Barriol Ethan IMA 201

TP Segmentation

Partie 1 : Détection de contours

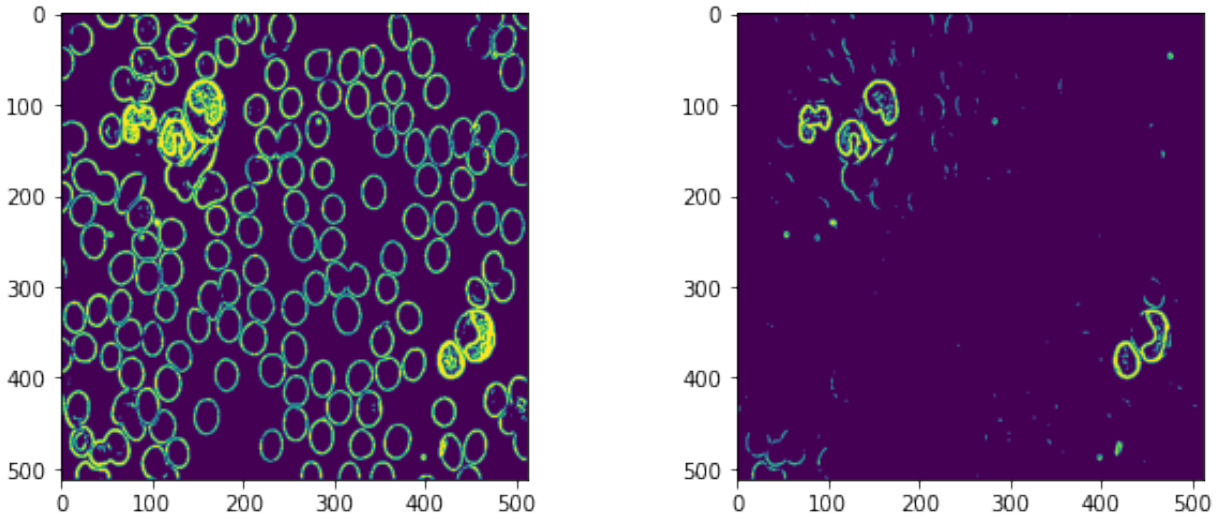
Filtre de gradient local par masque

On utilise le fichier python « sobel.py » sur l'image « cell.tif ».

Le filtre de Sobel est conçu spécifiquement pour détecter les contours dans une image. Il utilise une paire de noyaux de convolution pour calculer les gradients de l'image dans les directions horizontale et verticale. Ces gradients permettent de mettre en évidence les variations d'intensité qui correspondent généralement à des contours, tandis que la simple différence entre deux pixels voisins peut être sensible au bruit.

Il est nécessaire d'appliquer un filtre passe bas pour débruiter les zones homogènes de l'image et se rapprocher de l'hypothèse de continuité de l'intensité dans l'image.

En augmentant le seuil de la norme du gradient, on voit que certains contours ne sont plus détectés mais cela permet d'éliminer du bruit indésirable dans l'image. Il s'agit donc de choisir le bon seuil pour avoir une detection des contours optimale.



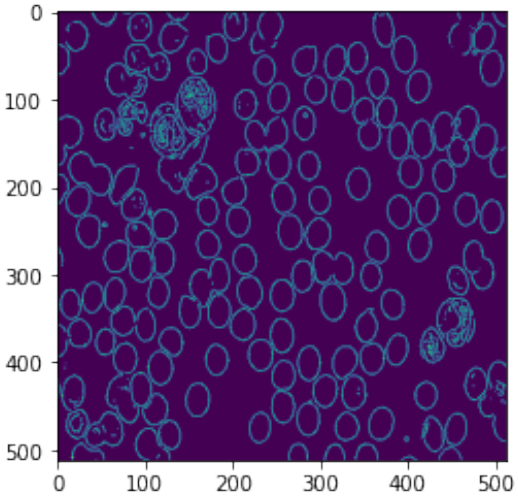
Seuil à 0.15 et 0.30

Si l'on applique des seuils assez bas, on voit que les contours sont assez épais et que ceux-si sont un peu décalés par rapport aux vrais contours car le filtre inclus des pixels qui ne sont pas sur la frontière.

Maximum du gradient filtré dans la direction du gradient

La fonction maximaDirectionGradient permet de déterminer les pixels de l'image qui sont des maxima du gradient dans la direction du gradient. On remarque que l'image obtenue est très bruitée. Cependant, les contours sont visibles, aux bons endroits et surtout il semblerait que leur intensité est normalisée.

Après l'application d'un seuillage d'intensité sur l'image obtenue, on remarque des contours plus affutés et le bruit dans l'image est atténué par rapport à la méthode précédente. La continuité des contours est aussi optimisée. Le paramètre optimal de seuil déterminé de façon experimental est environ de 0.15.



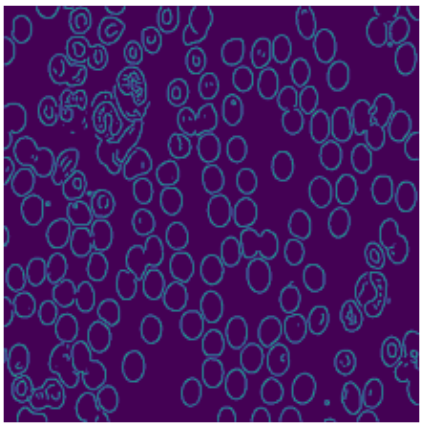
Seuil à 0.15 avec l'image obtenue en appliquant maximaDirectionGradient

Filtre récursif de Deriche

Le paramètre alpha du filtre de Deriche est crucial pour gérer la netteté de l'image filtrée. En effet, lorsque'il augmente, on a un rendu plus net.

On observe aussi que le temps de calcul ne change pas selon la valeur du paramètre alpha. C'est normal puisque celui-ci est stocké dans la variable ae au début de la fonction et n'est plus recalculé par la suite.

Les fonctions dericheSmoothX et dericheSmoothY sont utilisées pour appliquer un passe bas à l'image afin de se rapprocher de l'hypothèse de continuité de l'intensité, pour ensuite appliquer la détection de contours.

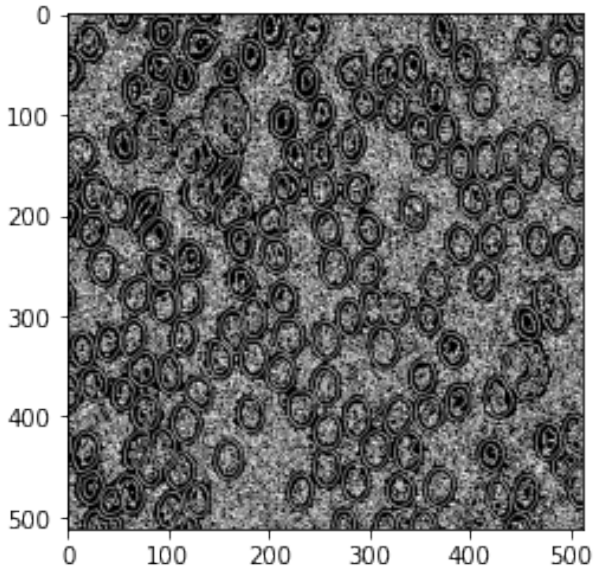


Résultat du seuillage avec le filtre récursif de Deriche

Passage par zero du Laplacien

Le paramètre alpha influence la détection des contours. Plus le alpha augmente plus les contours sont bien représentés et à la bonne position. Pour une valeur de alpha=0.9 le résultat du seuillage est acceptable.

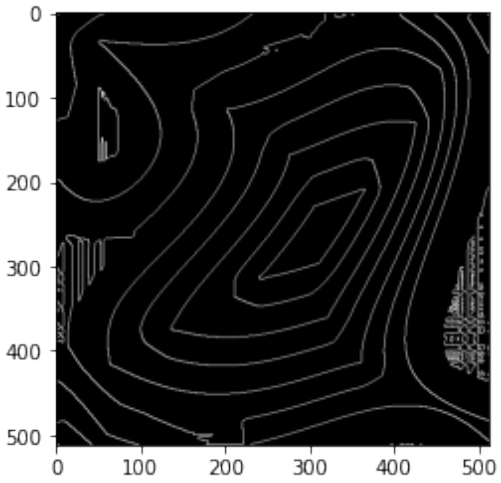
Sur l'image « cell.tif », les contours pour des alphas plus petits sont rectangulaires.



Contours obtenus avec alpha=3 pour « cell.tif »

On remarque qu'avec l'utilisation de cette méthode les contours sont beaucoup moins discernables qu'avec les deux méthodes précédentes.

Sur l'image « pyramide.tif » on observe des contours parasites sur des zones pourtant homogènes.



Contours obtenus avec alpha=3 pour « pyramide.tif »

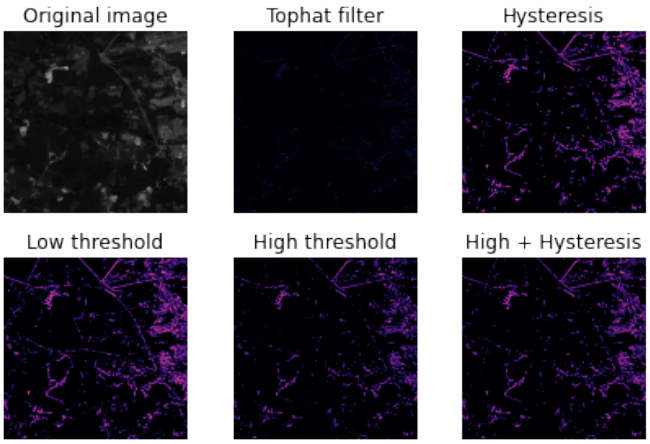
On remarque qu'en abaissant le paramètre alpha, cet effet est moins présent.

Changement d'image

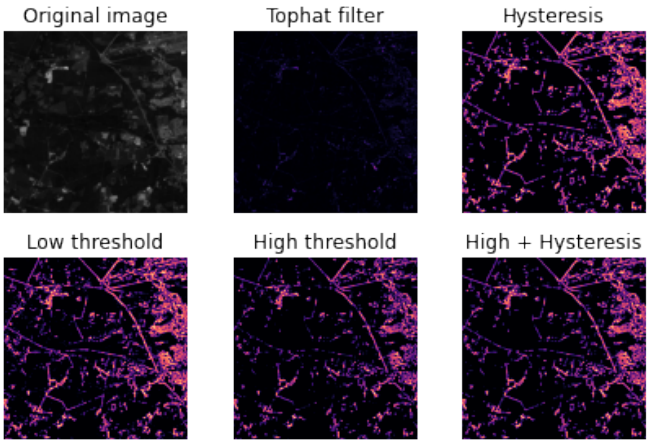
Dans le cas de « pyra-gauss.tif », il faut utiliser un filtre moins sensible au bruit comme par exemple de filtre de sobel.

Partie 2 : Seuillage avec hystérésis

On applique le filtre tophat à l'image « spot.tif ». On remarque que plus le rayon est élevé mieux les contours sont représentés sur l'image filtrée



« Spot.tif » filtré avec rayon r=3



« Spot.tif » filtré avec rayon r=5

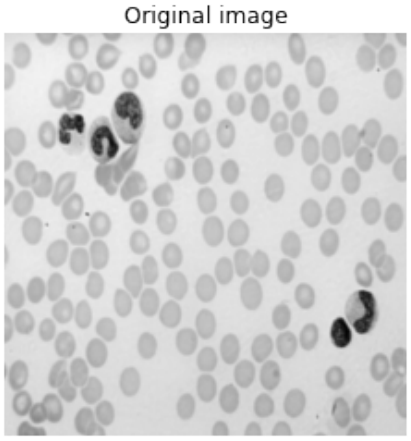
Si on augmente le seuil bas, la résolution des contours des objets dans l'image augmente.

Après plusieurs essais, on peut remarquer que le meilleur paramétrage pour cette image spécifique est r = 5, low = 2 et high = 7.

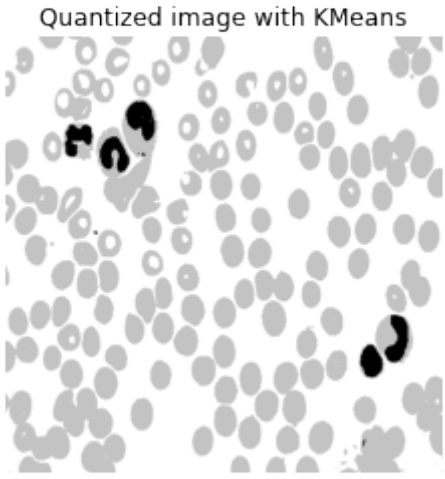
Partie 3 : Segmentation par classification : K Moyennes

Images à niveau de gris

On utilise à présent l'algorithme de K Means pour classifier chaque pixel. On voit que celui-ci permet d'avoir un résultat globalement satisfaisant en terme de précision et de position des contours. Il existe cependant quelques anomalies mais cela est assez mineur comparé aux résultats obtenus avec les méthodes précédentes. Avec deux classes, nous ne parvenons pas à detecter les différents types de cellules.

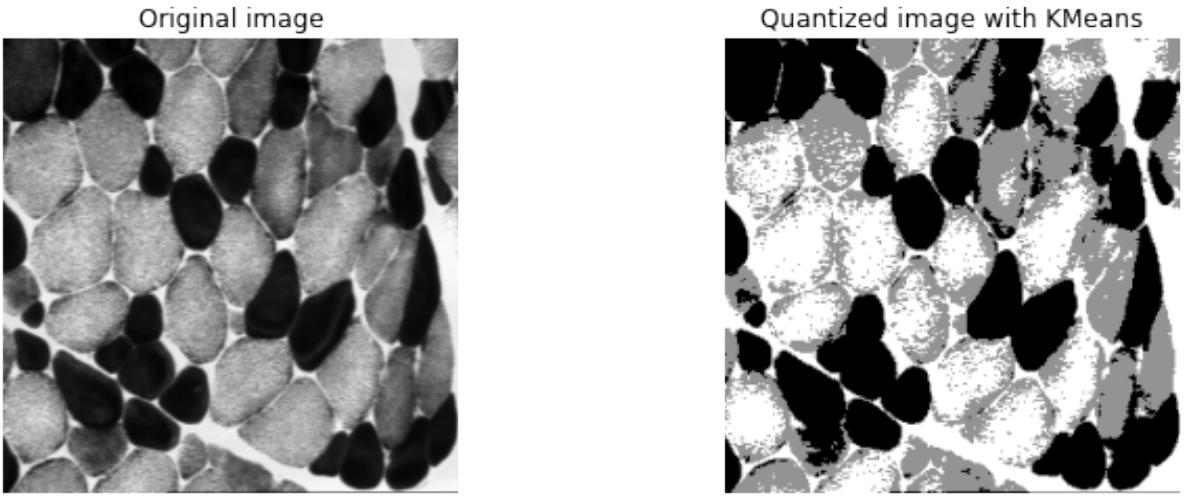


Afin de détecter les différents types de cellules, il suffit d'ajouter une troisième classe. On remarque que les résultats de la classification sont les mêmes que ce soit avec centres initiaux placés de manière déterministe ou non. Si on ajoute des classes supplémentaires, la détection des cellules n'est plus efficace.



K Means avec 3 classes

À présent si on utilise l'image « muscle.tif » avec la méthode k means à 3 classes, on remarque que la classification n'est plus si efficace. En effet, il y a plus de nuances de gris et certaines zones grises foncées ou très claires sont mal catégorisées.



K means avec 3 classes.

Une solution pour palier a ce problème est de filtrer l'image avec un filtre médian pour égaliser les zones homogènes des fibres.

Image en couleur

Si on teste l'algorithme de K means sur l'image « fleur.tif » avec 10 classes, on voit que globalement les caractéristiques de l'image sont bien distinguables. Cependant, les couleurs sont moins vives et les nuances sont moins importantes.

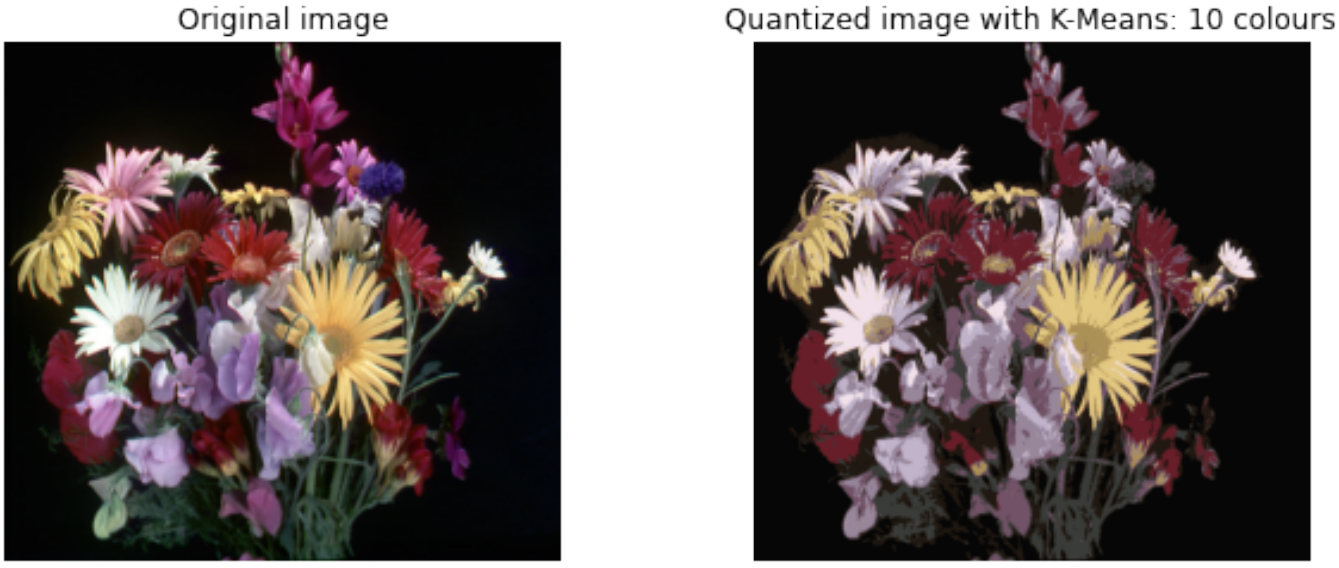


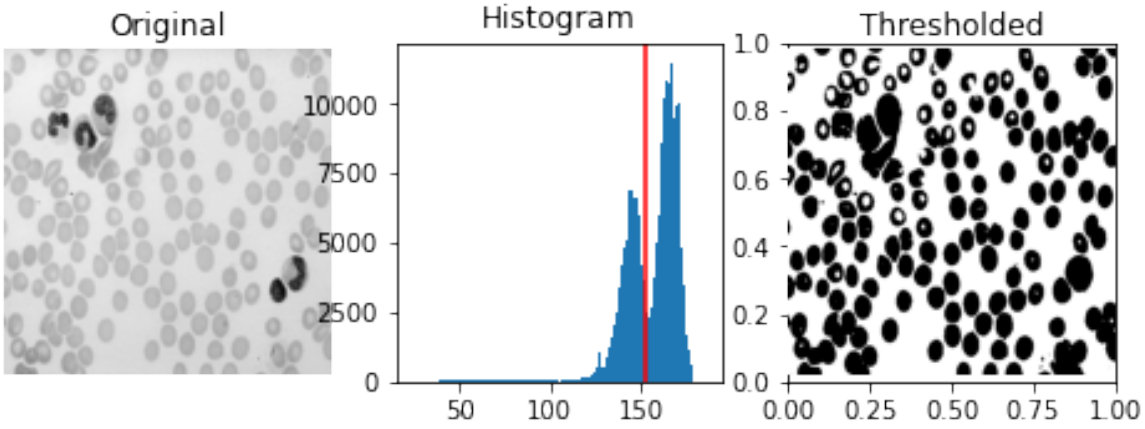
Image quantifiée avec 10 couleurs

Pour obtenir la même image, il faudrait avoir 256x256x256 classes différentes correspondants aux trois canaux de couleurs.

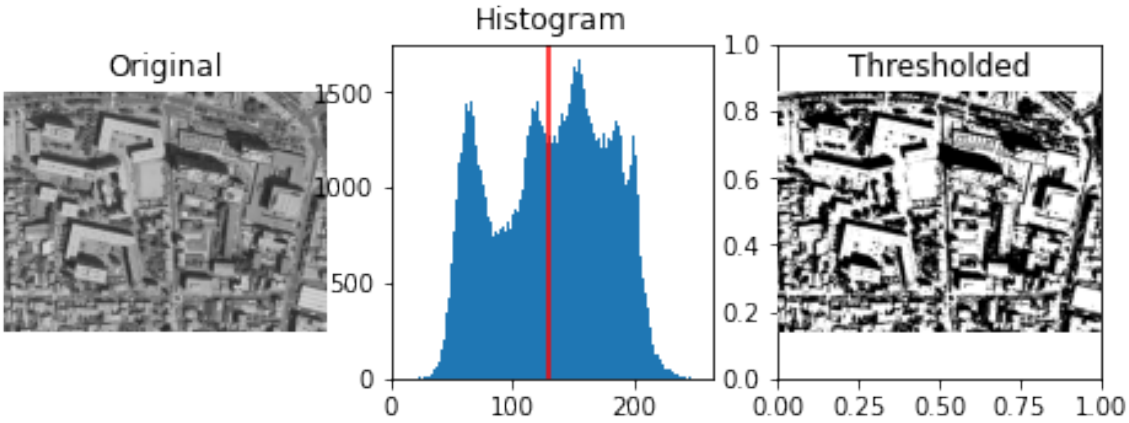
Pour « carte.tif », il suffit d'augmenter le nombre de classes de l'algorithme K means.

Partie 4 : Seuillage automatique : Otsu

Le script « otsu.py » permet de choisir le seuil optimal pour séparer l'image en deux canaux. Le script cherche à maximiser la variance interclasse à partir de l'histogramme de l'image.



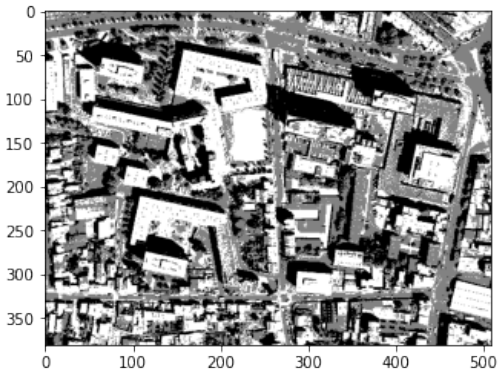
La méthode d'Otsu sur « cell.tif »



La méthode d'Otsu sur « Vincennes.tif »

Nous pouvons voir que cette méthode permet de seuiller assez correctement. Cependant, certaines images avec des niveaux de gris très étendus donnent des résultats moins satisfaisants.

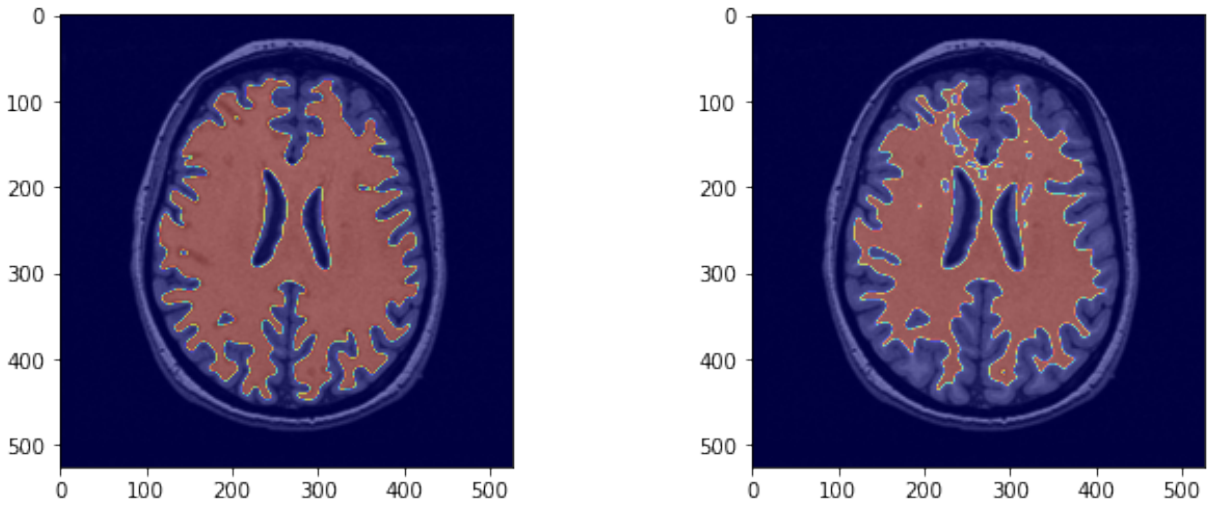
En modifiant l'algorithme d'Otsu, nous obtenons une image quantifiée avec 3 intensités de pixel.



Partie 5 : Croissance des régions

L'algorithme de croissance de régions (region growing en anglais) est une technique de segmentation en traitement d'image qui vise à partitionner une image en plusieurs régions ou objets en se basant sur les propriétés locales des pixels. L'idée principale est de faire pousser une région à partir d'un ou plusieurs pixels de départ (appelés graines) en ajoutant de manière itérative des pixels voisins qui satisfont un critère de similarité des niveaux de gris et un critère de connexion spatiale.

Le paramètre thresh dans l'algorithme de croissance de régions contrôle la similitude requise entre les pixels voisins et les pixels existants dans la région en cours de croissance. Un seuil haut permet une croissance rapide et peut provoquer une sur-segmentation, tandis qu'un seuil bas restreint la croissance et peut entraîner une sous-segmentation.



thresh=5 à gauche et thresh=2 à droite

Le seul paramètre qui va influencer si on segmente de la matière blanche ou de la matière grise est la point initial. Si celui-ci est dans de la matière grise alors l'algorithme segmentera aussi cette matière grise.