Benjamin Ternot 10/10/2022

TP segmentation

1 Détection de contours

1.1 Filtre de gradient local par masque

Comparé au filtre dérivé, le filtre Sobel calcule non seulement le gradient dans une direction, mais lisse également dans la direction orthogonale, le rendant moins sensible au bruit.

Il est nécessaire d'effectuer un filtrage passe-bas de l'image avant d'appliquer le filtre car sinon, l'image possède tellement d'irrégularités que l'on trouve des valeurs très grandes pour la norme du gradient et ainsi, nous ne trouvons pas le bon contour.

Comme on peut le constater ci-dessous, de bons contours globaux ont été obtenus (seuil 0,1). La plupart des cellules sont solides (continues) et relativement petites. Les cellules avec différentes nuances de gris posent un léger problème. L'augmentation de ce seuil crée des contours discontinus (moins de contours que ceux réellement détectés) tandis que l'abaissement du seuil entraîne la détection de contours qui ne sont que du bruit.

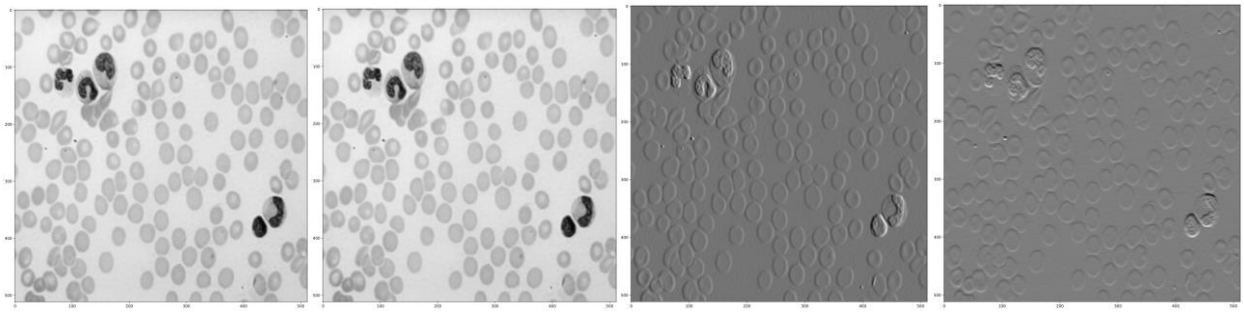
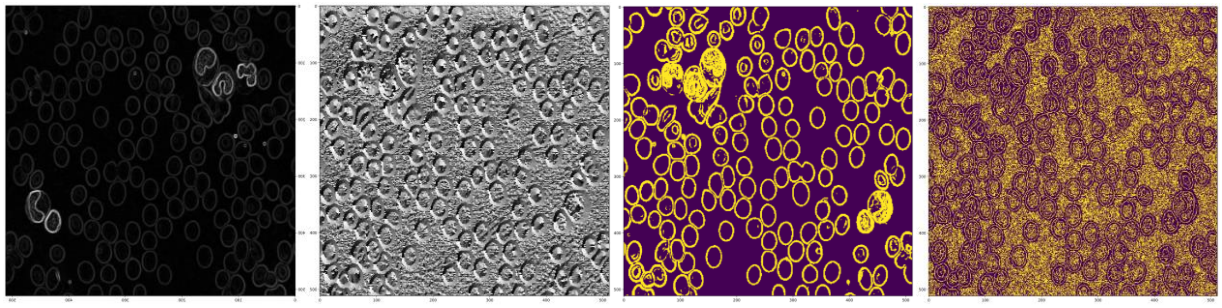


Image originale | Filtrée par passe-bas | Dérivée partielle en x | Dérivée partielle en y



Norme gradient | Direction gradient | Norme seuillée | Maxima gradient dans sa direction

1.2 Maximum du gradient filtré dans la direction du gradient

La fonction **maximaDirectionGradient** optimise les contours selon l'interpolation linéaire entre la norme et le gradient.

L'augmentation de ce seuil crée des contours discontinus (moins de contours que ceux réellement détectés) tandis que l'abaissement du seuil permet la détection de contours qui ne sont que du bruit.

Un seuil de 0,1 donne (empiriquement) de bons résultats : on cherche un compromis entre continuité du contour et robustesse au bruit.

1.3 Filtre récursif de Deriche

```
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] - b1[j-2]*ae**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] - b2[j+2]*ae**2
66          b2[nc-2]=b2[nc-3]
67          b2[nc-1]=b2[nc-3]
68          gradx[i,:]=c*ae*(b1-b2);
69
70      return gradx
71
```

Un alpha plus élevé augmente la sensibilité à la détection des contours et donc la sensibilité au bruit. La réduction de l'alpha trouvera des contours avec des bords plus doux.

La complexité de l'algorithme est $O(n^2)$ (nombre de lignes multiplié par le nombre de colonnes dans l'image) et ainsi la complexité ne dépend pas de alpha (si on considère que les multiplications pour des nombres de cet ordre de grandeur n'influent pas sur la complexité).

Le filtre de Deriche est utilisé pour son effet similaire au filtre de Sobel.

1.4 Passage par zéro du laplacien

L'augmentation du paramètre alpha entraîne des contours de bruit. En le réduisant, on obtient des formes supplémentaires délimitées par un nouveau contour.

Une principale différence avec ce nouveau procédé est que le laplacien donne des contours fermés, alors que les contours précédents pouvaient donner des lignes ouvertes.

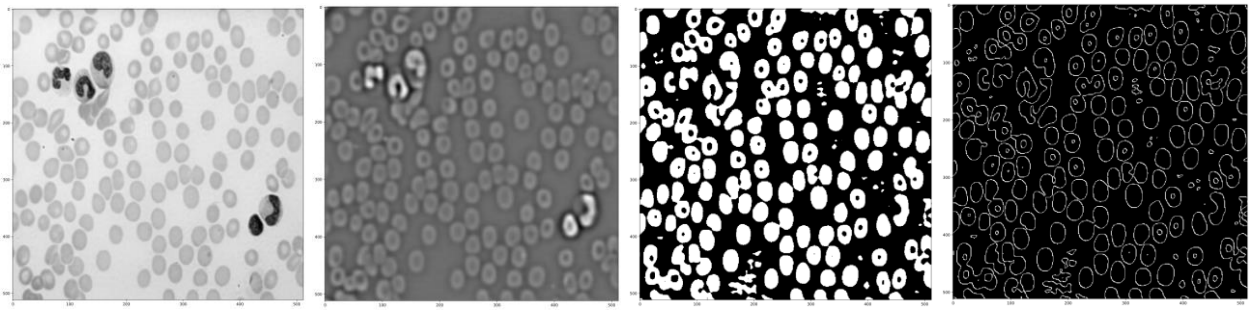
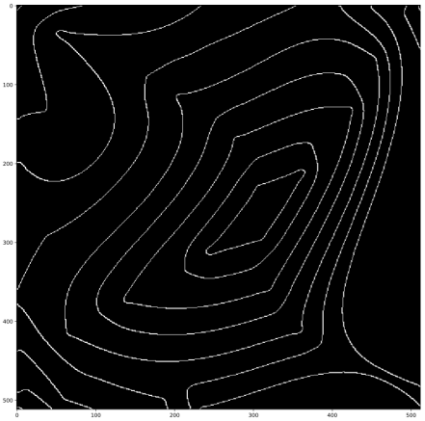


Image originale | Laplacien | Image binarisée | Contours fermés

Sur l'image pyramide.tif, afin de supprimer les « faux » contours détectés (cf ci-dessous, sur les coins par exemple), on peut au préalable effectuer un filtrage gaussien.



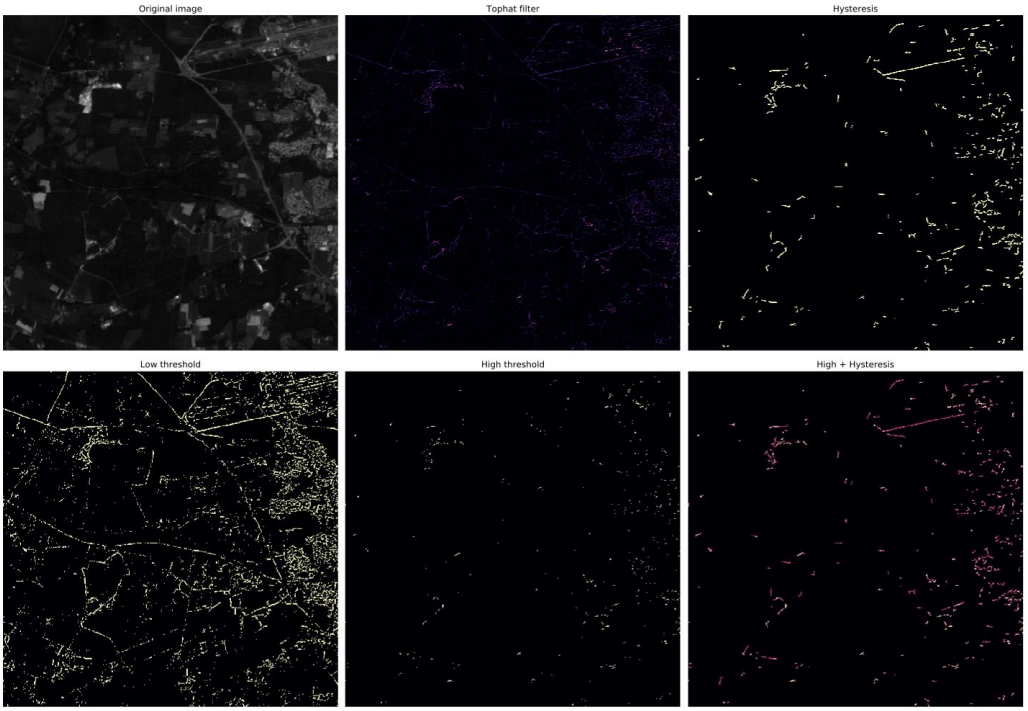
1.5 Changez d'image

Pour segmenter l'image pyra-gauss.tif, je choisirais le filtre récursif de Deriche.

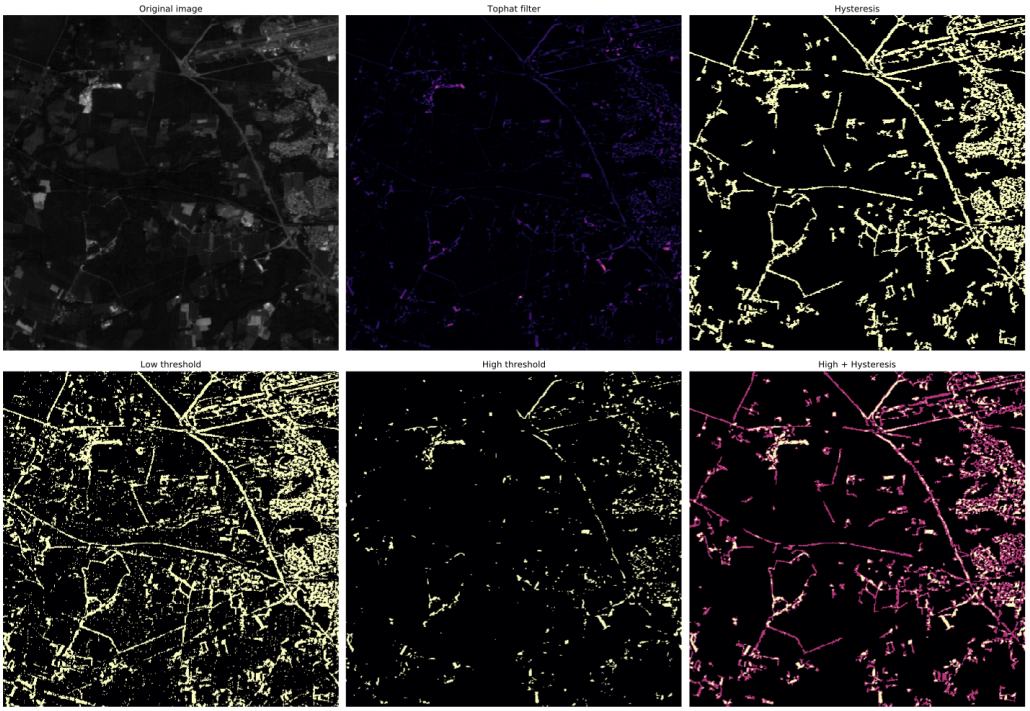
En pré-traitement, on peut effectuer un filtrage (médian ou bilatéral par exemple), et en post-tratement par le filtre de Deriche, on peut effectuer un seuillage, un maxima local du gradient, la méthode hystérésis, poursuite et fermeture.

2 Seuillage avec hystérésis

2.1 Application à la détection de lignes



Résultat après filtre chapeau haut de forme



Rayon = 3

En augmentant le rayon, on détecte plus de lignes.

Les seuils *low=2* et *high=9* donnent selon moi le meilleur résultat.

La méthode appliquée à 1.1 donne de meilleurs résultats avec la méthode d'hystérésis. Les bords dont plus cohérents et il y a moins de bruit que le résultat précédent.

3 Segmentation par classification : K-moyennes

3.1 Images à niveaux de gris

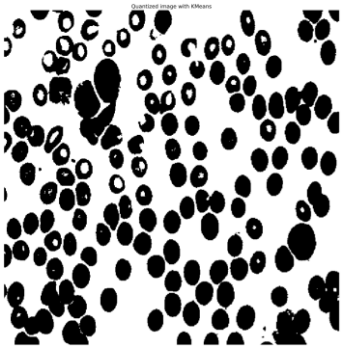
En général, il en résulte une bonne segmentation des cellules. Néanmoins, si l'intérieur de la cellule est très clair, il y a une erreur de classification. On peut pour y remédier essayer d'appliquer par exemple une fermeture géométrique.

La méthode **k-mean** fait un calcul de probabilités en fonction de la distance aux centres choisis. Par conséquent, démarrer l'algorithme avec un centre dans chaque cluster donne de meilleurs résultats.

La méthode **random** est aléatoire, et ne nécessite pas d'initialisation particulière.

Avec **ndarray**, on peut définir les centres comme on le souhaite.

Dans **cell.tif** à 2 classes, la classification est constante car il y a 2 classes très bien marquées. Cependant, lorsque l'algorithme est appliqué à des images où les différences entre les classes sont moins évidentes, ou contiennent plus de classes, il devient non constant (même avec **k-means**).



k-mean pour 2 classes pour **cell.tif**

Le problème avec **muscle.tif** est qu'il y a 3 classes bien marquées : fibres très noires, fibres grises, pas de fibres (transparent). La méthode fonctionne bien pour la classification des fibres sombres et des zones claires, mais les fibres grises ne sont pas des zones de gris constant et ainsi ne sont pas correctement détectées. Il faut pour cela au préalable lisser au maximum les zones qui devraient apparaître sensiblement constantes.

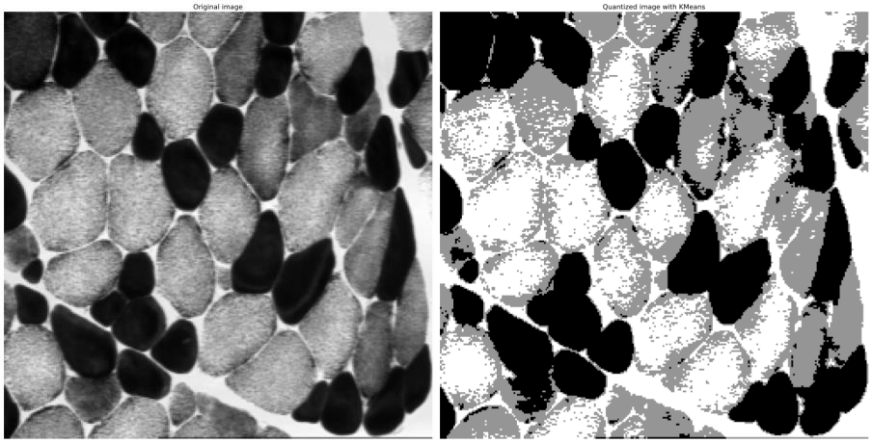


Image **muscle.tif** originale **k-mean** avec 3 classes

Le filtrage de l'image d'origine améliore les résultats de l'algorithme. En effet, cela permet d'atténuer le bruit et les changements très brusques de niveaux de gris, et ainsi une meilleure détection par la méthode **k-mean**.

3.2 Image en couleur

Avec 10 classes, nous obtenons une représentation assez fidèle de l'image. Seules les ombres et les détails ont disparus, et quelques couleurs assez proches se mélangent (le rose est détecté soit comme mauve, soit comme rouge sombre).

Image originale

Image après classification en 10 classes

Avec 5 et 6 classes, les couleurs sont assez mal représentées, il manque surtout les teintes rouges (moins présentes). A partir de 7 classes, le résultat commence à représenter correctement l'image, même si les couleurs ne sont pas aussi bien représentées qu'avec plus de classes.

7 classes

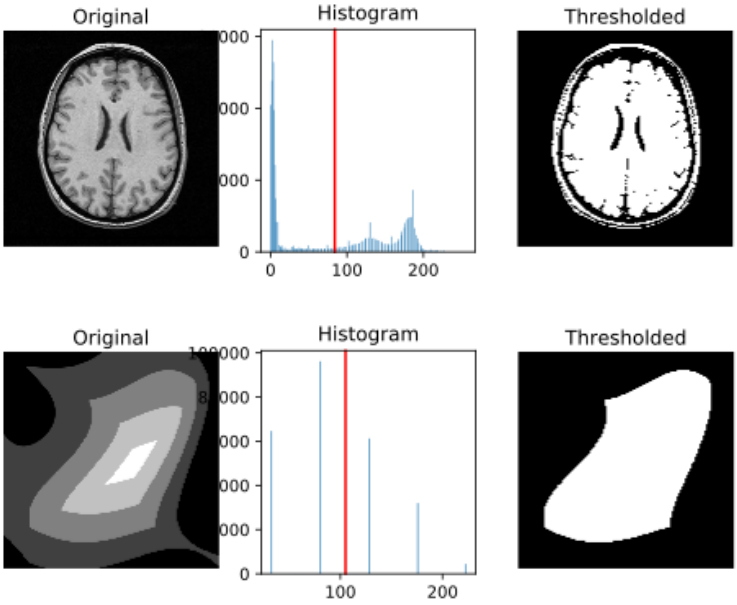
6 classes

5 classes

Pour obtenir un bon résultat, nous pouvons filtrer afin d'égaliser les niveaux de gris d'une même classe, en retirant les irrégularités, et appliquer la méthode **k-mean**, avec k=3.

4 Seuillage automatique : Otsu

L'algorihme fonctionne bien pour les images qui possèdent 2 classes (cerveau, cellules,…), mais ne permet pas de détecter plus de classes (ex : pyramide).



Cette méthode fonctionne l'image de norme du gradient car les normes plus grandes, représentant les contours sont séparées des plus faibles, qui n'en sont pas.

5 Croissance de régions

Quand nous augmentons le seuil, les régions sont plus continues, mais sont trop grandes. En diminuant, c'est l'inverse.