

## TP2 - Modèles Déformables

### 1 Paramètres

Pour chaque méthode, fournissez des explications détaillées sur :

- l'interprétation de chaque paramètre et son rôle,
- son influence sur le résultat de la segmentation lorsque sa valeur varie.

#### 1.1 Contours actifs paramétriques

Les "contours actifs," également appelés "snakes," sont des courbes paramétriques déformables utilisées en vision par ordinateur et en traitement d'images pour des tâches telles que la segmentation d'objets, la détection de contours et la modélisation de formes. Ces courbes actives peuvent changer leur forme pour s'adapter aux contours d'objets dans une image grâce à l'utilisation d'une énergie interne et externe. L'énergie interne favorise la régularité de la courbe, tandis que l'énergie externe attire la courbe vers les bords ou les contours de l'objet dans l'image. Les contours actifs sont très utiles pour segmenter des objets dans des images et sont utilisés dans diverses applications, y compris l'analyse d'images médicales, la retouche d'images et la conception assistée par ordinateur.

Dans ce TP, la fonction `segmentation.active_contour` [1] de la bibliothèque scikit-image est utilisée pour effectuer la segmentation d'une image en utilisant des contours actifs (snakes). Nous allons analyser le rôle de chaque paramètre :

- **image** : Il s'agit de l'image en niveaux de gris sur laquelle on souhaite effectuer la segmentation à l'aide du contour actif. L'algorithme va chercher à ajuster le snake pour qu'il corresponde aux caractéristiques de cette image (figure 1).
- **snake** : Il s'agit du contour actif initial que l'on fournit. C'est un tableau NumPy qui contient les coordonnées des points du contour initial (figure 2).
- **alpha** : C'est un paramètre qui contrôle l'énergie interne du snake, influençant sa rigidité (figure 3).
- **beta** : Il s'agit d'un paramètre qui contrôle l'énergie externe du snake, influençant comment le snake est attiré par les bords de l'objet dans l'image (figure 4).
- **w\_line** : C'est un paramètre qui permet de contrôler le poids de l'énergie interne du snake par rapport à l'énergie externe (figure 5).
- **w\_edge** : C'est un paramètre qui contrôle le poids de l'énergie externe du snake par rapport à l'énergie interne (figure 6).
- **gamma** : C'est un paramètre qui contrôle l'importance de l'énergie externe dans la détection de bords (figure 7).

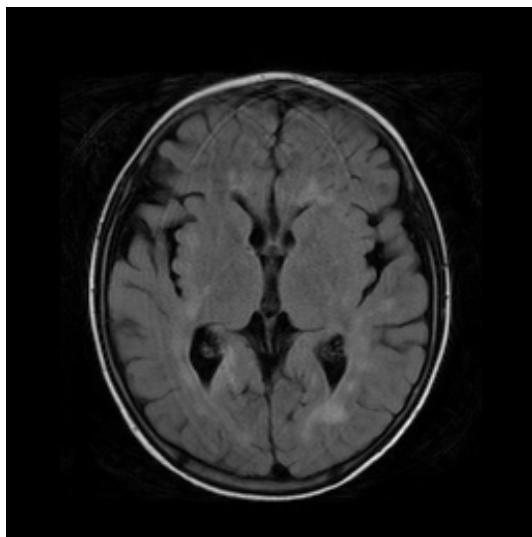


Figure 1: Image d'origine

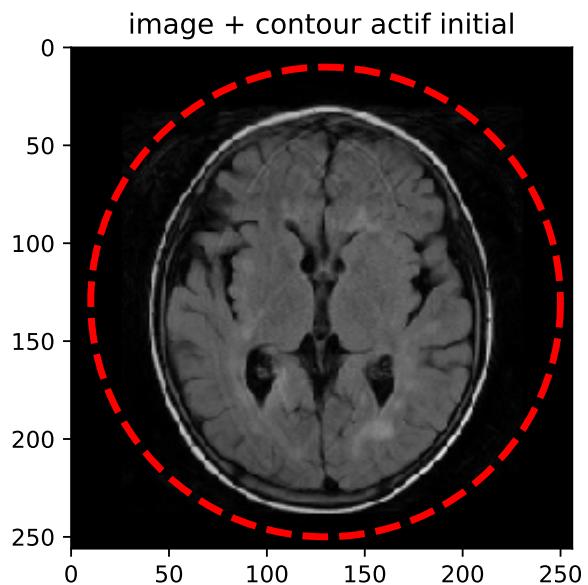


Figure 2: Dans notre cas, on a spécifié manuellement le contour actif initial.

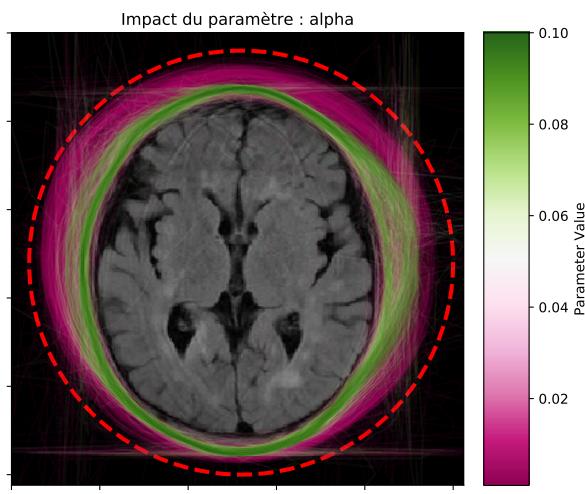


Figure 3: Une valeur plus élevée d'alpha rend le snake plus rigide, tandis qu'une valeur plus faible le rends plus flexible

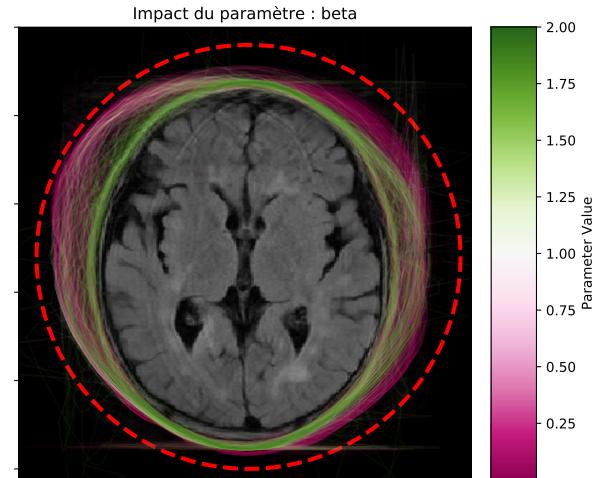


Figure 4: Une valeur plus élevée de beta augmente l'attraction aux bords.

- **max\_px\_move** : C'est la distance maximale en pixels de laquelle chaque point du snake peut se déplacer lors de chaque itération. Il limite la vitesse de convergence du snake (figure 8).
- **max\_num\_iter** : C'est le nombre maximum d'itérations que l'algorithme effectuera pour ajuster le snake. Une fois que ce nombre est atteint, l'algorithme s'arrête, que le snake ait convergé ou non (figure 9).

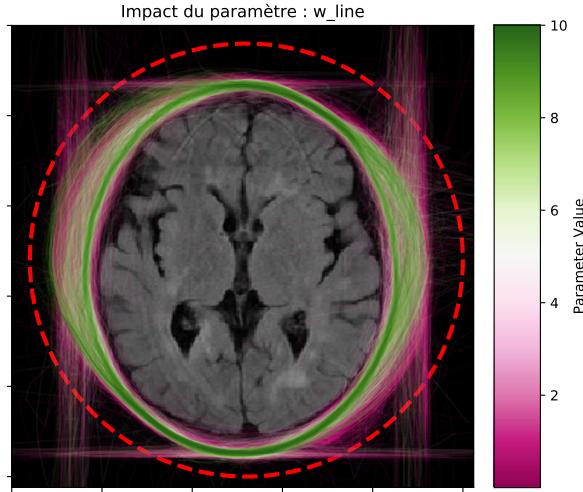


Figure 5: Il influence la balance entre la régularité du contour (valeurs élevées) et son ajustement aux bords de l'objet (petites valeurs).

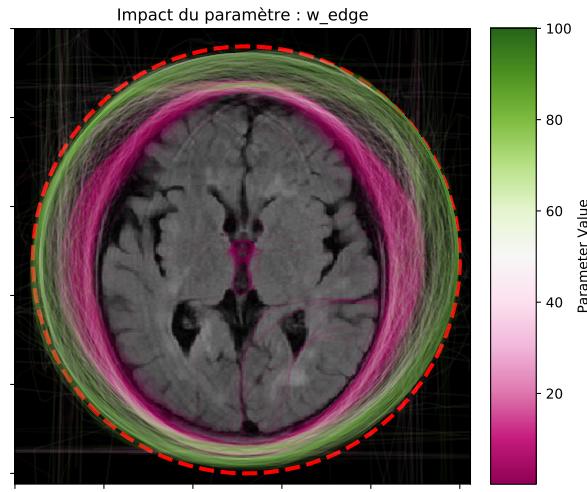


Figure 6: Ce paramètre influence à quel point le snake est attiré par les bords de l'objet (petites valeurs) par rapport à sa régularité (grandes valeurs).

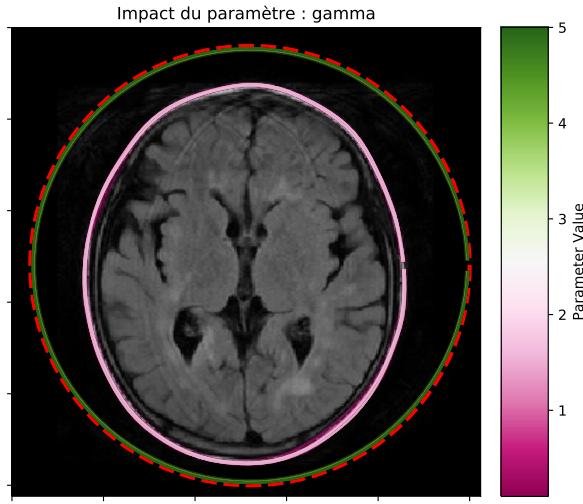


Figure 7: Une valeur plus élevée de gamma rend le snake plus sensible aux changements d'intensité des pixels.

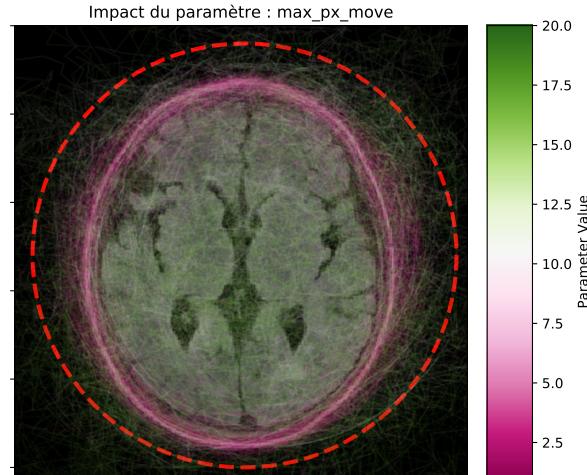


Figure 8: Une valeur élevée semble empêcher la convergence du snake.

- **convergence** : C'est un seuil qui détermine à quel point le snake est considéré comme convergé. L'algorithme s'arrête si la variation du snake entre deux itérations consécutives est inférieure à ce seuil (figure 10).
- **boundary\_condition** : Il spécifie les conditions de bord pour le snake. On peut choisir entre "free" (libre), "fixed" (fixe), ou "periodic" (périodique) pour définir comment le snake interagit avec les bords de l'image (figure 11).

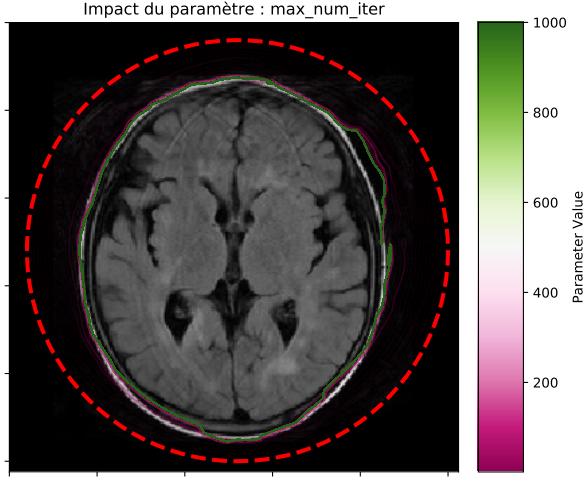


Figure 9: Ici, on a testé pour des valeurs allant de 1 à 1000, et on obtient le même résultat à chaque fois. Apparemment, le snake converge très rapidement pour cette image.

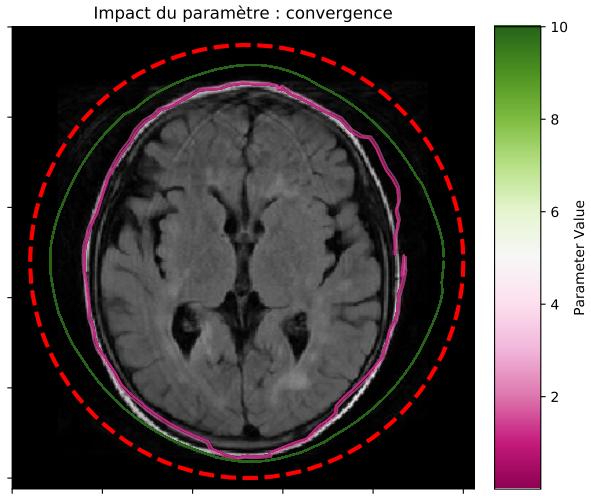


Figure 10: Logiquement, pour un seuil élevé, le snake n'a pas le temps de converger.

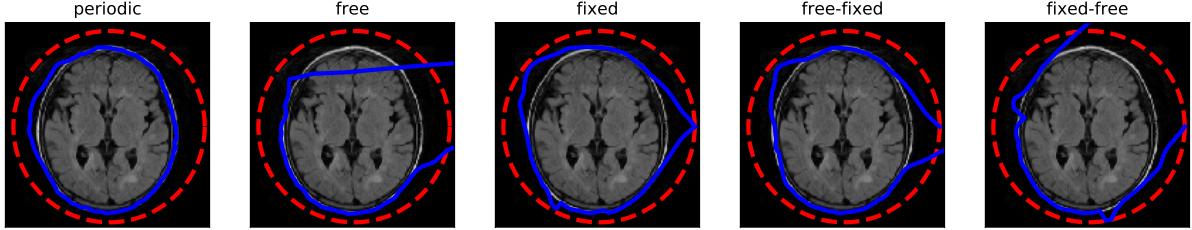


Figure 11: Le mode 'periodic' relie les deux extrémités du snake, le mode 'fixed' maintient les points d'extrémité en place, et le mode 'free' permet une libre mouvance des extrémités. Les modes 'fixed' et 'free' peuvent être combinés en utilisant 'fixed-free' ou 'free-fixed' [1]

## 1.2 Représentation implicite (Chan & Vese method)

Le "méthode Chan-Vese" est une technique populaire de segmentation d'images qui utilise des méthodes de surfaces de niveau (level sets) pour séparer une image en deux régions : avant-plan et arrière-plan. Cette méthode est basée sur le concept d'homogénéité de région, supposant que les pixels dans chaque région partagent des caractéristiques similaires, comme des valeurs d'intensité proches. Elle cherche à minimiser une fonction d'énergie qui prend en compte l'ajustement des données par rapport à un modèle d'intensité constant, la régularité de la forme de segmentation, et la taille des régions. En utilisant cette méthode, on peut segmenter une image en identifiant des régions homogènes en termes d'intensité.

La fonction `skimage.segmentation.chan_vese` [2] dans la bibliothèque scikit-image est utilisée pour effectuer la segmentation d'une image en utilisant la méthode de segmenta-

tion Chan-Vese. Voici une explication détaillée de chaque paramètre de cette fonction :

- **image** : Il s'agit de l'image en niveaux de gris que l'on souhaite segmenter. La méthode Chan-Vese tente de séparer cette image en deux régions distinctes (avant-plan et arrière-plan) en fonction des caractéristiques d'intensité.
- **mu**: Il s'agit du terme de régularisation qui contrôle la régularité du contour de séparation. Une valeur plus élevée de mu favorisera des contours plus lisses, tandis qu'une valeur plus faible permettra des contours plus irréguliers (figure 12).

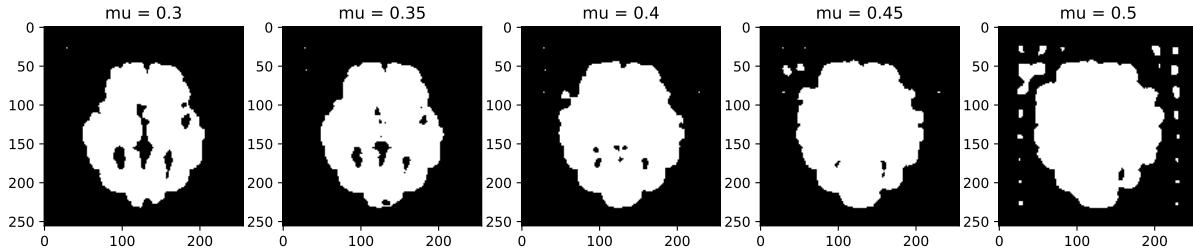


Figure 12: Impact du paramètre mu

- **lambda1** : C'est le coefficient d'ajustement de données pour la première région (avant-plan). Il mesure à quel point la région avant-plan est homogène en termes d'intensité (figure 13).

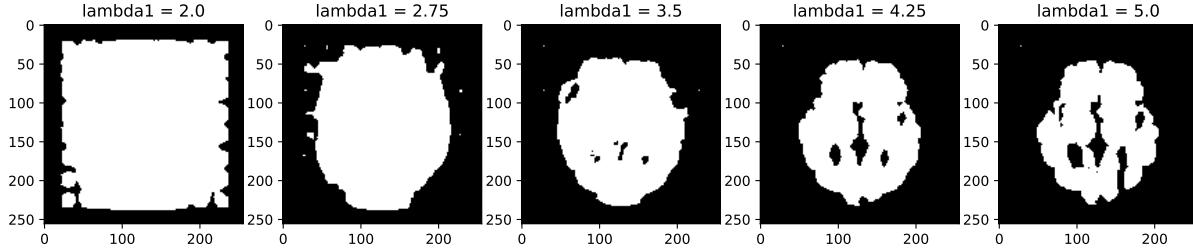
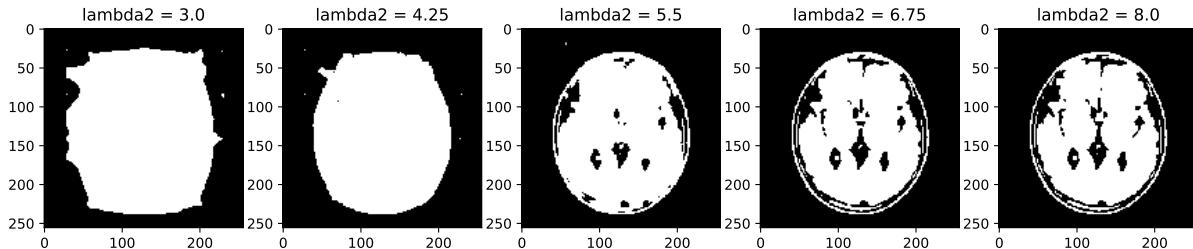
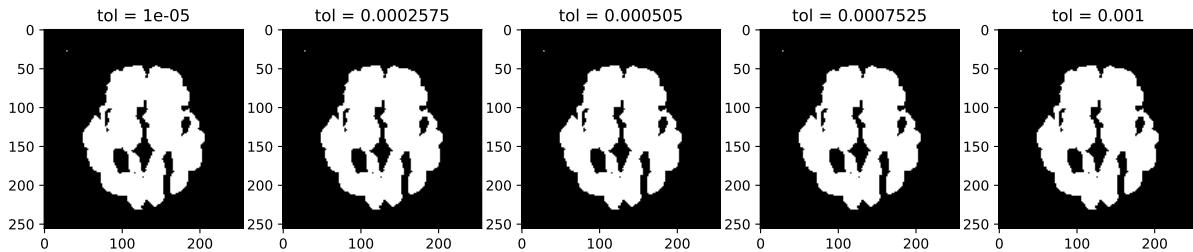
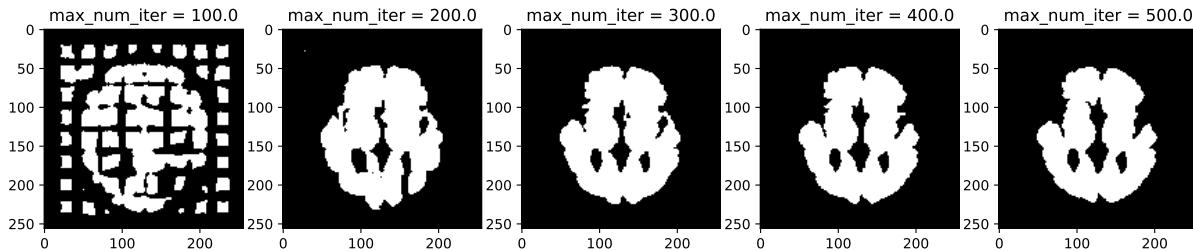
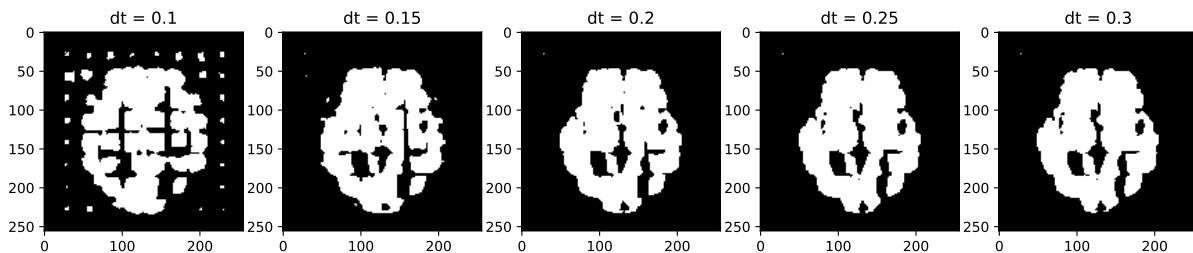


Figure 13: Impact du paramètre lambda1

- **lambda2** : C'est le coefficient d'ajustement de données pour la deuxième région (arrière-plan). Il mesure l'homogénéité de l'intensité de l'arrière-plan (figure 14).
- **tol** : C'est la tolérance qui détermine à quel point la méthode s'arrête en fonction de la convergence. L'algorithme s'arrêtera lorsque la variation de la fonction de niveau de seuil est inférieure à cette valeur (figure 15).
- **max\_num\_iter** : C'est le nombre maximal d'itérations que l'algorithme effectuera pour converger vers une solution. Si la convergence n'est pas atteinte après ce nombre d'itérations, l'algorithme s'arrête (figure 16).
- **dt** : C'est le pas de temps utilisé pour les itérations. Il contrôle la vitesse à laquelle le contour de séparation évolue (figure 17).

Figure 14: Impact du paramètre **lambda2**Figure 15: Impact du paramètre **tol**Figure 16: Impact du paramètre **max\_num\_iter**Figure 17: Impact du paramètre **dt**

- **init\_level\_set** : Il s'agit de l'initialisation du niveau de seuil. On peut fournir un tableau NumPy représentant la forme initiale de notre contour de séparation. Ce contour de niveau initial peut être une forme approximative du contour que l'on souhaite obtenir (figure 18).
- **extended\_output** : Un paramètre booléen qui détermine si des informations

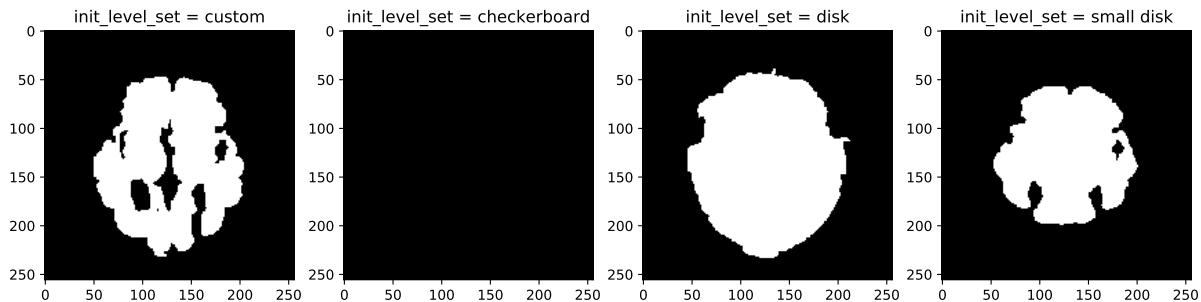


Figure 18: Impact du paramètre **init\_level\_set**

supplémentaires doivent être renvoyées en plus du résultat de la segmentation. Si True, la fonction renverra des informations telles que l'évolution de l'énergie, le nombre d'itérations, et d'autres données utiles.

## 2 Segmentation

Pour au moins l'une des quatre images et une autre de votre choix (soit une autre parmi les images fournies, soit n'importe quelle autre image), proposez une méthode de segmentation en utilisant l'une des approches précédentes pour segmenter certaines des structures anatomiques. Justifiez vos choix, expliquez les résultats obtenus et discutez des idées potentielles pour améliorer les résultats.

### 2.1 Segmentation par contours actifs

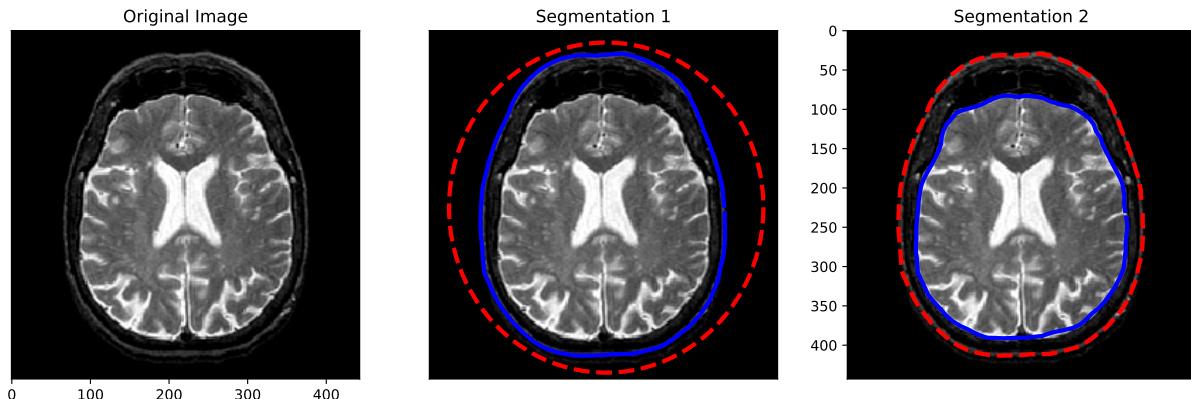


Figure 19: Segmentation par contours actifs d'une image de cerveau.

On commence par manuellement créer un contour actif autour du cerveau, qui nous permet de réaliser une première segmentation du cerveau. Ensuite, on utilise ce résultat comme nouveau contour actif et on réalise une seconde segmentation qui nous permet de segmenter la partie intérieure du cerveau (figure 19).

## 2.2 Segmentation par Chan Vese

### 2.2.1 Radiographie cervicale

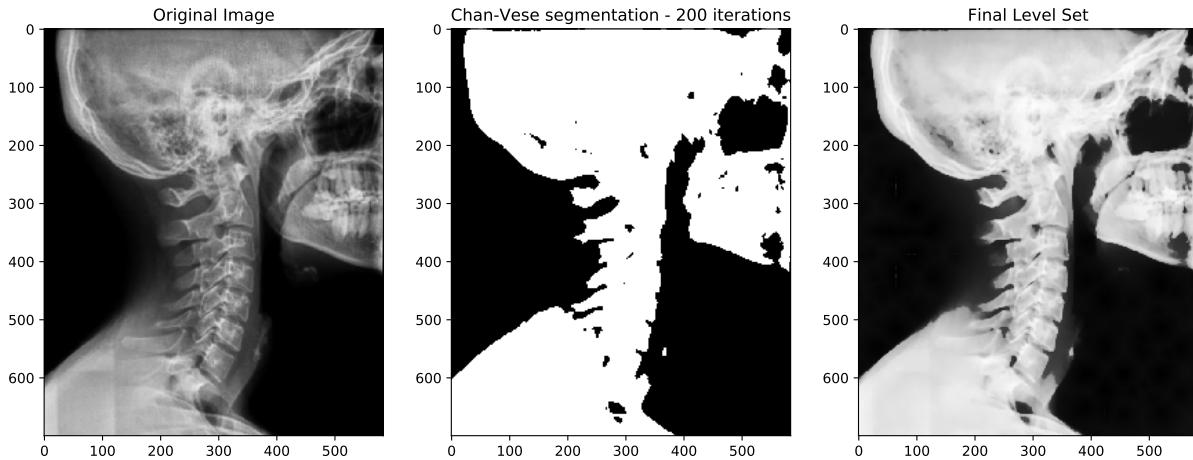


Figure 20: Segmentation par Chan Vese d'une radiographie cervicale.

On a tenté une segmentation sur une image de radiographie cervicale. Après avoir trouvé de bons paramètres, cela a plutôt bien fonctionné (figure 20).

### 2.2.2 Radiographie de la main

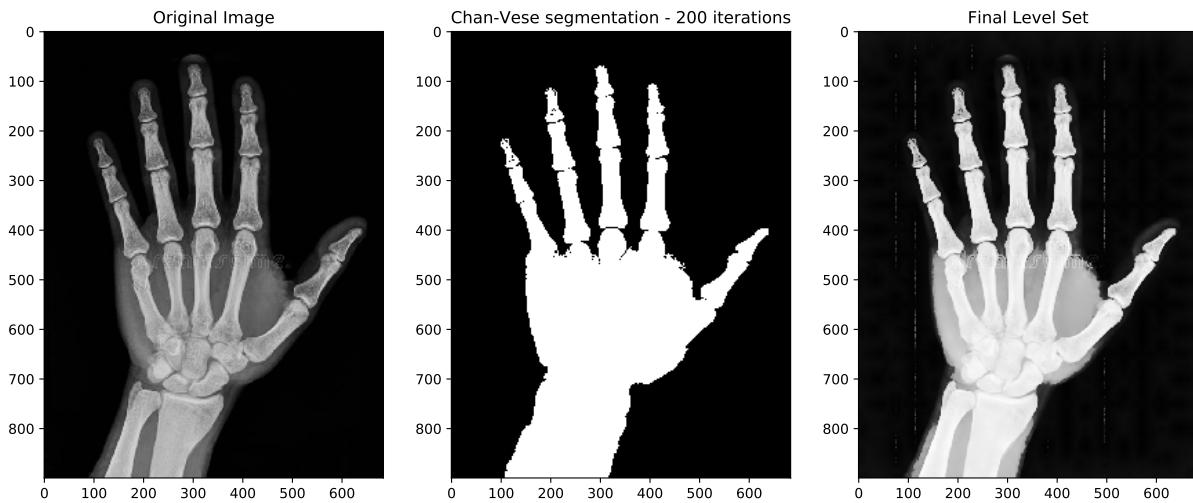


Figure 21: Segmentation par Chan-Vese d'une radiographie de la main.

On essaye de segmenter le squelette de la main dans une image de radiographie (figure 21). Cela fonctionne bien au niveau des doigts mais moins bien au niveau de la paume. En effet, segmenter la paume de la main dans une radiographie peut être un défi en raison des variations d'opacité des tissus mous. Voici quelques idées pour améliorer la segmentation dans cette zone :

- On pourrait appliquer des techniques de **prétraitement** pour améliorer la qualité de l'image (e.g. égalisation d'histogramme, réduction de bruit, ou correction de contraste pour mieux distinguer les tissus mous).
- On pourrait utiliser une **initialisation adaptative** pour le level set initial (Chan-Vese). Au lieu d'une initialisation uniforme, on démarrerait avec un contour initial basé sur la détection des bords ou des caractéristiques locales de la paume.
- On pourrait combiner la méthode Chan-Vese avec d'autres méthodes de segmentation, telles que les **réseaux de neurones convolutifs** (CNN), pour une segmentation plus robuste.
- On pourrait utiliser la connaissance a priori de la forme de la paume pour limiter la **région d'intérêt** (ROI) de la segmentation.
- Enfin, on peut encore **expérimenter avec les paramètres** de la méthode Chan-Vese pour trouver les valeurs optimales qui s'adaptent le mieux aux caractéristiques de notre image.

## References

- [1] [scikit-image.org/docs/stable/api/skimage.segmentation.active\\_contour.](https://scikit-image.org/docs/stable/api/skimage.segmentation.active_contour.html)
- [2] [scikit-image.org/docs/stable/api/skimage.segmentation.chan\\_vese.](https://scikit-image.org/docs/stable/api/skimage.segmentation.chan_vese.html)