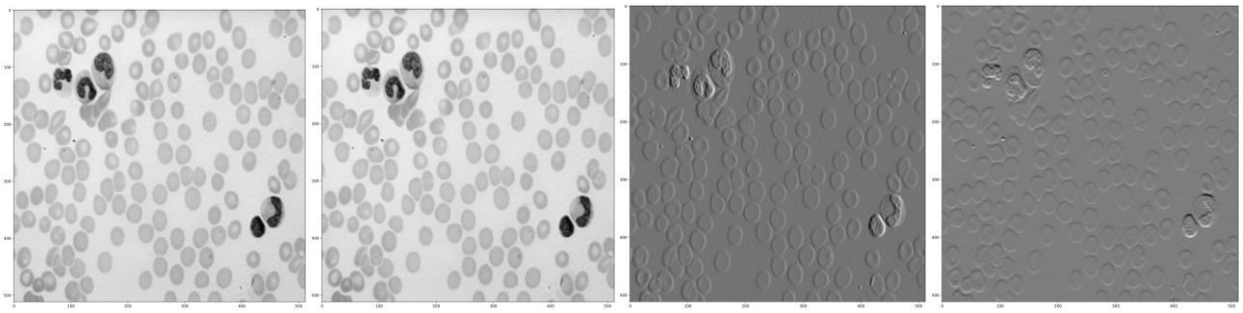
Travaux pratiques 3 - IMA201- Réponses

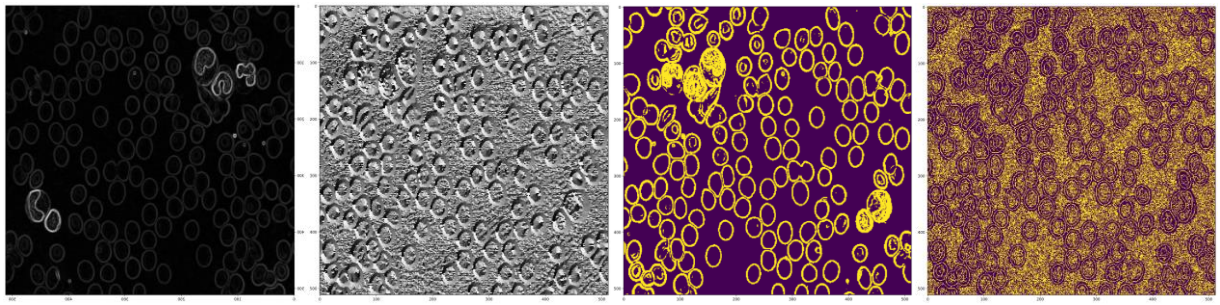
Júlia Togashi de Miranda

1. Détection de Contours

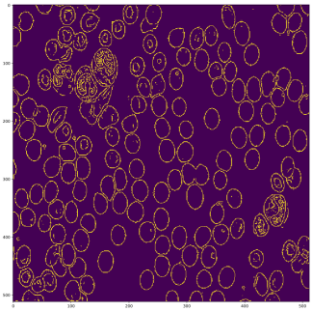
1.1. Filtre de gradient local par masque



L'image cell.tif originale ; après passer pour un filtre passe-bas ; dérivées partielles suivant x et y, respectivement



Norme et direction du gradient ; le Norme seuillée et le Maxima du gradient dans la sa direction



Contours

Par rapport au filtre différence, le filtre Sobel en plus de fait le calcul du gradient dans une direction, il effectue un lissage dans la direction orthogonale, ça fait qui il soit moins sensibles au bruit.

Oui. C'est nécessaire, parce que quand nous ne passons l'image avant pour un filtre passe-bas nous trouvons des très grandes valeurs pour la norme du gradient, parce qu'il y a beaucoup de variation dans l'image, donc nous ne pouvons pas trouver les bons contours.

Comme nous pouvons observer au-dessus, globalement (avec le seuille égale à 0.1) nous avons obtenu des bons contours. Pour la majorité des cellules ils sont fermes (continus), ne sont pas très grosses. Nous avons des petits problèmes avec des cellules que différences niveaux de gris. Si nous augmentons cet seuille, nous pouvons avoir des contours qui ne sont pas continu (on détecte moins que les vrais contours) et si on diminue, nous détectons des bruits ne que sont pas de contours.

1.2. Maximum du gradient filtré dans la direction du gradient

Il optimise les contours selon l'interpolation linéaire entre la norme et le gradient. Si nous augmentons cet seuille, nous pouvons avoir des contours qui ne sont pas continu (on détecte moins que les vrais contours) et si on diminue, nous détectons des bruits ne que sont pas de contours. Le seille égale à 0.1 nous donne des bons résultats (compromis entre robustesse au bruit et continuité des contours).

1.3. Filtre récursif de Deriche

```
42 ################################################
43 # 22 mai 2019
44 def dericheGradX(ima,alpha):
45
46
47     nl,nc=ima.shape
48     ae=math.exp(-alpha)
49     c=-(1-ae)*(1-ae)/ae
50
51     b1=np.zeros(nc)
52     b2=np.zeros(nc)
53
54     gradx=np.zeros((nl,nc))
55
56
57 #gradx=np.zeros(nl,nc)
58     for i in range(nl):
59         l=ima[i,:].copy()
60         for j in range(2,nc):
61             b1[j]=l[j-1]+2*ae*b1[j-1] -ae*ae*b1[j-2]
62         b1[0]=b1[2]
63         b1[1]=b1[2]
64         for j in range(nc-3,-1,-1):
65             b2[j]=l[j+1]+2*ae*b2[j+1]-ae*ae*b2[j+2]
66         b2[nc-1]=b2[nc-3]
67         b2[nc-2]=b2[nc-3]
68         gradx[i,:]=c*ae*(b1-b2);
69
70
71     return gradx
72
```

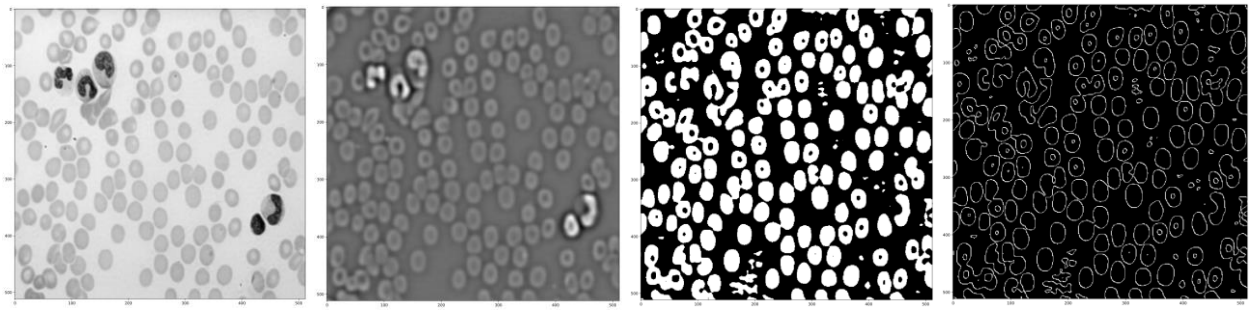
L'image pyrabruit.tif originale et le contour donne par le filtre Deriche (alpha égale a 1,0.3 et 3, respectivement)

Quand nous augmentons alpha on est plus sensible pour la détection du contour, par conséquence, on est plus sensible pour le bruir. Quand nous diminuions le alpha, nous trouvons des contours avec des bordes plus douces.

Non. Parce que la complexité de l'algorithme est $O(n^2)$, qui c'est le nombre de lignes fois le nombre de columens dans l'image. Le alpha est justement pour la multiplication, et le exponentiel de moins alpha est calcule en O(1), et il ne dépend pas de le valeur de alpha.

Il a le même effet que le filtrage de l'image dans le cas du filtre de Sobel.

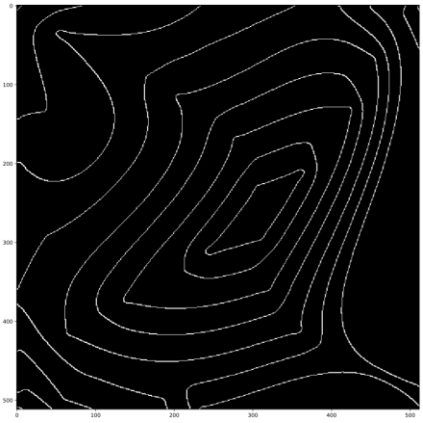
1.4. Passage par zéro du laplacien



L'image cell.tif originale ; Le laplacien ; après la binarisation et les contours fermées

Quand nous augmentons le paramètre alpha, nous avons comme résultat des contours du bruit. Quand nous le diminuons, nous trouvons des concours ajoutes des formes.

La principale différance c'est que le laplacien nous donne des contours fermés, alors que les précédents peuvent nous donner des lignes ouvertes



Les faux contours

Nous pouvons passer par un filtre gaussienne avant.

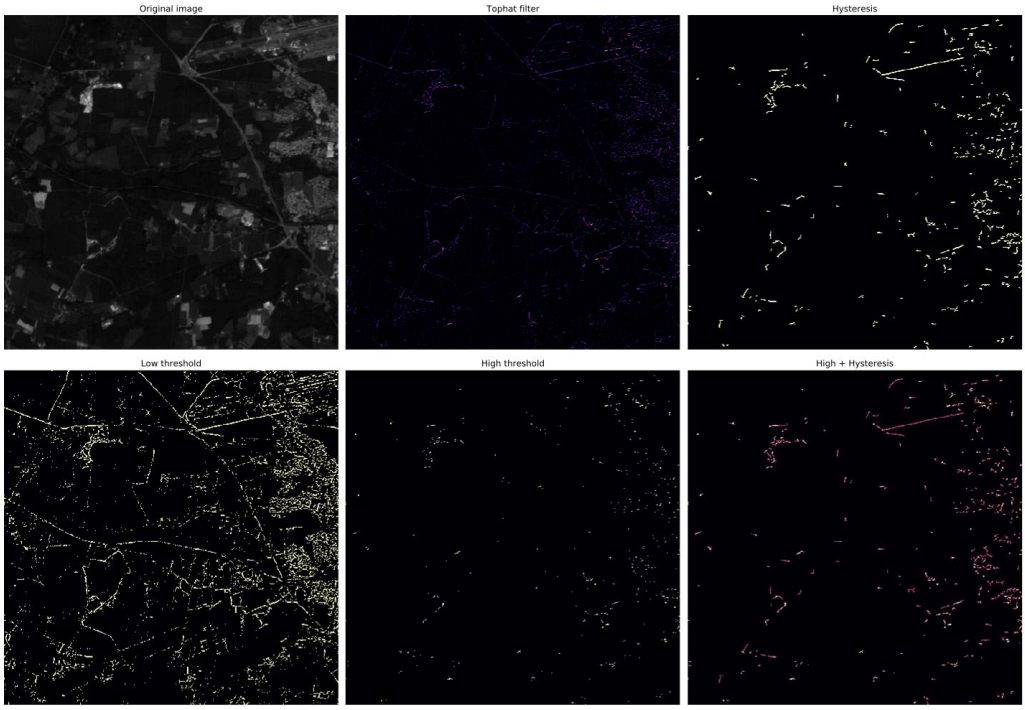
1.5. Changez d'image

Le Filtre récursif de Deriche.

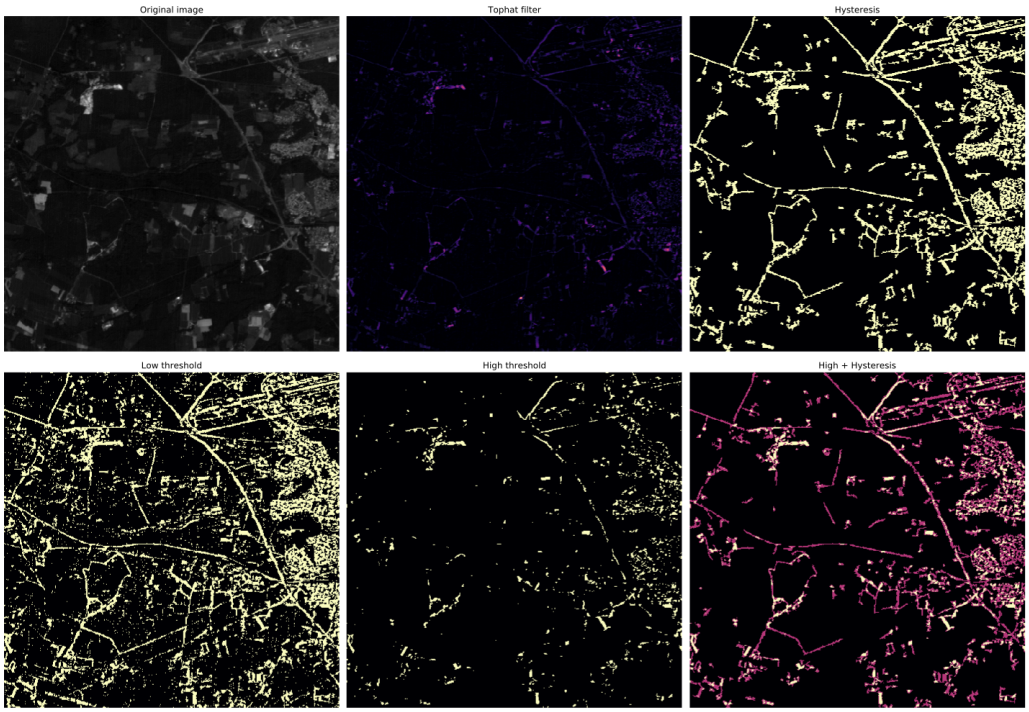
- *Pré-traitements : filtrage (médian, filtre bilatéral, ...)*
- *Post-traitements : seuillage, max local du gradient, hystérésis, poursuite, fermeture*

2. Seuillage avec hystérésis

2.1. Application à la détection de lignes



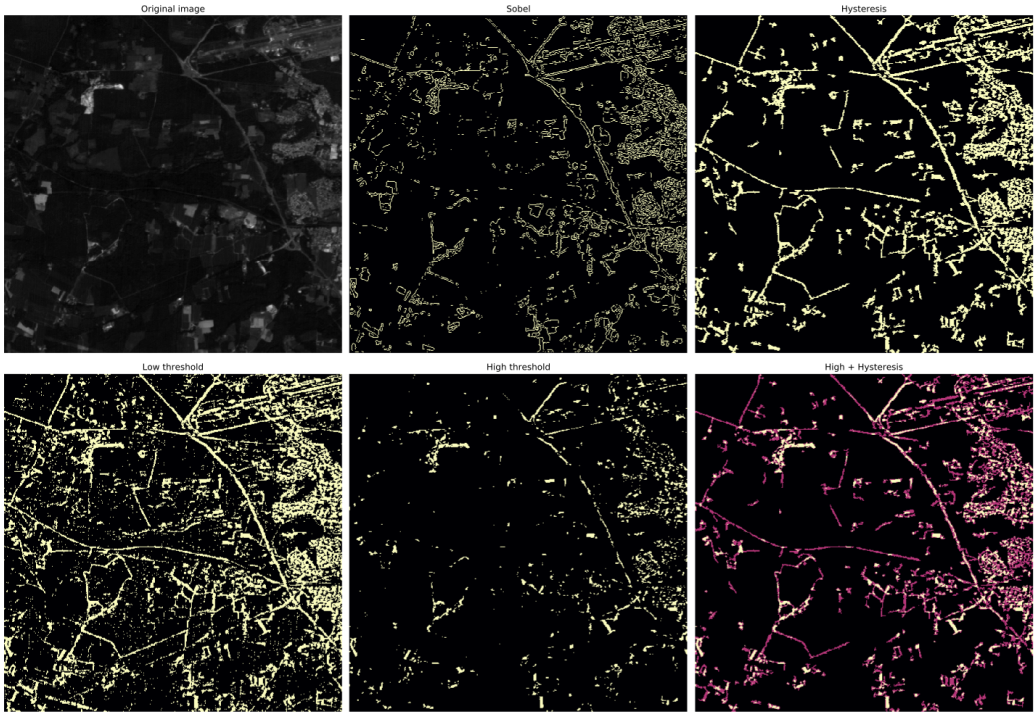
Application du le filtre du Chapeau haut de forme sur l'image spot.tif



Quand nous augmentons le rayon de l'élément structurant, on détecte plus de lignes (rayon égale à 3)

Quels sont les seuils qui donnent, à votre avis, le meilleur résultat ?

Les seuilles low = 2 et high = 9

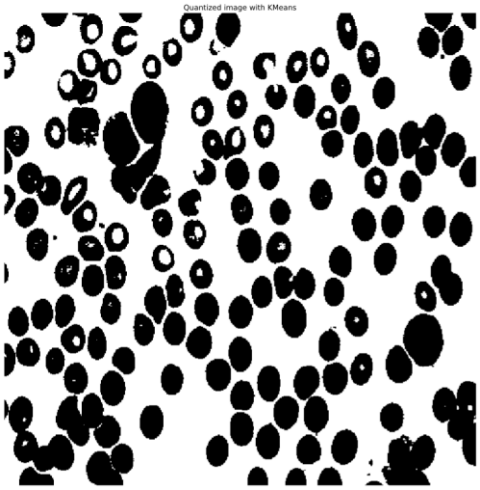


Seuillage par hysteresis sur Sobel

Comme nous pouvons percevoir, avec le seuillage par hystérésis, la méthode que nous avons appliquée sur 1.1 a des meilleurs résultats. Nous avons plus de contours continus et moins de bruit que dans le résultat original.

3. Segmentation par classification : K-moyennes

3.1. Image à niveaux de gris



Résultat avec k-means pour 2 classes

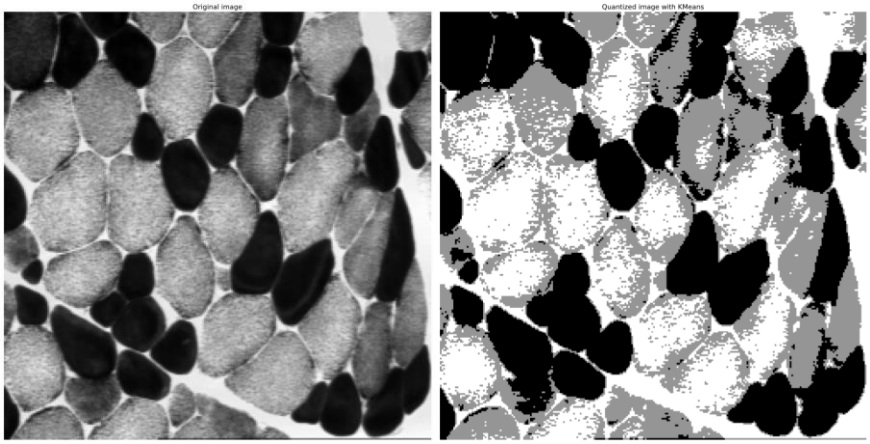
En général, il fait une bonne segmentation des cellules. Il y a une erreur de classification quand l'intérieur de la cellule c'est très clair. Nous pouvons essayer d'appliquer une méthode de morphologie mathématique, comme une fermeture pour essayer de resolutionner ce problème.

- k-means++ : Il fait un calcule de probabilité. Si le point c'est loin des centroids que on a déjà choisir, il y a une grande probabilité, si non, un petit. Donc nous avons une probabilité plus grande de commencer l'algorithme avec un centre en chaque cluster. Normalement, il donne des meilleurs résultats

- random : c'est aléatoire.

- ndarray : vous pouvez définir les centres.

Pour cette image, avec deux classes, la classification est constante, parce qu'ils sont deux classes très bien marques. Mais, si on applique l'algorithme sur une image ou la différence entre las classes n'est pas très clair, ou avec plus classes, ce n'est pas constant (même avec l'initiation k-means++)



L'image muscle.tif originale, et la classification du k-means pour 3 classes.

Notre problème c'est que quand nous regardons, ils ont trois classes bien marque. Les fibres très noires, les fibres gris, et qui n'est pas de fibre (clair). Nous pouvons bien classifier les fibres noires et les zones claires, mais en fait, les fibres grises ne sont pas des zones plates avec un niveau de gris constant. Il y a des petits points noirs et des petites zones claires, donc cette classification ne marche pas bien. Nous devons transformer ces zones en zones plates avant appliquer l'algorithme

Le filtrage de l'image original améliore le résultat de l'algorithme parce qu'il adoucie le bruit et des changements à niveau de gris très brutal, que sont des problèmes comme nous avons déjà percevoir sur l'image du muscle.

3.2. Image en couleur

L'image fleur.tif originale, et la classification du k-means pour 10 classes.

En rapport a l'image originale, nous avons trouvé une bonne approximation avec dix classes. Nous avons perdu des petites détailles, qui nous donne de sensation de volume (la fleur jaune est très plate), mais nous pouvons reconnaitre tous les éléments.

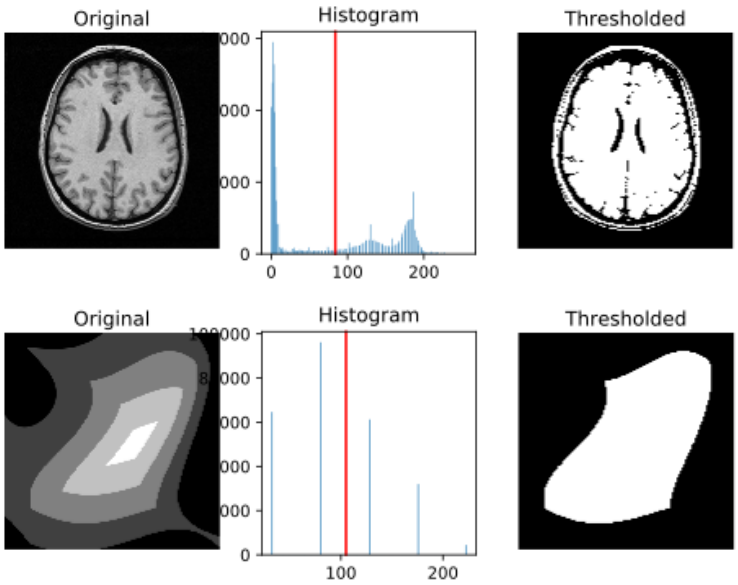
Peut-être le minimum c'est 7 classes. Avec 6, on observe seulement la couleur jaune et des gris. Et avec 5 seulement des gris.

K-means pour 7,6 et 5 classes, respectivement.

Nous pouvons passer l'image pour un filtre, pour équaliseur les niveaux de gris pour un même classe (remorver les rayons et les points). Après nous appliquons le k-means sur l'image filtre, avec k=3.

4. Seuillage automatique : Otsu

Nous cherchons pour optimiser la meilleure seille qui nous donne le major k : multiplication des probabilités á priori, fois la différence de moines entre-classes.



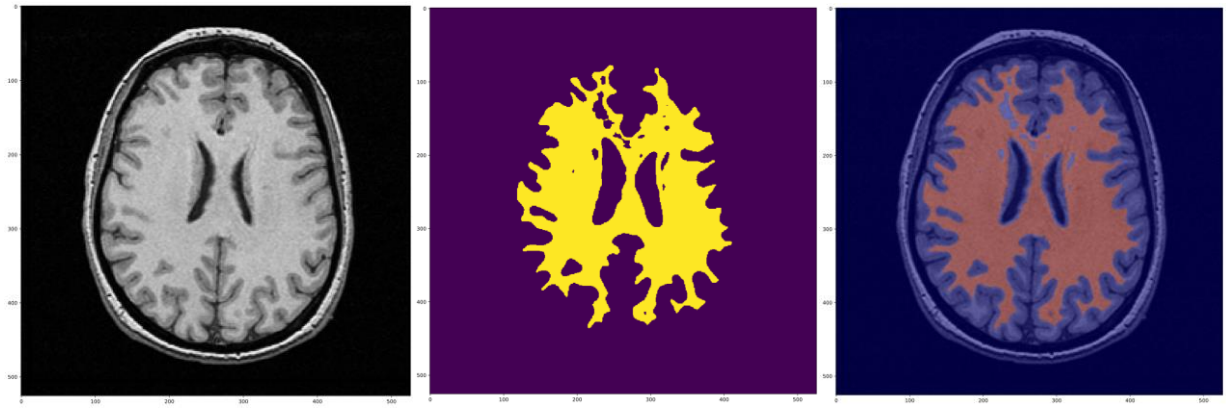
Otsu sur l'image cerveau.tif et sur pyramide.tif

L'algorithme nous donne un bon résultat pour les images qu'ont en fait deux classes (comme le cerveau ou les cellules), et des résultats qui ne se marchons pas aux autres images.

Oui. Nous pouvons utiliser sur l'image de la norme du gradient. Les normes plus grandes (plus claires), qui sont les contours devrait être sépare du noir (qu'est pas de contour).

```
def otsu_thresh(im):

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

    maxt1=0
    maxt2=0
    maxk=0

    for t1 in range(256):
        for t2 in range(256):
            w0=0
            w1=0
            w2=0
            m0=0
            m1=0
            m2=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*w2*(m0-m1)*(m0-m1)*(m0-m2)*(m0-m2)*(m1-m2)*(m1-m2)

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

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

5. Croissance de régions



L'image cerveau.tif originale ; le Mask et le résultat du Croissance de régions

Le contraint c'est si la différence absoute entre la moyenne du voisinage moins le moyenne de l'image c'est moins qui un seille fois l'écart-type.

Quand nous augmentons le seuille, nous allons ajouter plus de pixels à l'objet (des régions plus continues au fin), et si nous le diminuons, on ajout moins de pixels.

La position du germe initiale. Oui, si le germe initiale c'est dans la matière grise.