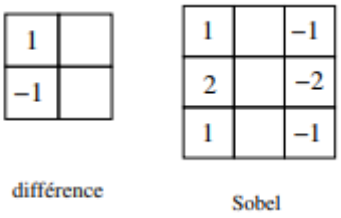
Giovanni Benedetti da Rosa

TP3- IMA 201

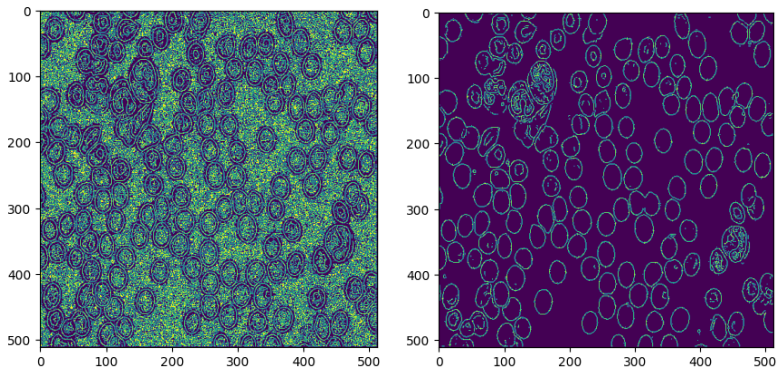
1 Détection de contours

1.1 Filtre de gradient local par masque

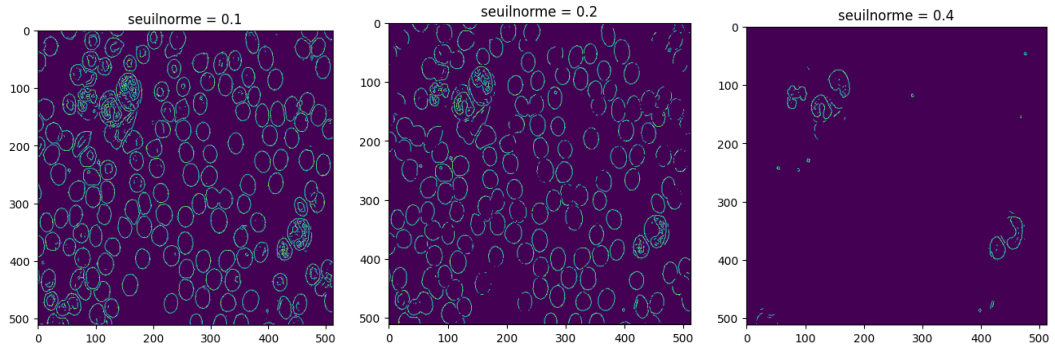
Le filtre de différence est un opérateur utilisé pour estimer la dérive d'une image dans une seule direction, ce qui signifie qu'il mesure principalement les variations d'intensité dans cette direction particulière, en négligeant les bords et contours. Le filtre de Sobel utilise deux noyaux de convolution distincts, un pour chaque direction(pour chaque gradients). Ces noyaux sont conçus pour capturer les variations d'intensité dans ces deux directions, en fournissant une réponse plus robuste à contours.



Il n'est pas nécessaire d'appliquer un passe-bas, parce qu'il dejá fait cela dans la convolution(passe-bas + differentiation). Néanmoins, l'utilisation d'un filtre Gaussienne pour réduire le bruit de l' image peut aider dans la performance.



Il semble que pour l'image cell.tif dans ce cas, la valeur idéale de seuil soit égale à 0,1. En augmentant cette valeur, nous réduisons la sensibilité au bruit, mais nous commençons à obtenir des contours non continus. Cependant, en diminuant davantage, nous obtenons des contours de parties qui ne représentent pas l'image de cellules.



1.2 Maximum du gradient filtré dans la direction du gradient

Le critère de qualité qui sera optimisé pour ce processus est la localisation des bords détectés, puisque le gradient est toujours perpendiculaire au bord détecté.

On peut voir une optimisation en utilisant une interpolation bilinéaire dans le point (i,j) de la norme, parce que non nécessairement la direction de la norme est une valeur entière de l'image. Après ça il y a une vérification si on est vraiment dans la direction de la maximum gradient.

Le meilleure valeur a été 0.1, compromis robustesse au bruit et continuité des contours. On peut voir qu' en variant ça, le résultat de gradient change aussi.

1.3 Filtre récursif de Deriche

Lines modifiés:

```
b1[j] = I[j-1] + 2 * ae * (b1[j - 1]) -(ae)**(2)*b1[j-2]
b2[j] = I[j-1] + 2 * ae * (b1[j + 1]) -(ae)**(2)*b2[j+2]
```

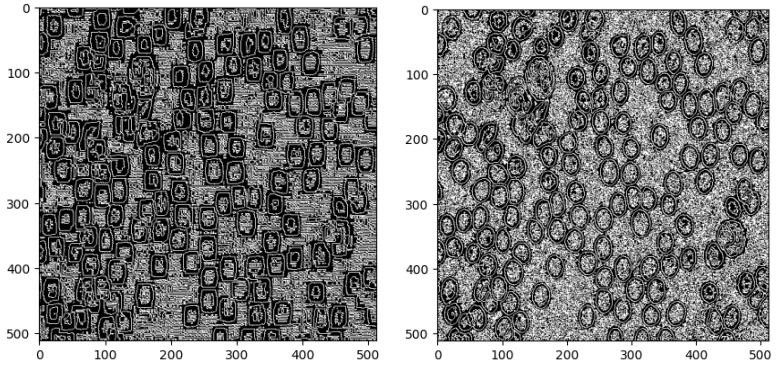
Varíation du paramètre alpha

Avec un alpha mineur les bords sont plus définis et avec moins de bruits en comparaison avec les valeurs plus hautes, puisque ce paramètre est inversement proportionnel à l'écart-type du filtre log de Canny.

Le temps de calcul ne dépend pas de la valeur de α, parce que la complexité du filtre dépend seulement de la dimension de l'image($O(n^2)$).

Les fonctions *Smooths* produisent un résultat similaire à une filtre Sobel, pour rencontrer les contours des images, en appliquant un passe-bas.

1.4 Passage par zéro du laplacien



variation du paramètre

En augmentant la valeur de alpha jusqu' à une limite les countours aimeillore, parfois en créant des bords artificiels La principale différence c'est que le laplacien retourne toujours un contourne fermée, au contraire des autres opérateurs.

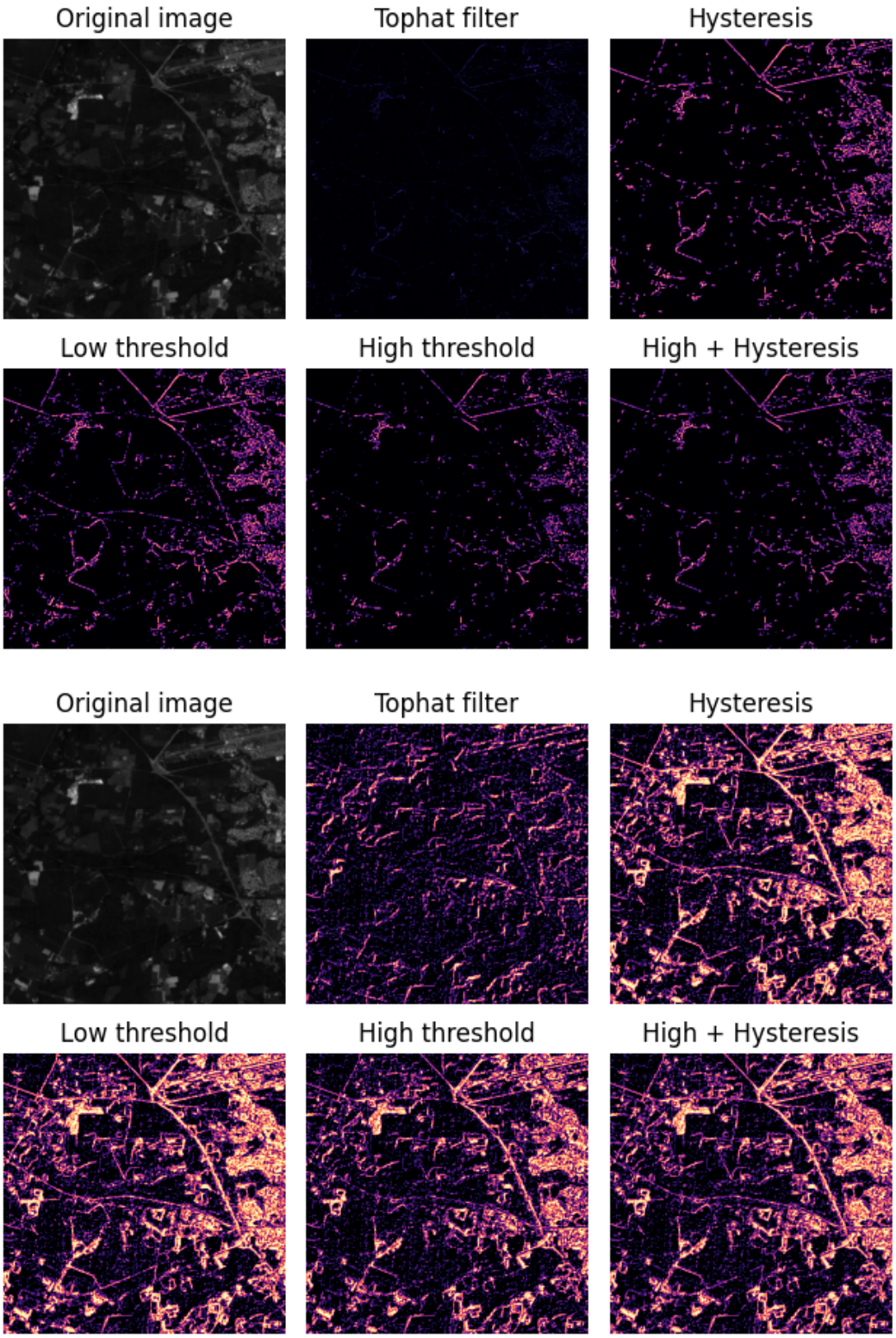
Sur l'image pyra-gauss.tif, on pourrait appliquer une filtre gaussienne ou moyenne pour essayer de retirer le bruit, en lissant l'image, une fois que cet opérateur est très sensible à des petites variations.

1.5 Changez d'image

Le filtre Deriche est basé sur des techniques de filtrage récursif, qui lui permettent de capturer les variations à différentes échelles. Je l'utiliserais. Avant, c'était une bonne pratique d'utiliser une gaussienne ou moyenne, pour lisser l'image et éviter des contours dans les bruits. Pour les post-treatments, on pourrait appliquer, seul igae, max local du gradient, etc..

Des bons résultats ont été obtenus avec un filtre médian suivi de le filtre de Deriche.

2 Seuillage avec hystérésis



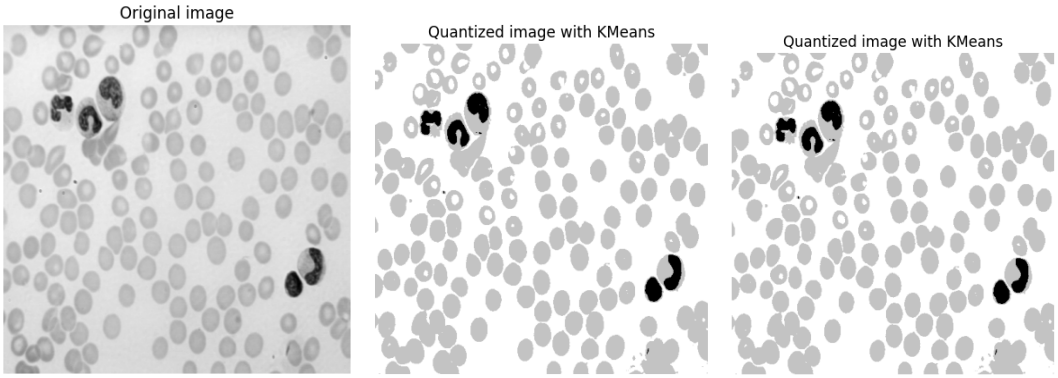
En augmentant les valeurs de rayon plus détaillées sont capturés par l'algorithme, et bien sûr plus de bruits. On peut voir que les valeurs impaires des rayons produisent de meilleurs résultats, pour une question de symétrie.

À mon avis, le meilleur résultat est: low = 2, high =9, rayon =7.

Je me suis rendu compte qu'en appliquant ce filtre sur l'image cell.tif sur laquelle sorbel avait été appliqué auparavant, le résultat de cet opérateur renforce les contours mal remplis.

3 Segmentation par classification : K-moyennes

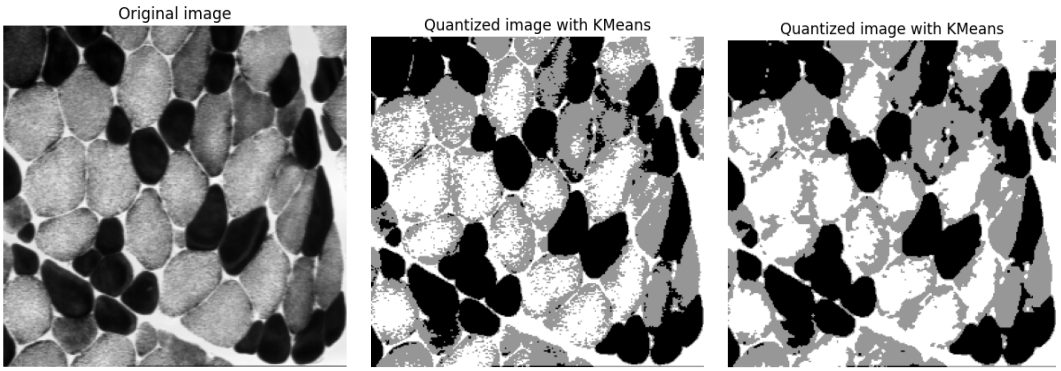
3.1 Image à niveaux de gris



Clairement cela ne qualifierais bien en deux classes les cellules, parce que l'image a trois couleurs, mais si on impose 3 classes ça fonctionne bien.

Les 3 différents possibles initialization:

- k-means++: Sélectionne les points basées sur une distribution empirique des probabilités qui sélectionne les points plus proches á les centroids en utilisant l'inertie.
- random: sélectionne aléatoirement dans les points de le data set
- array: l'utilisateur choisit les centres initiaux



Comparaison avant et après la filtrage

3.2 Image en couleur

Les difficultés surviennent car l'une des cellules a une teinte très proche du blanc (extrémité de l'image), renforcée par des points blancs. En appliquant un filtre, on supprime ces points, uniformisant ainsi la région et améliorant le processus.

Même si la couleur et la profondeur ont été un peu perdues, le résultat est assez bon. Pour moi le nombre minimum pour que l'image se parait vraiment à l'image initiale est 6. Après, pour être vraiment similaire, une valeur proche de 60 est nécessaire.

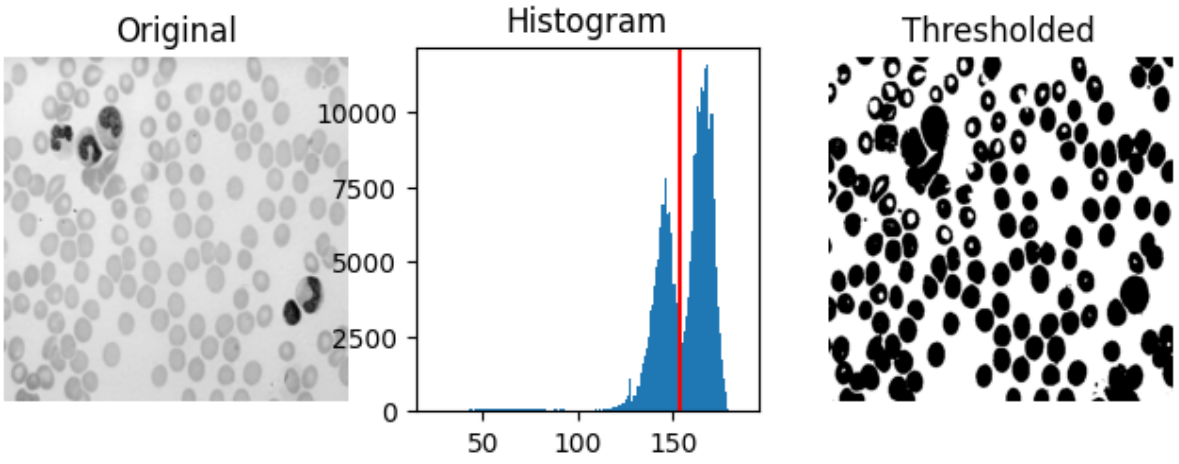
Pour obtenir les planches-mères, j'essaierais de faire sembler les diagrammes de les images après une conversion pour le grayscale.(Je ne sais pas si j'ai bien compris cette partie).

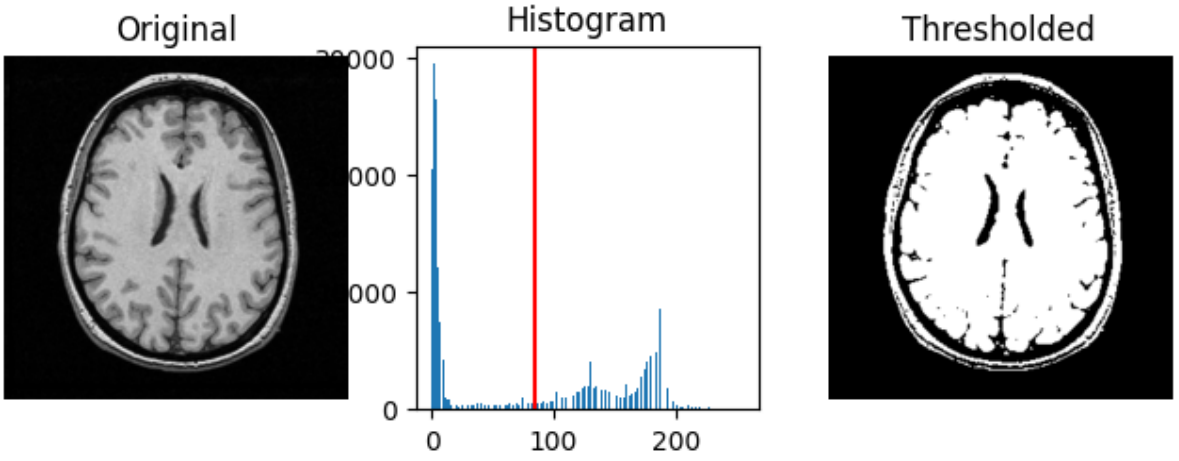
4 Seuillage automatique : Otsu

L'objectif de l'algorithme d'otsu est de minimiser la variance intra-classe ou à maximiser la variance inter-classe.

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$$

qui est exprimée en termes des probabilités de classe w_i et les moyennesμ_i.





Il est possible de constater que l'algorithme a bien clusterisé en deux classes.

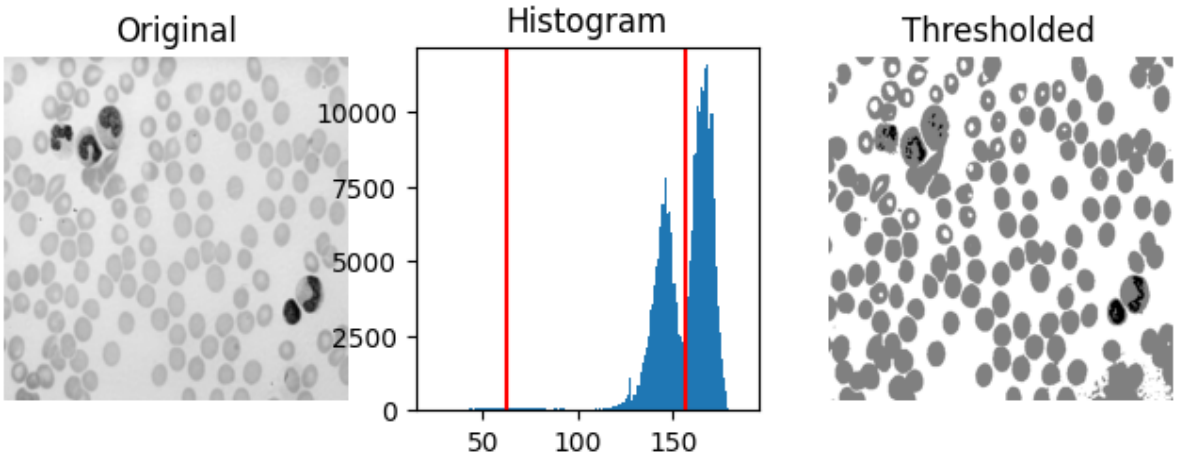
C'est possible de sélectionner correctement une image de norme du gradient, puisque les normes les plus grandes (contours) normalement seraient séparées de les normes plus bas.

Script pour 3 classes

```
def otsu thresh(im):

   h=histogram(im)
   m=0
   for i in range(256):
      m=m+i*h[i]

   maxt1, maxt2=0, 0
   max k = 0

   for t1 in range(256):
      for t2 in range(256):
         w0=0
         w1, w2=0,0
         m0=0
         m1, m2 = 0,0
         for i in range(t1):
            w0=w0+h[i]
            m0=m0+i*h[i]
         if w0 > 0:
            m0=m0/w0

         for i in range(t1,t2):
            w1=w1+h[i]
            m1=m1+i*h[i]
         if w1 > 0:
            m1=m1/w1

         for i in range(t2,256):
            w2=w2+h[i]
            m2=m2+i*h[i]
         if w2 > 0:
            m2=m2/w2

         k=w0*w1*(m0-m1)*(m0-m1) + w1*w2*(m1-m2)*(m1-m2)
```

```
        if k > maxk:
            maxk=k
            maxt1=t1
            maxt2=t2

    thresh=(maxt1, maxt2)
    return(thresh)
```

pou

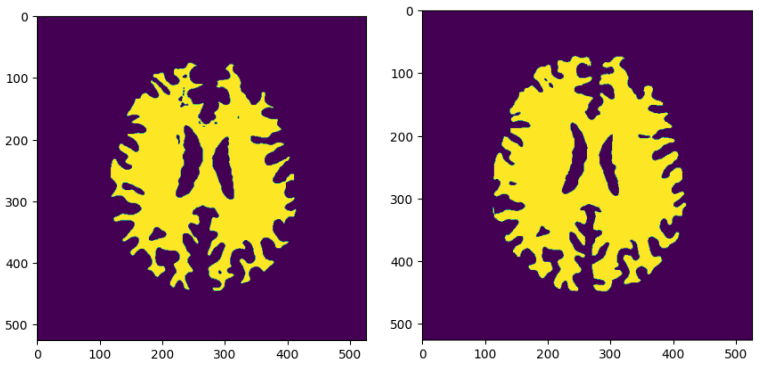


Résultat pour l'algorithme implémenté

5 Croissance de régions

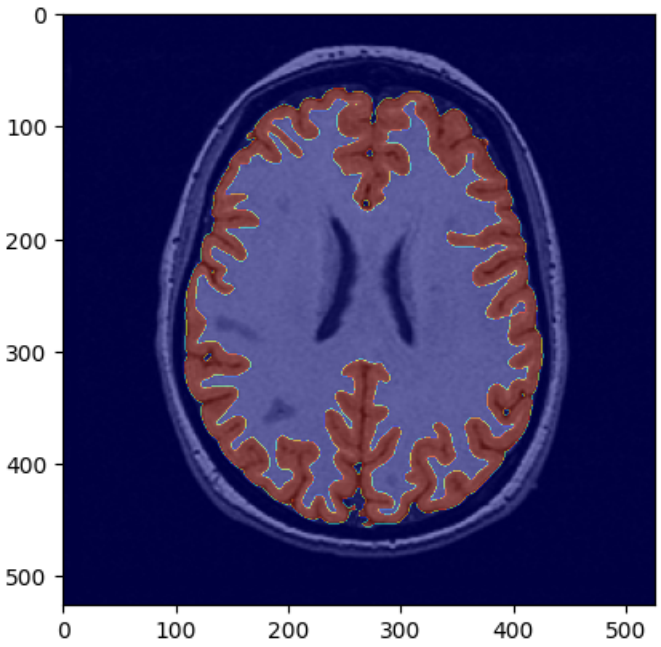
Les contraintes sont que la différence entre la moyenne d'un pixel ajouté avec ses voisins et la moyenne de cette pixel initiale avec ses voisins doit être inférieure à un seuil multiplié par l'écart-type(local) d'un pixel originale avec ses voisins.

En augmentant thresh l'algorithme considère que plus pixels faisant partie de la région.



Pour moi, je pense que thresh=5, rayon=3 et (x0,y0)=(300,300) segmentent correctement la matière blanche.

Pour la matière grise fonctionne raisonnablement bien avec: thresh=3, rayon=3 et (x0,y0)=(270,350) segmenente correctement.



J’ai eu des difficultés pour comprendre les questions suivantes.

• Quel est le prédicat mis en place dans ce script ?
Prédicat de Similarité basé sur la Moyenne et l'Écart-type local.

• Proposez un autre algorithme qui n’utilise pas la croissance de régions, mais qui donne le même résultat.
Examiner la différence de moyenne entre un pixel candidat et un pixel initial, comparée à un seuil défini en fonction de l'écart-type local. Les pixels dont la différence de moyenne est inférieure au seuil sont ajoutés à la région en cours de développement.

• Proposez un prédicat qui nécessite réellement un algorithme de croissance de région.

Prédicat de Connectivité. L'idée de ce prédicat est de vérifier si un pixel courant est connecté à un pixel initial (graine) par le biais de pixels voisins similaires. La connectivité peut être définie en termes de voisinage.