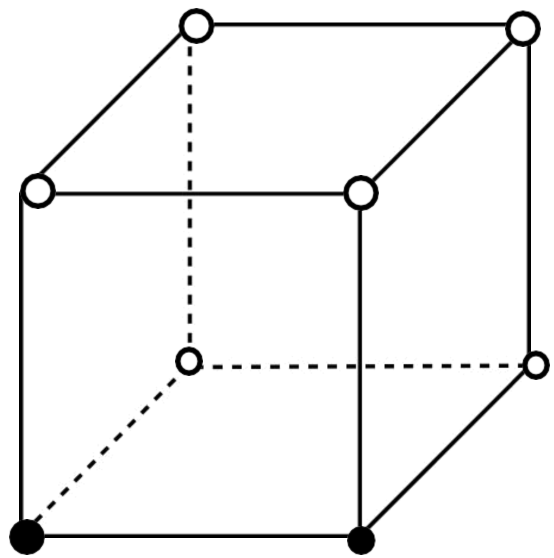
En 3 dimensions…

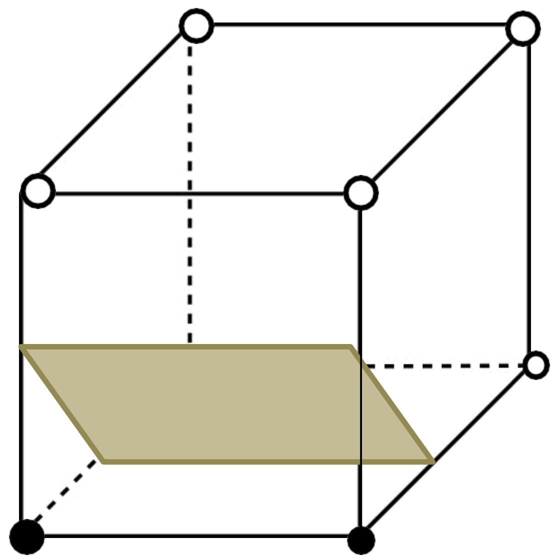


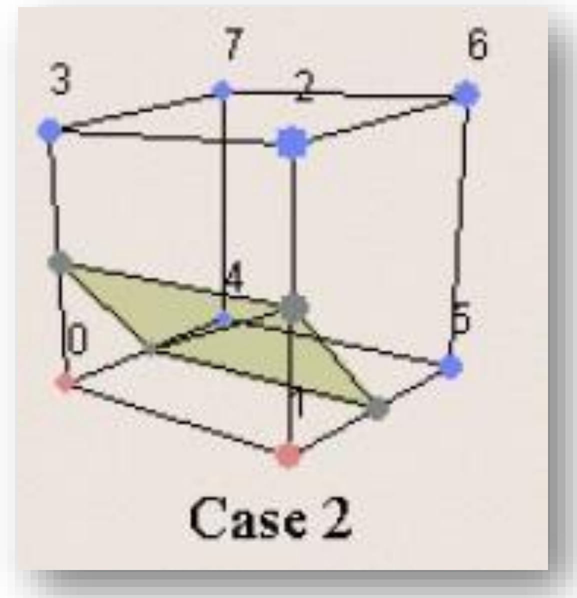
→ Algorithme du **Marching Cubes** qui va construire la surface par groupes de 2x2x2 voxels.

Lorensen, W. E.; Cline, Harvey E. (1987). "Marching cubes: A high resolution 3d surface construction algorithm". ACM Computer Graphics 21 (4): 163–169.







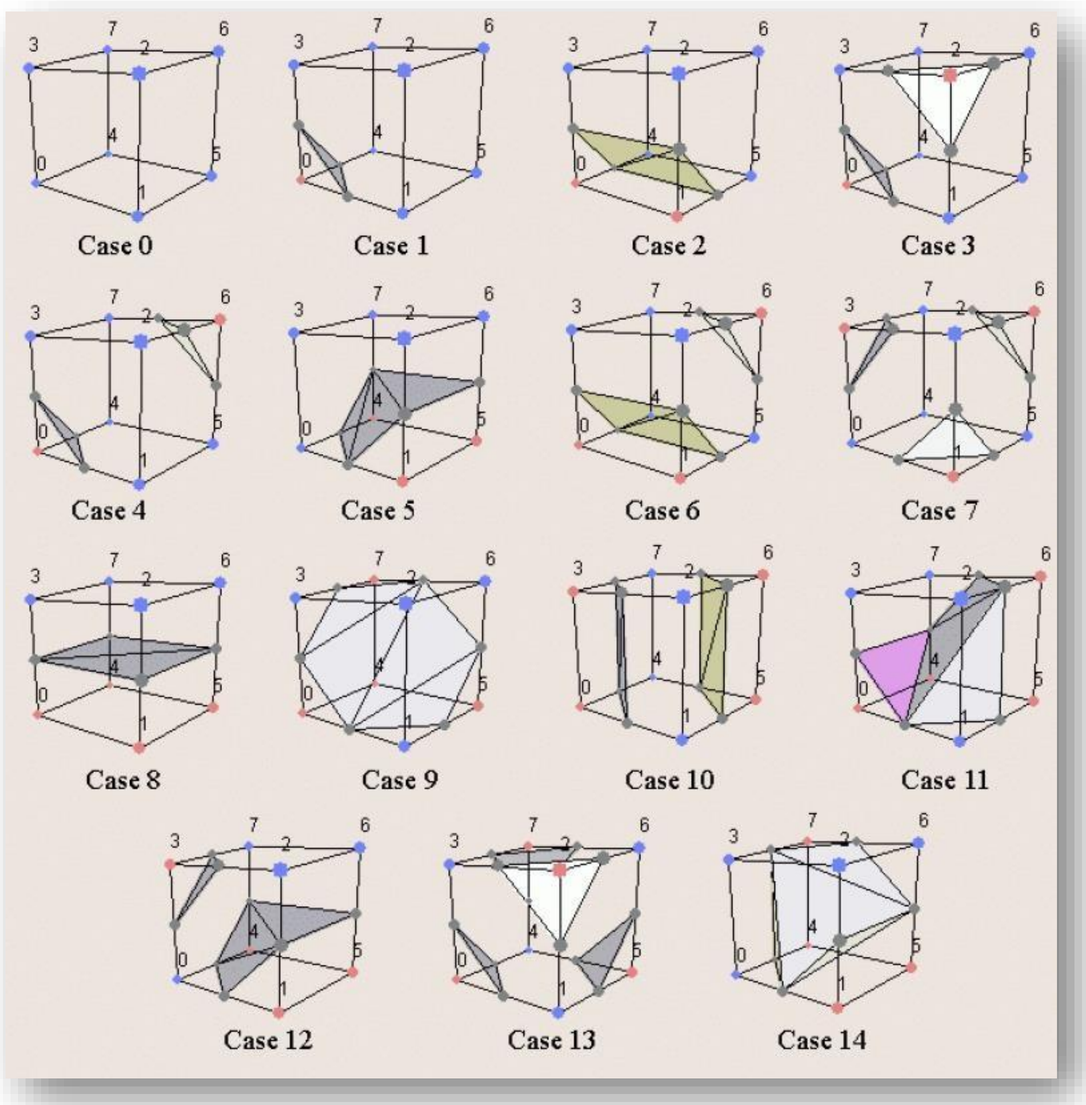


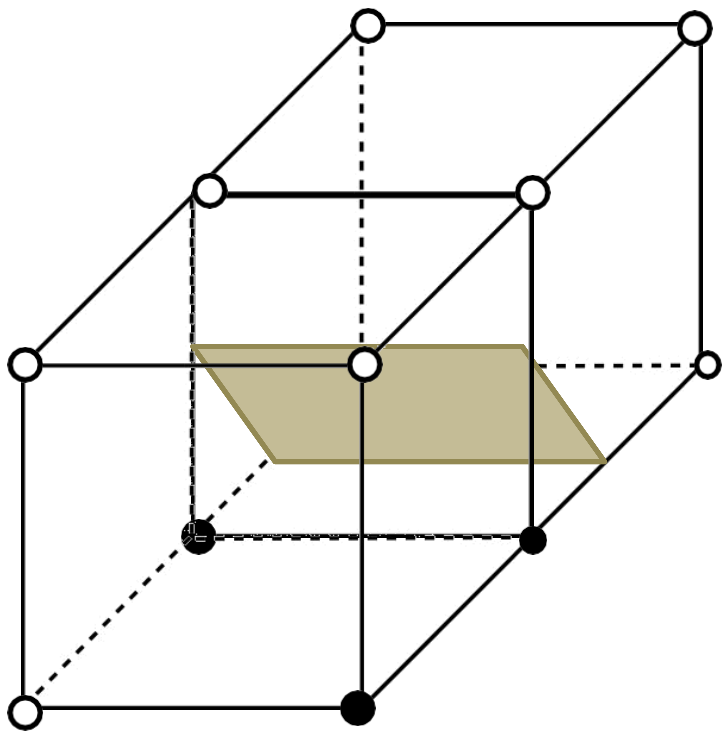
•Toutes les configurations (33) ont été listées.

Le problème se généralise en 3D avec $2^8 = 256$ configurations possibles.

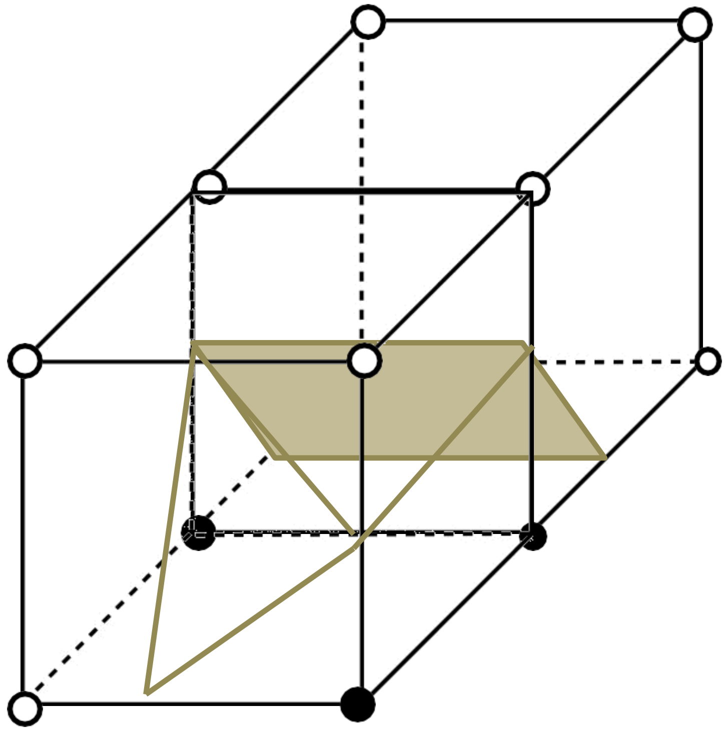
→ 15 configurations de base, les autres par symétrie

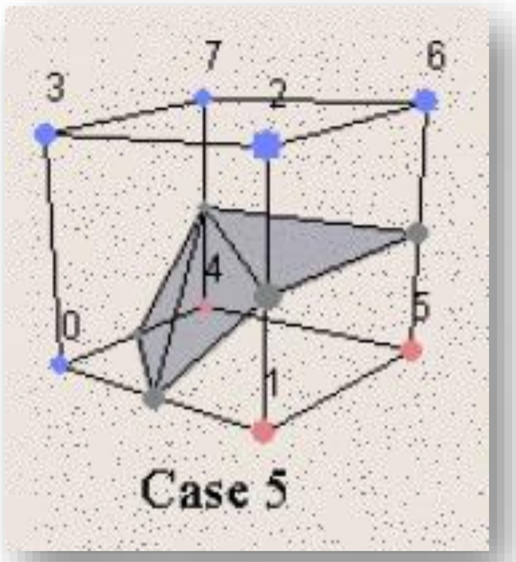
Et on peut construire un ou plusieurs polygones qui représentent l'isosurface à l'intérieur de la cellule..



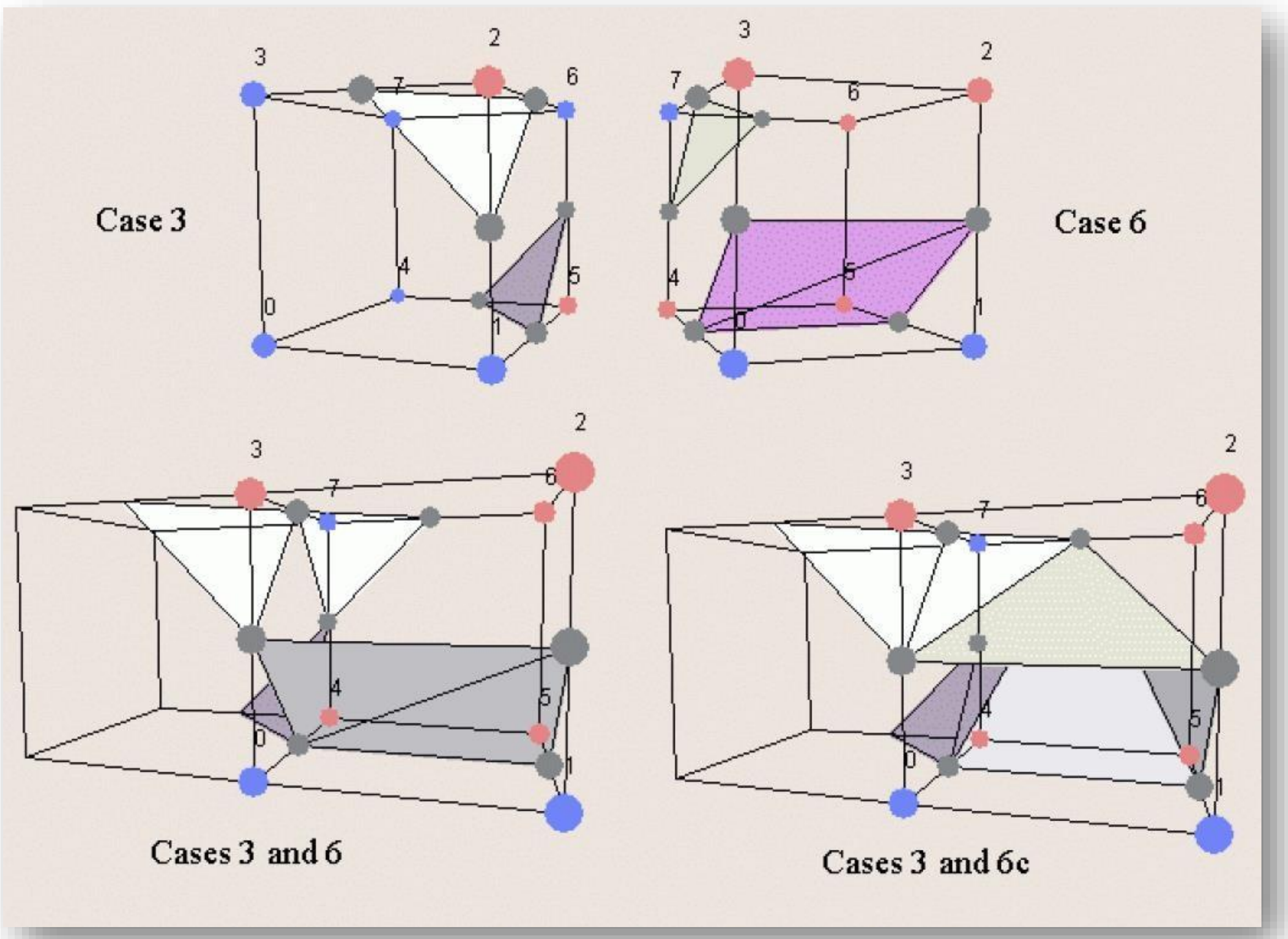


• Nouveau bloc de 2x2x2 voxels

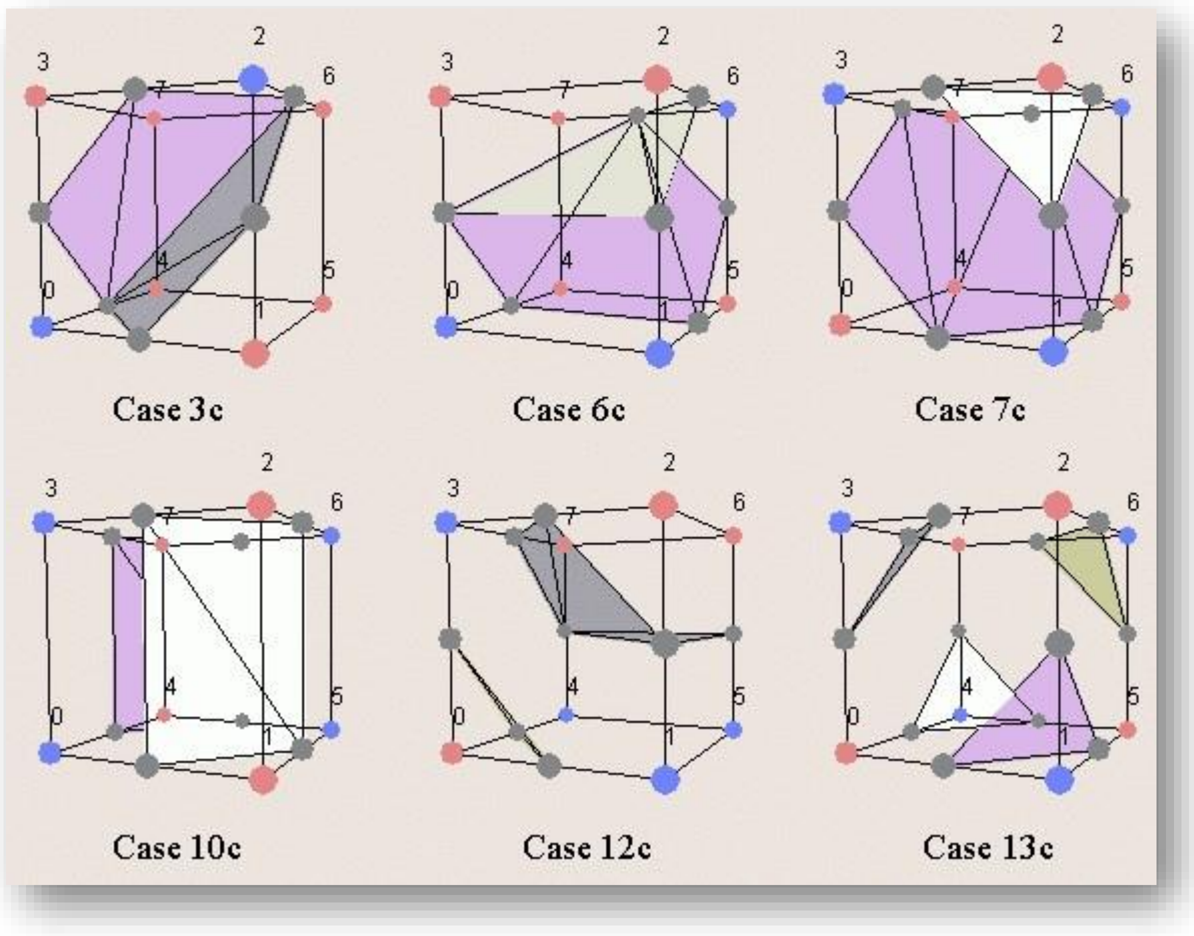



3
7
2
6
4
0
5
1
Case 5

Le problème est plus complexe qu'en 2D : le choix d'une configuration dans une cellule n'est pas indépendant des autres cellules → un mauvais choix peut générer des trous dans la surface.



6 configurations complémentaires sont ajoutées pour résoudre les cas problématiques

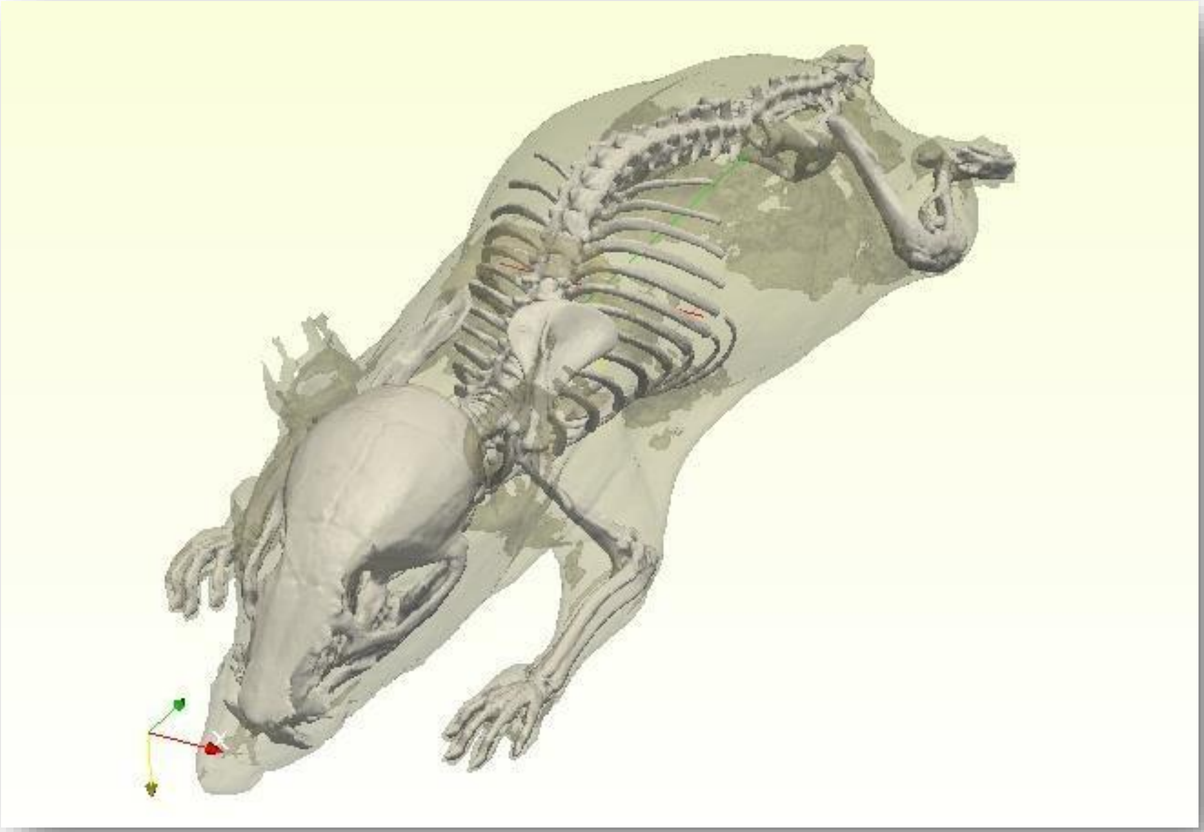


https://www.youtube.com/watch?v=B_xk71YopsA

https://www.youtube.com/watch?v=Xz3txaYgtKA

Visualisation surfacique

→ Une ou plusieurs surfaces **fermées** (sauf sur les bords de l'image 3D)
→ Visualisation avec des logiciels d'infographie ou de CAO
→ Effets graphiques (ombrage, transparence…)
→ Simulation possible (découpe par exemple)



Sources

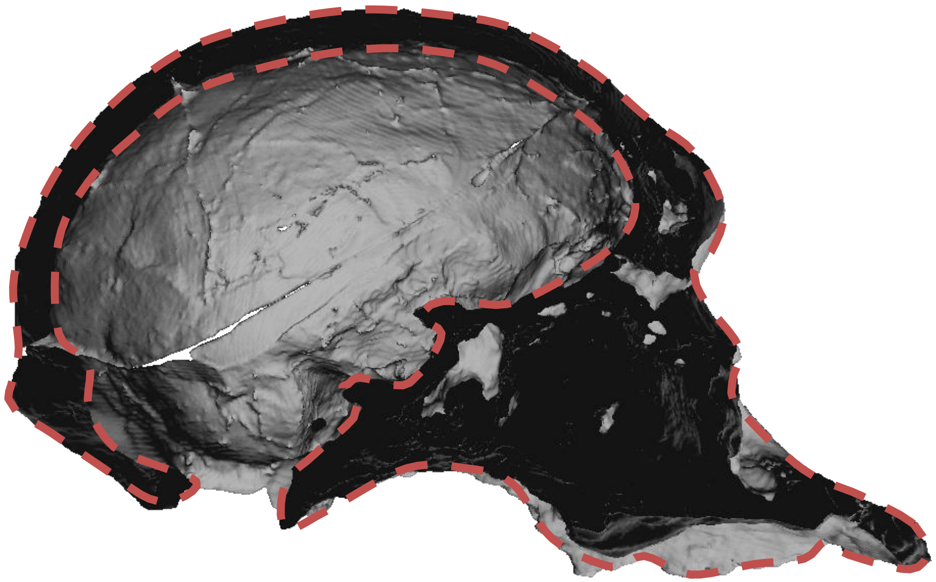
- Segmentation d’images médicales, H. Delingette
  http://lsiit-miv.u-strasbg.fr/ecoleTIM/download/cours-TIM-CNRS-Delingette-2008.pdf

- Wikipedia

- Guide to Medical Image Analysis - Methods and Algorithms
  Series: Advances in Computer Vision and Pattern Recognition
  Toennies, Klaus D.
  2012, 2012, XX, 468 p. 327 illus.

Principle of the method

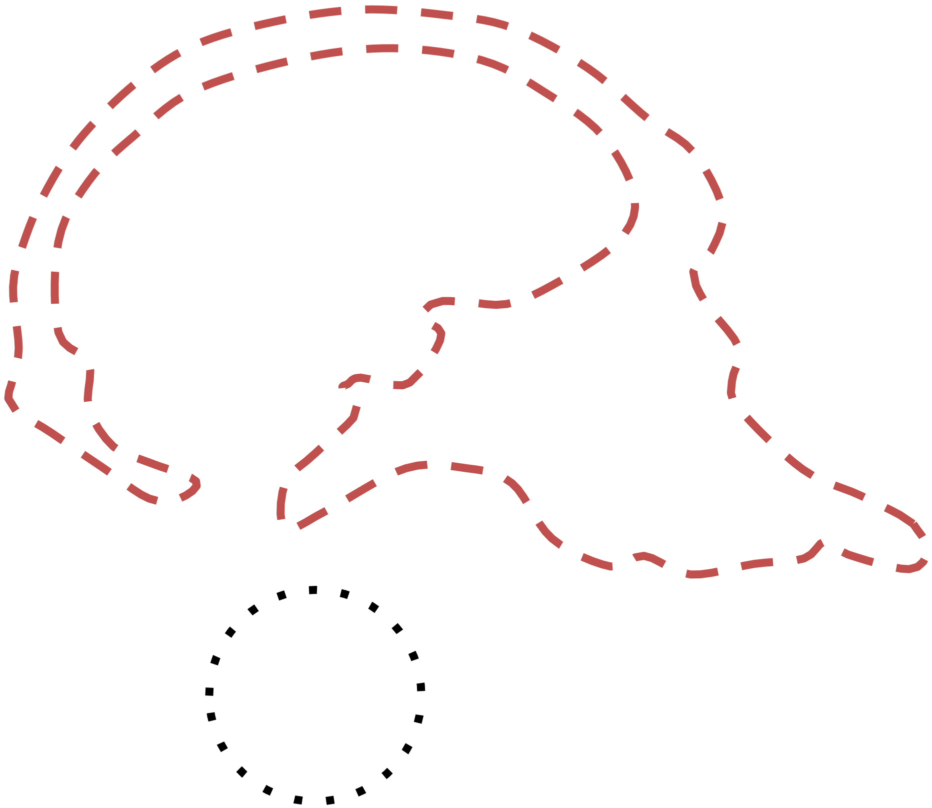
- The data are represented by a set of 3D points



Principle of the method

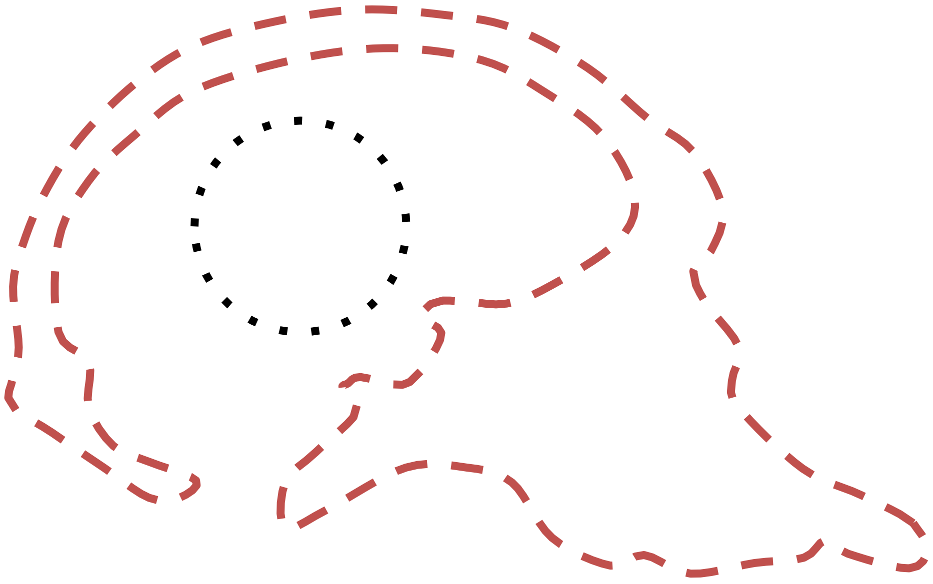
- The data are represented by a set of 3D points

- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)



Principle of the method

- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data



Principle of the method

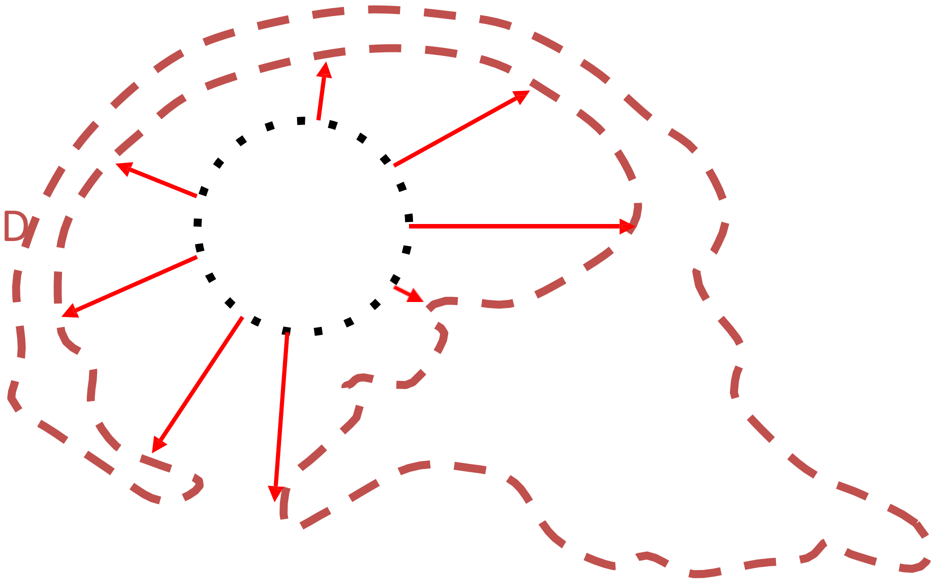
- The data are represented by a set of 3D points

- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)

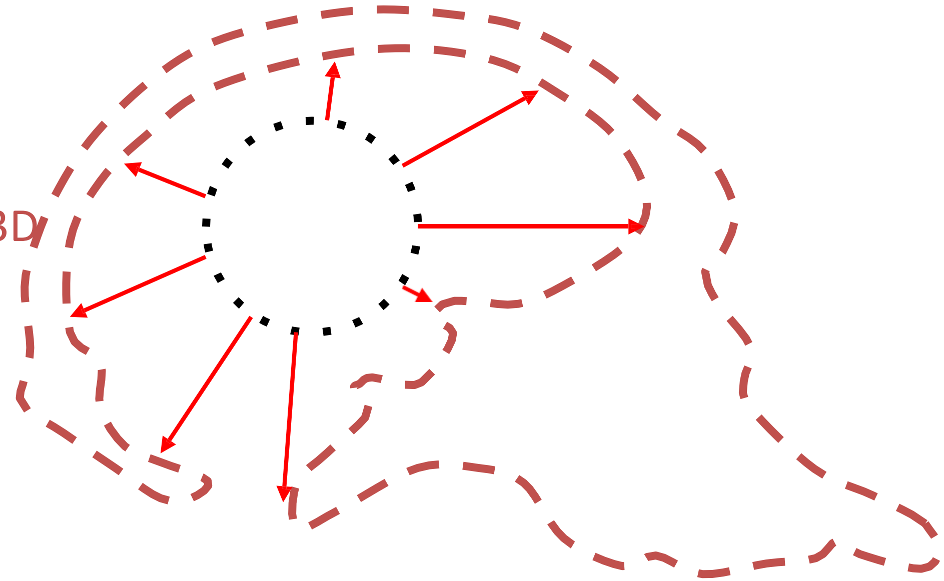
- The surface mesh is initially positioned "in the middle" of the data

- This surface will deform under the influence of:

  - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P}_i$ towards the data
  - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)



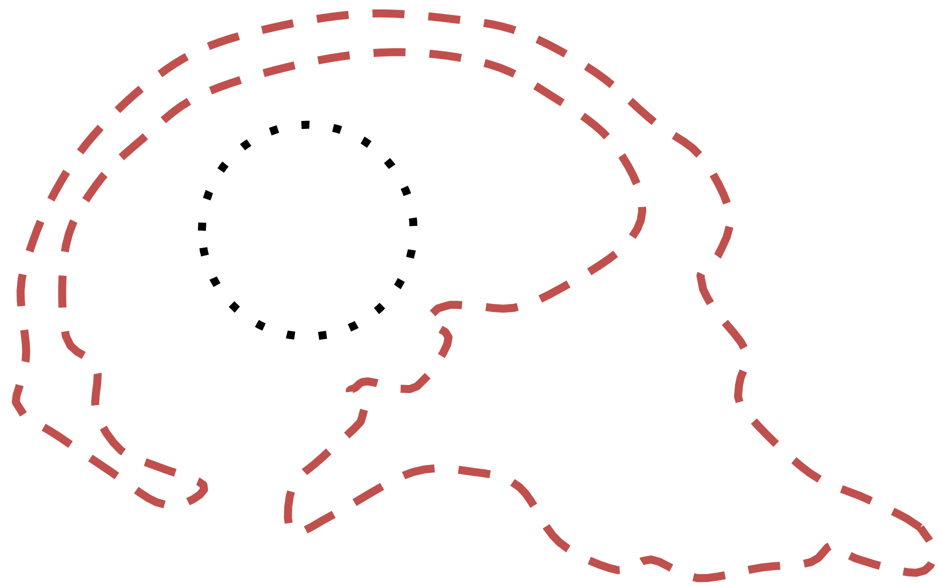
Principle of the method



- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P_i}$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
    - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P_i}$ towards the data
    - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices $\mathbf{P_i}$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

Principle of the method

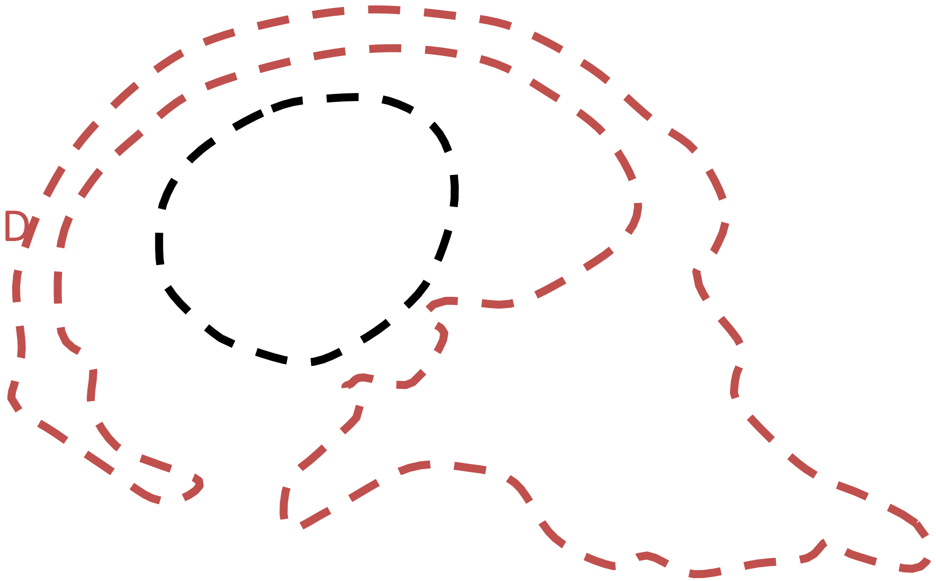


- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
  - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P}_i$ towards the data
  - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices $\mathbf{P}_i$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

- Iterate the process until the vertices $\mathbf{P}_i$ do not move anymore.

Principle of the method

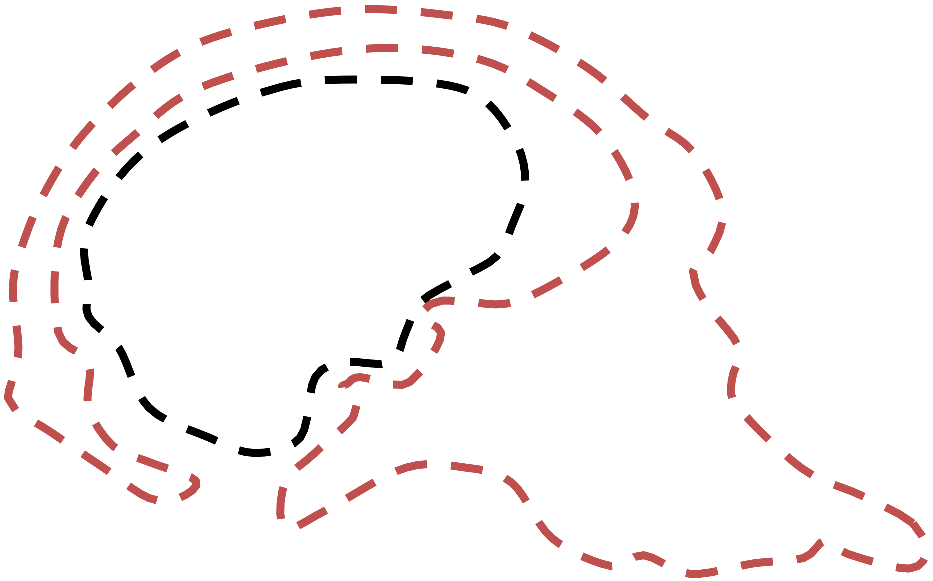


- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices **P**$_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
  - an external force **F**$_{ext}$ which attracts the vertices **P**$_i$ towards the data
  - an internal force **F**$_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices **P**$_i$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

- Iterate the process until the vertices **P**$_i$ do not move anymore.

Principle of the method

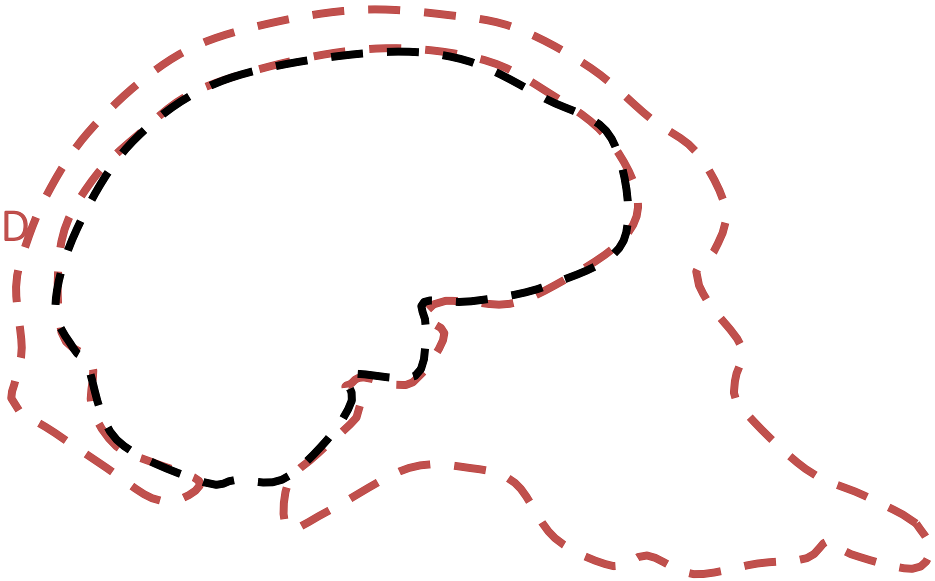


- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
  - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P}_i$ towards the data
  - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices $\mathbf{P}_i$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

- Iterate the process until the vertices $\mathbf{P}_i$ do not move anymore.

Principle of the method

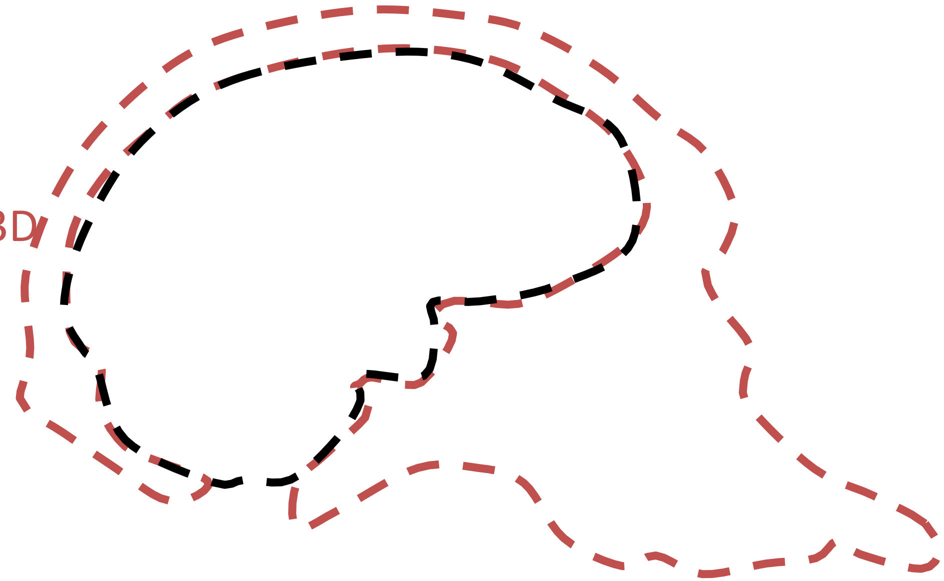


- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
  - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P}_i$ towards the data
  - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices $\mathbf{P}_i$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

- Iterate the process until the vertices $\mathbf{P}_i$ do not move anymore.

Principle of the method



- The data are represented by a set of 3D points
- Let a simple closed surface mesh composed of 3D vertices $\mathbf{P}_i$ (e.g. a sphere)
- The surface mesh is initially positioned "in the middle" of the data
- This surface will deform under the influence of:
  - an external force $\mathbf{F}_{ext}$ which attracts the vertices $\mathbf{P}_i$ towards the data
  - an internal force $\mathbf{F}_{int}$ which tends to keep the surface smooth (e.g. curvature continuity)
- At time t, all the vertices $\mathbf{P}_i$ follow the evolution law:

$$P_i^{t+1} = P_i^t + (1-\gamma)(P_i^t - P_i^{t-1}) + \alpha_i \mathbf{F}_{int} + \beta_i \mathbf{F}_{ext}$$

- Iterate the process until the vertices $\mathbf{P}_i$ do not move anymore.
- Eventually, add more vertices in the mesh when the distance between the existing vertices becomes too large in order to recover the details.