I- Détection de contours

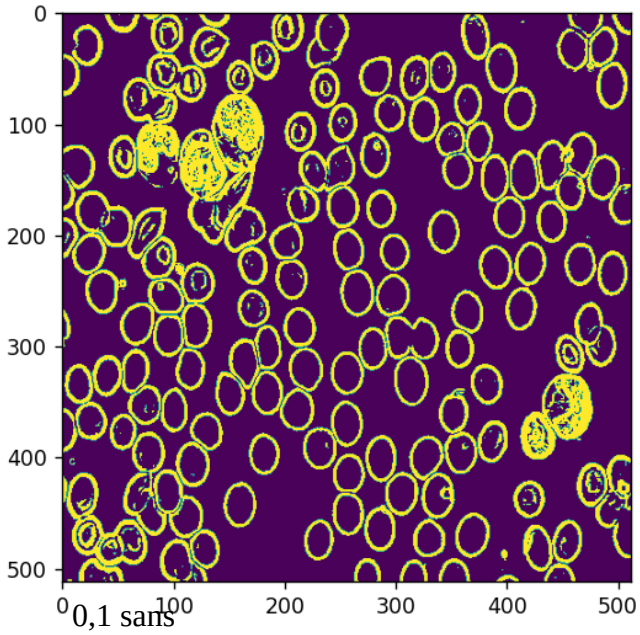
<u>Pour le bruit, on prend un bruit gaussien de variance 1</u>

1. Filtre de gradient local par masque

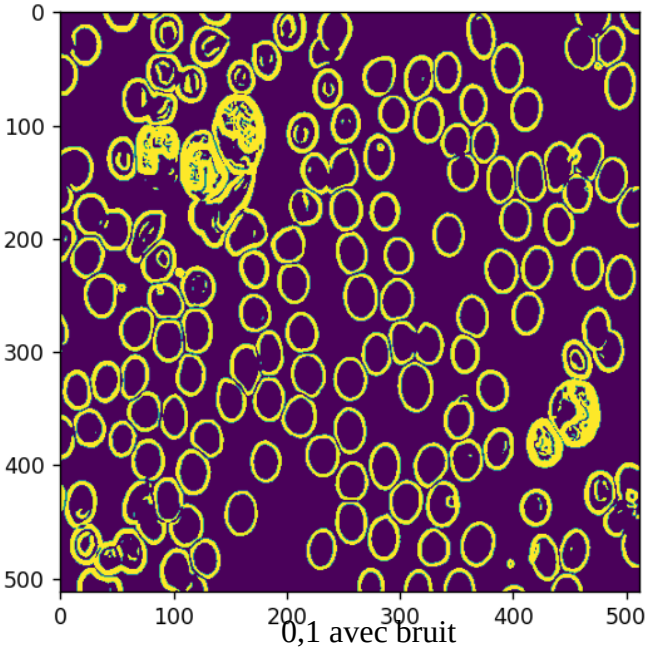
- Le filtre de Sobel permet de calculer un gradient en prenant en compte les pixels en diagonal du pixel considéré pour garantir de meilleurs contours, là où le filtre différence ne considère que le pixel suivant.

- Il n'est pas nécessaire d'effectuer un passe-bas avant d'utiliser le filtre de Sobel mais si jamais l'image venait à être bruitée, le résultat serait impacté et alors, avoir filtré l'image auparavant permet de réduire le bruit tout en gardant un résultat montrant les contours de l'image.

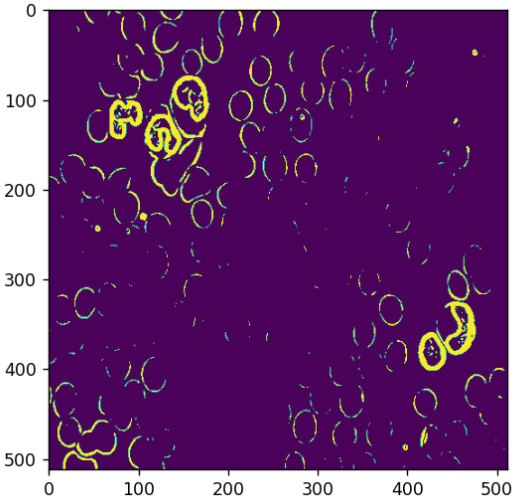
- Lorsque le seuil de gradient vaut 0,1 (valeur initiale) le bruit en semble pas avoir d'effet sur les contours et ceux ci sont bien continus bien que plutôt épais. Augmenter le seuil a pour effet d'ignorer encore plus le bruit et d'affiner les contours mais cela finit par les briser car la norme du gradient n'est pas constante sur ceux-ci. Au contraire, diminuer le seuil permet au gradient du bruit de s'exprimer ce qui dégrade les contours les rendant plus épais jusqu'à créer un image complètement bruitée dans laquelle on ne les distingue plus.



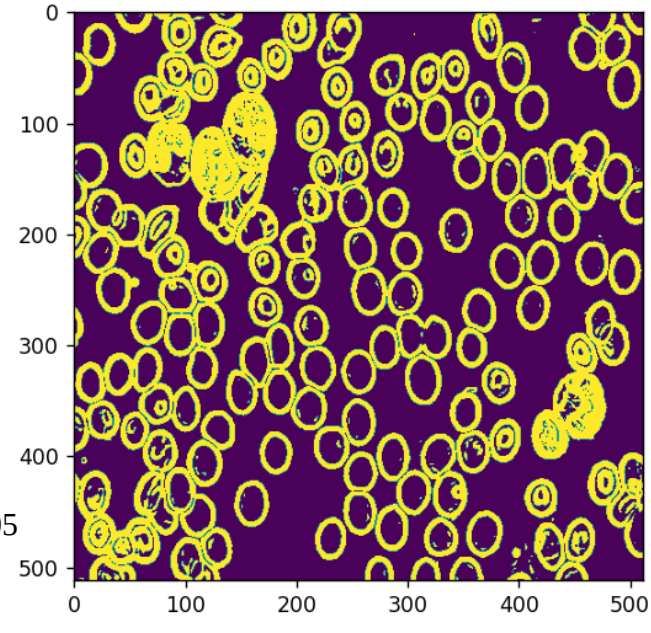
0,1 sans bruit



0,1 avec bruit

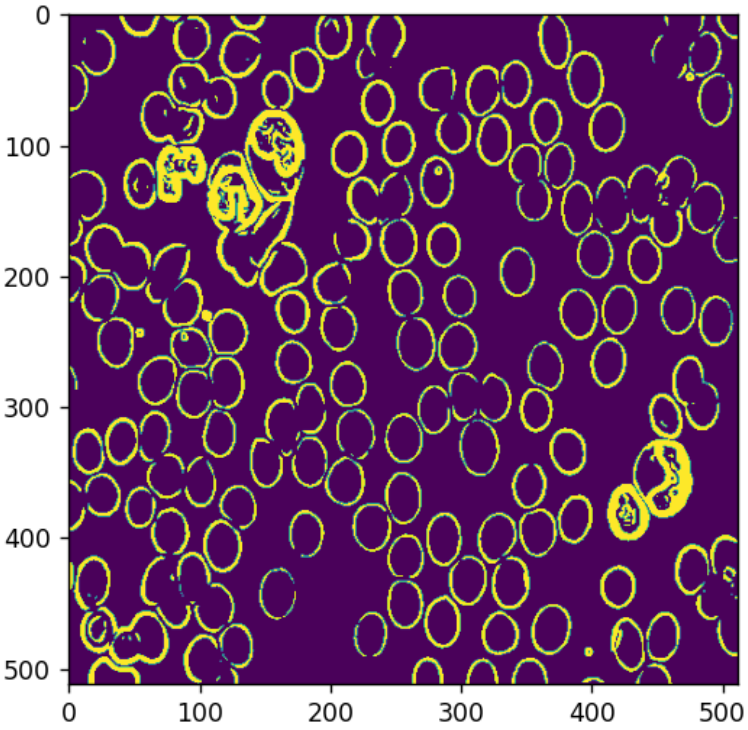


0,2

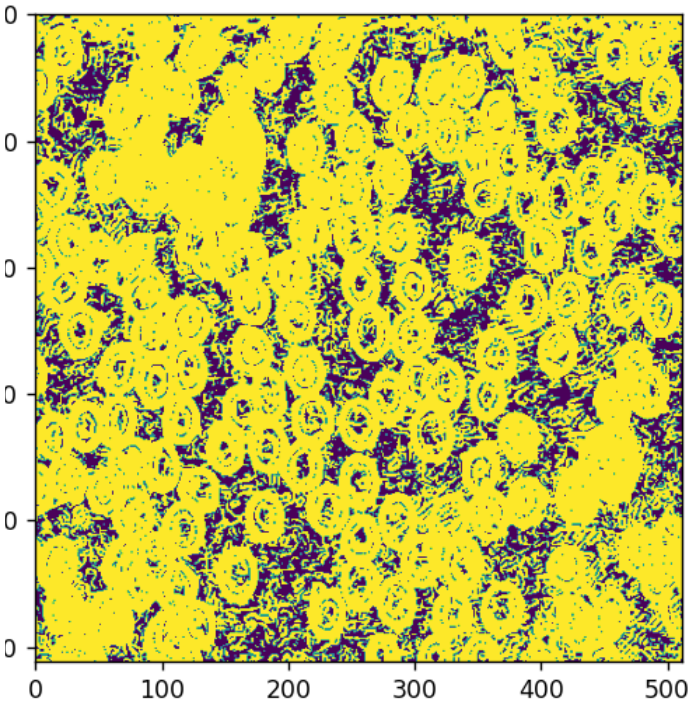


0,05

0,14



0,01

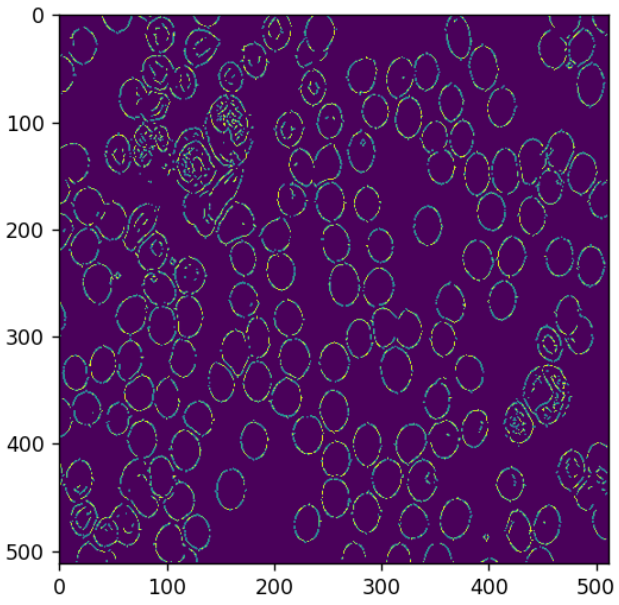


2. Maximum de gradient filtré dans la direction du gradient

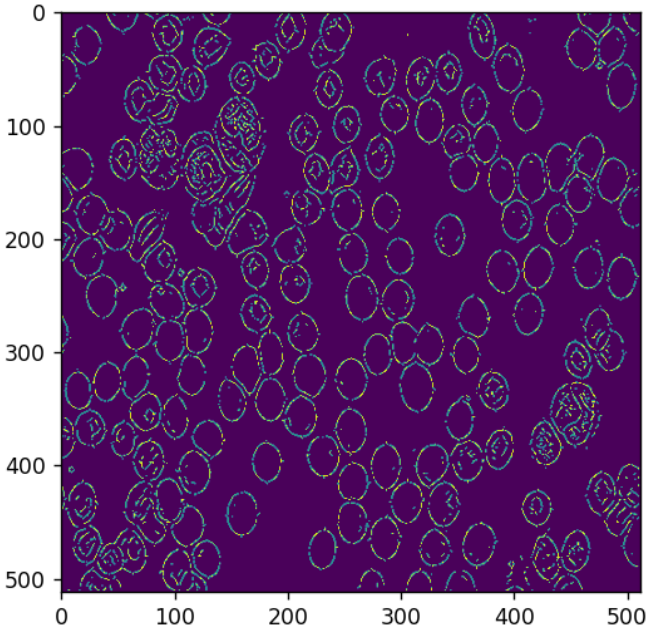
- La fonction maximaDirectionGradient permet d'affiner les contours pour que ceux-ci aient une épaisseur d'un pixel.

- Tout comme précédemment, augmenter le seuil permet de faire disparaître les contours de norme trop faible mais créera aussi des discontinuités chez les autres. Diminuer le seuil ne les rendra pas plus épais mais permettra au gradient du bruit d'apparaître.

- Les seuils 0,2 et 0,05 présentent soit trop de détails, soit trop peu (les contours ont presque disparus), le seuil optimal se situe autour de 0,1 , on retient 0,13 qui permet de supprimer des traces à l'intérieur des contours sans dégrader leur bordures extérieures.

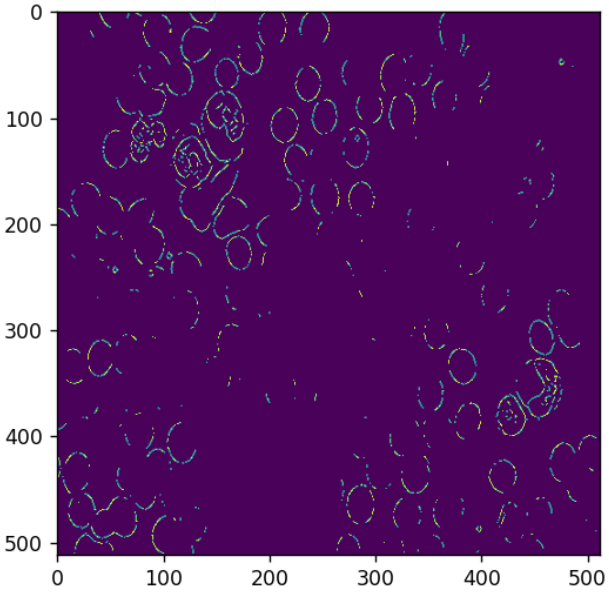


Seuil 0,1

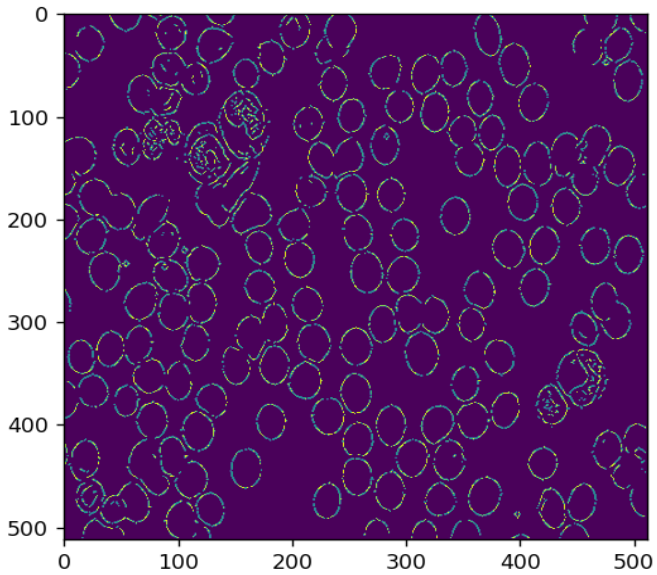


Seuil 0,05

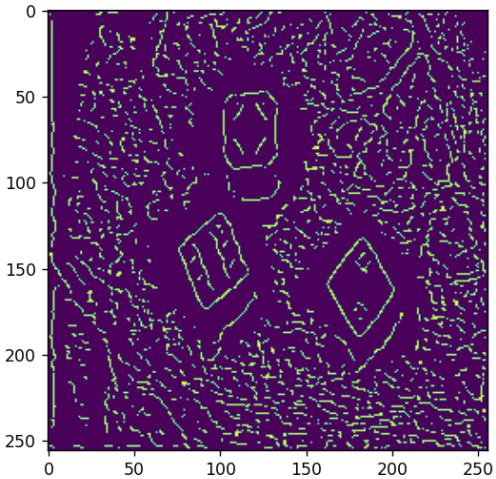
Seuil 0,2



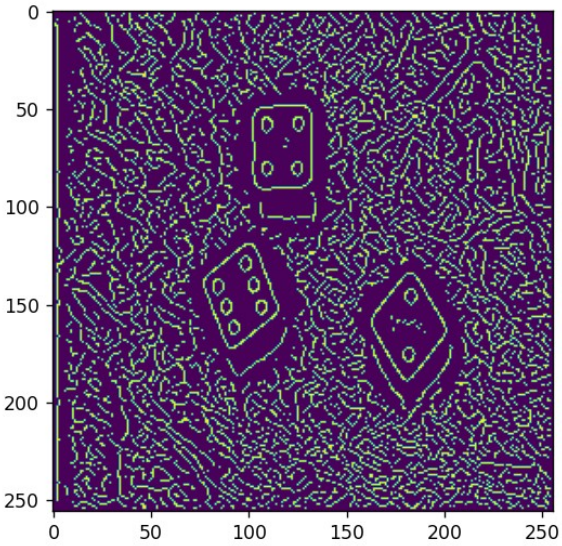
Seuil 0,13



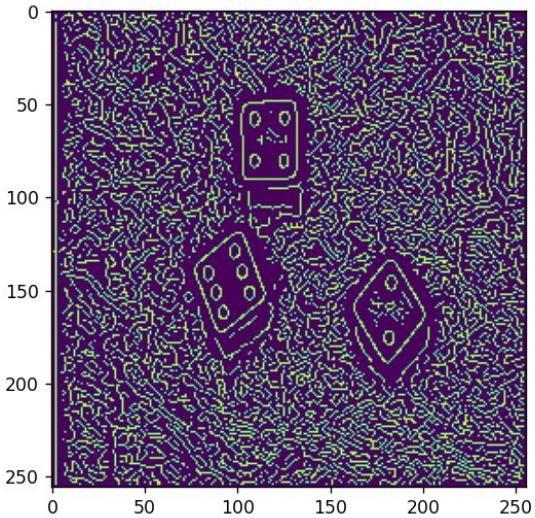
3. Filtre récursif de Deriche



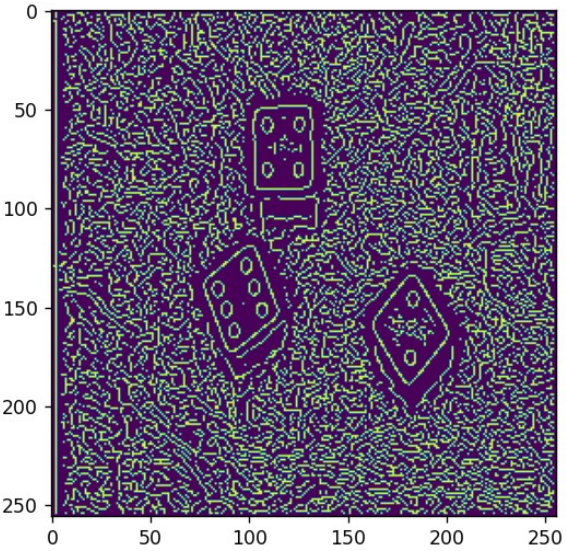
α = 0,3



α = 1



α = 2



α = 3

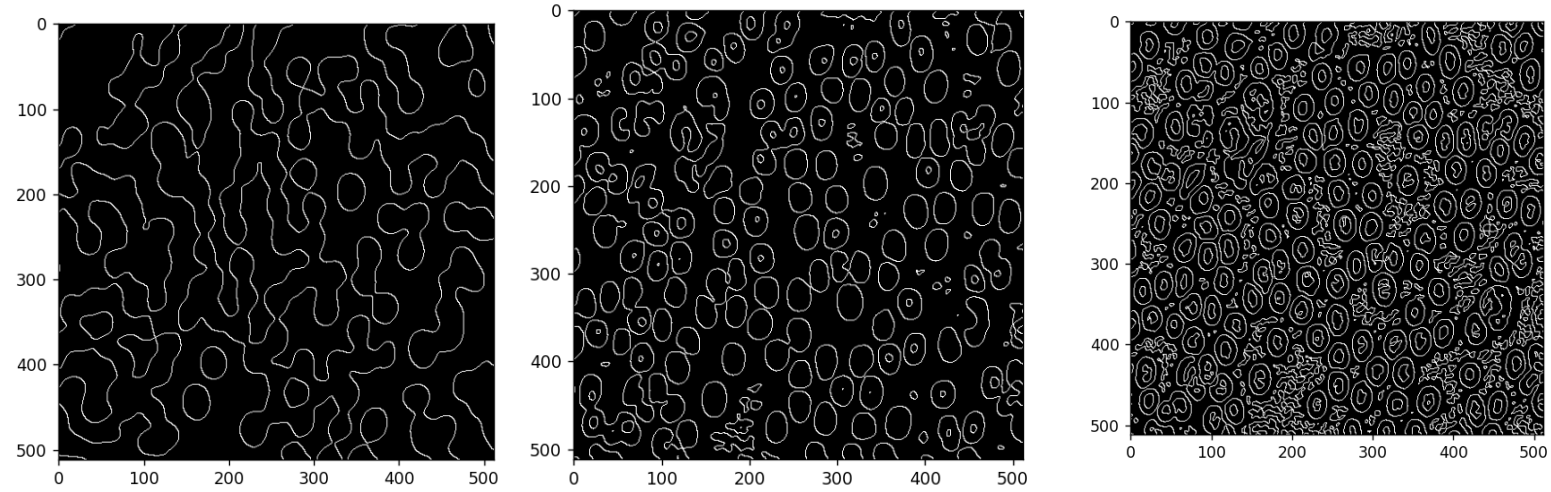
- Plus le paramètre α est grand, plus les contours sont précis (ci-dessus, au départ on ne connaît pas le score des dés) mais en échange ceux de l'arrière plan sont également de plus en plus présents. Il faudrait un moyen de les supprimer mais un seuil ne semble pas fonctionner, leur norme ayant l'air d'être proche des contours des objets nous intéressant.

- Le temps de calcul est inversement proportionnel à alpha, cela est du au fait que $e^{-\alpha}$ est inversement proportionnel à α et tend vers 0, il est donc plus rapidement arrondi à 0 par python ce qui réduit le nombre de calculs effectués.

- Les fonctions derichesmooth prennent en compte plus de valeurs de l'image d'origine, tout comme le filtre de Sobel comparé au filtre différence, afin d'avoir des contours plus réguliers.

4. Passage par zéro du laplacien

α = 0,2 α = 0,5 α = 1



- Plus le paramètre α est élevé, plus on voit de détails apparaître, il faut trouver un équilibre entre avoir les vrais contours (α = 0,2 ne le fait pas) et ne pas en créer et/ou garder des contours non intéressants (ce que α = 1 fait)

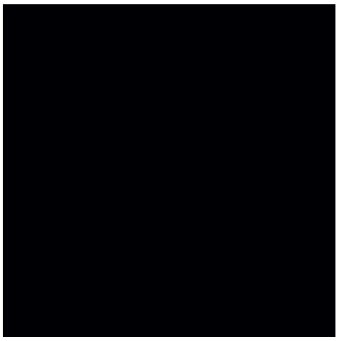
- La principale différence entre la méthode d'annulation du laplacien et les méthodes précédentes appliquées sur l'image cellules.tif est l'apparition des contours des noyaux des cellules. Auparavant, leur différence de niveau de gris trop faible par rapport au reste de la cellule les empêchait d'apparaître.

-

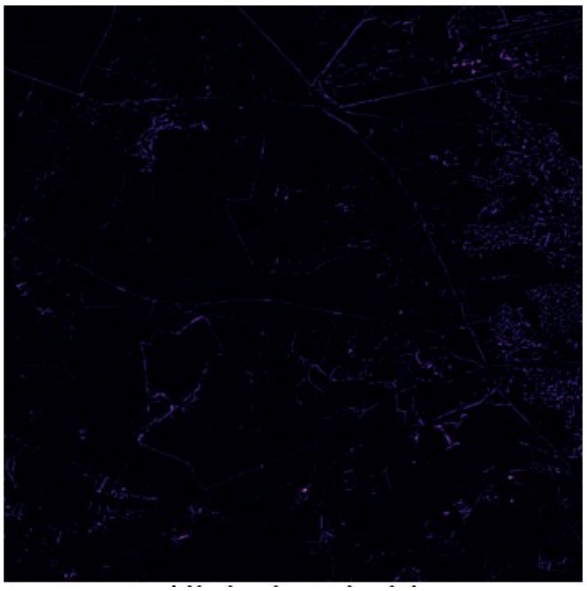
5. Changez d'image

II- Seuillage avec hystérésis

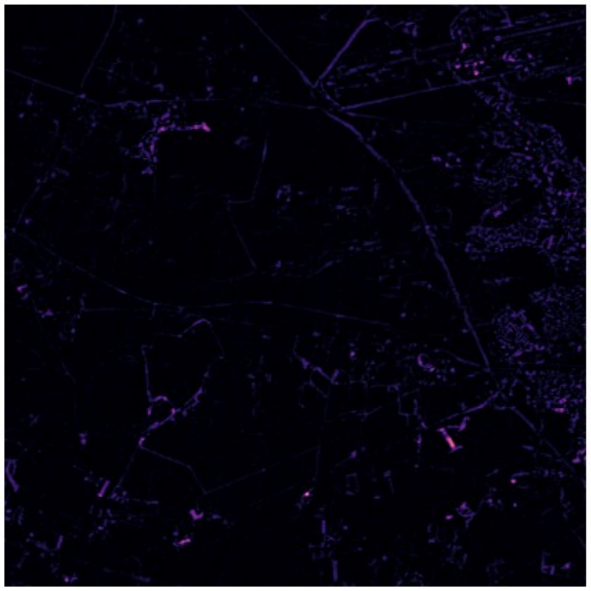
1. Application à la détection de lignes



Rayon = 1
(image noire)



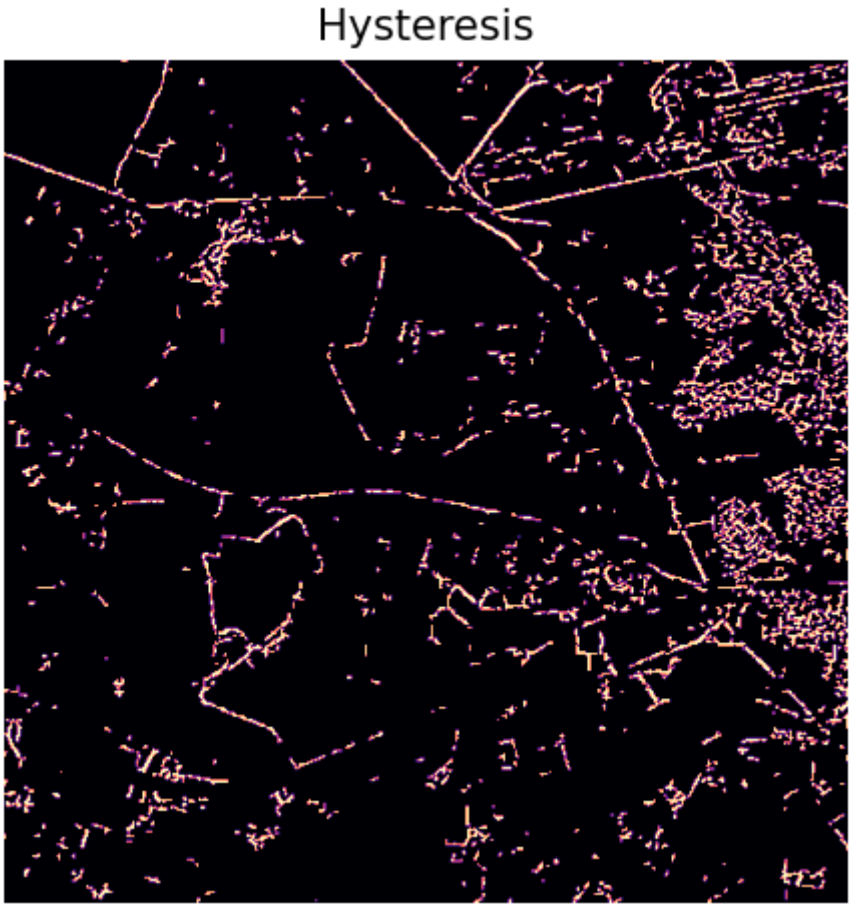
Rayon =3



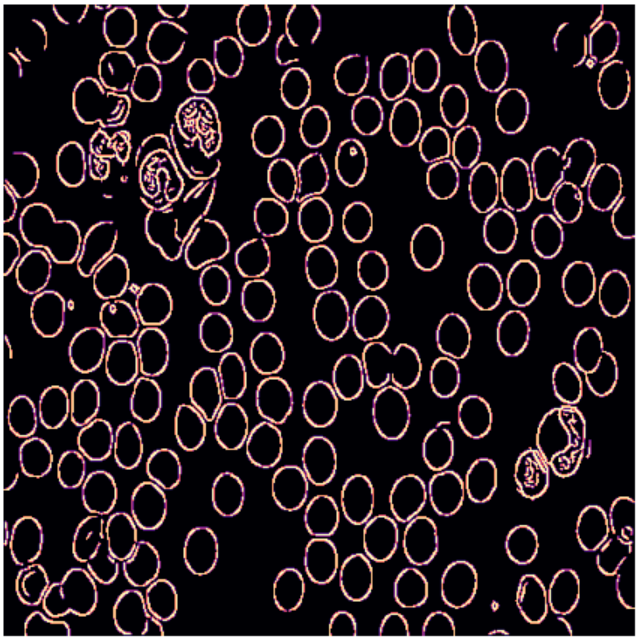
Rayon = 5

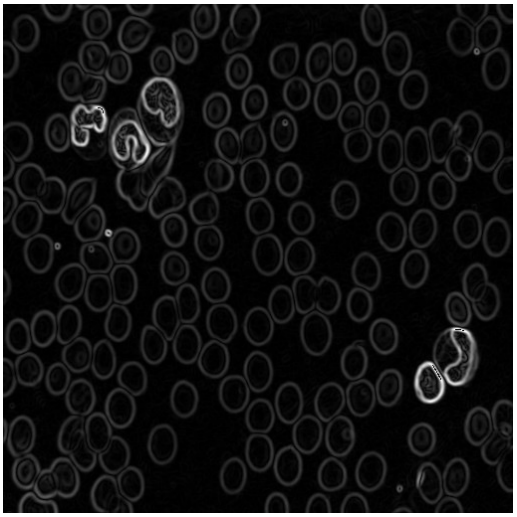
- Augmenter le rayon permet de détecter plus de lignes mais également des éléments moins intéressant qui viennent parasiter l'hystérésis plus tard. Diminuer le rayon fait détecter moins de lignes jusqu'à ne rien détecter du tout pour un rayon égal à 1.

- Diminuer le seuil bas peut faire apparaître des détails inutiles et augmenter le seuil haut peut éliminer des lignes importantes sur l'image, il faut chercher un équilibre : Le couple (seuil bas, seuil haut) = (2,4) donne le résultat suivant où on trouve un peu de bruit et la grande majorité des contours principaux.



En appliquant la méthode d'hystérésis à l'image des contours obtenue grâce au gradient de Sobel avec des seuils 5 et 13 on obtient l'image suivante : Les contours sont affinés et ceux des noyaux des cellules ainsi que les parasites dans ces dernières ont disparus.

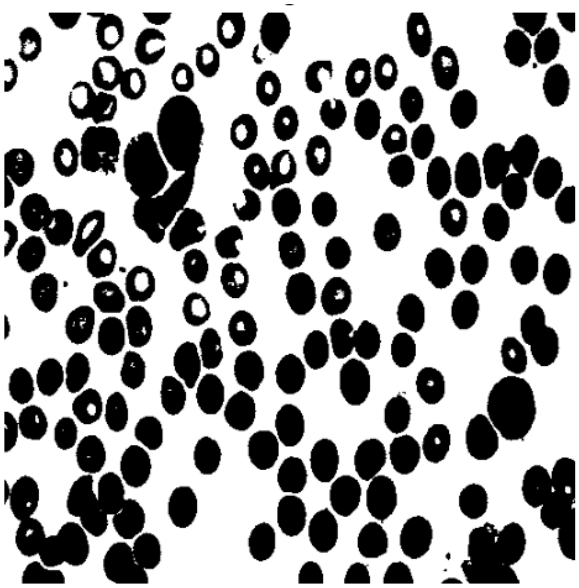




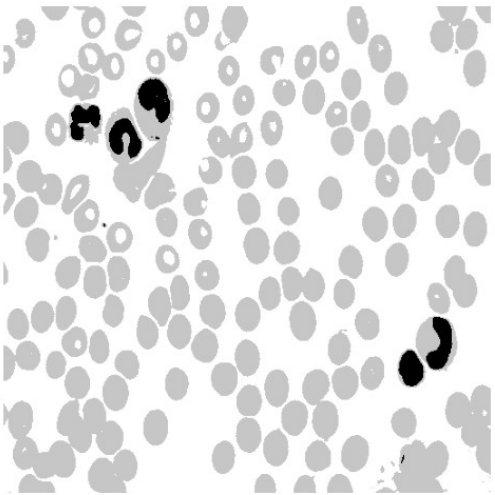
III- Segmentation par classification : K-moyennes

1. Image à niveaux de gris

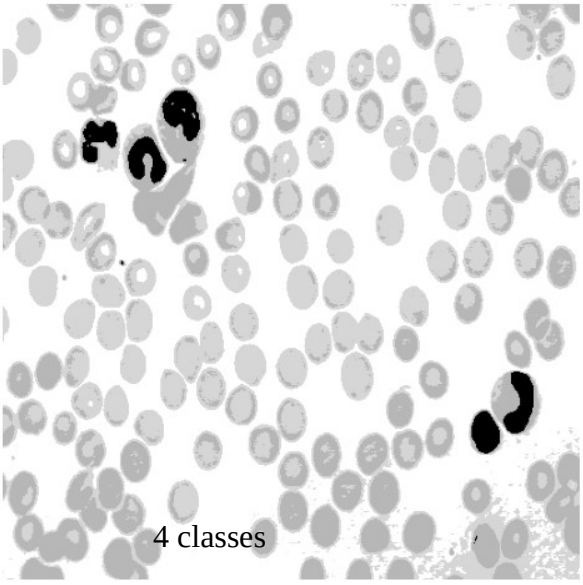
- La classification en deux classes donne l'image suivante. On peut distinguer les cellules de l'arrière plan même si certaines ne sont pas fermées et celles superposées sont invisibles, on peut augmenter le nombre de classes à 3 ou 4 ce qui devrait régler le problème.

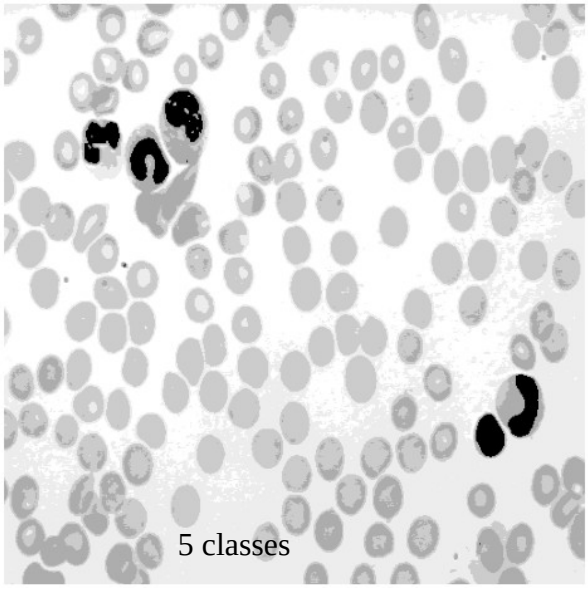


Voici l'image obtenue avec 3 classes, on détecte bien toutes les cellules ainsi que leurs noyaux :



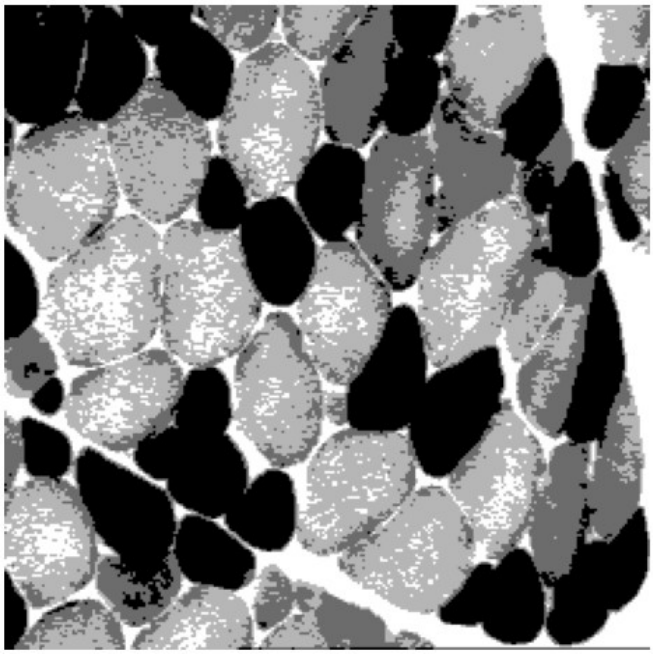
-L’initialisation par défaut des centres des classes donne l’image ci-dessus et se fait en appliquant l’algorithme greedy k-means++ qui consiste à selectionner les meilleurs marqueurs sur plusieurs itérations en se basant sur leur contribution à l’inertie de l’image obtenue grâce à une loi de probabilité empirique.
(cf https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)
Ils peuvent également être choisis parmi les lignes de l’image de manière aléatoire en choisissant l’option init=’random’
Enfin on peut spécifier soit une matrice contenant leur coordonnées, soit une fonction qui les initialise





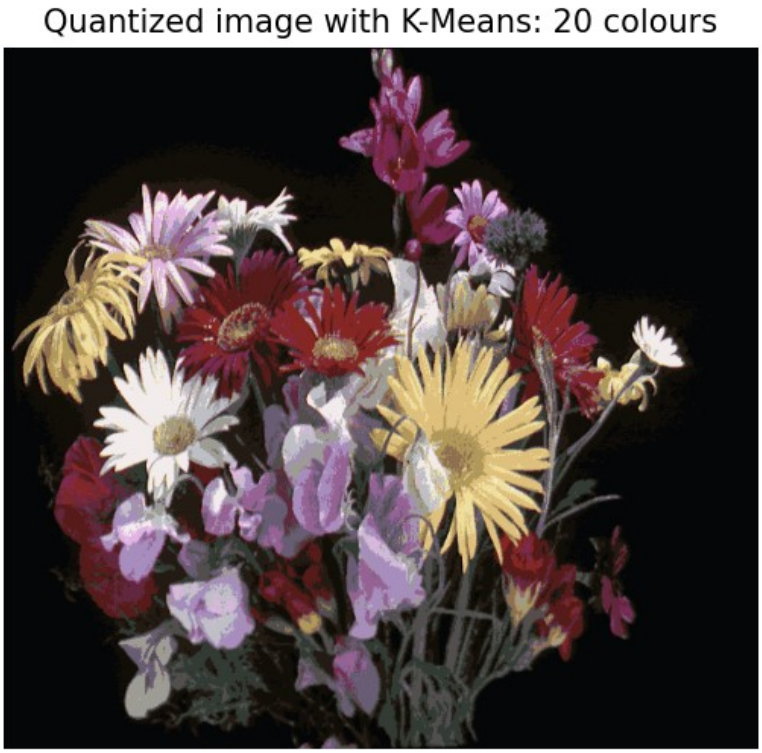
- Les classes détectées sont les mêmes avec initialisation aléatoire, on en déduit que la classification est stable.

- La segmentation en deux classes de muscle.tif fait disparaître un deux types de cellules dans l’arrière plan. Elles réapparaissent avec plus de classes mais contrairement à cellules.tif, elles ne sont pas unicolores. Appliquer un filtre moyenneur ou médian au préalable permet de retirer la texture ce qui rend la classification meilleure.



2. Image en couleur

- Les couleurs sont ajoutées en partant du coté rouge du spectre. Ainsi les couleurs vertes et bleues ne sont pas bien apparentes, les objets de telles couleurs font cependant partie de classes séparées des autres. Les autres couleurs sont également moins lumineuses.



- Avec 20 classes environ, on retrouve presque les couleurs d'origine pour la plupart des fleurs, hormis la fleur violette en haut à droite qui ne récupère pas sa couleur même avec 50 classes.

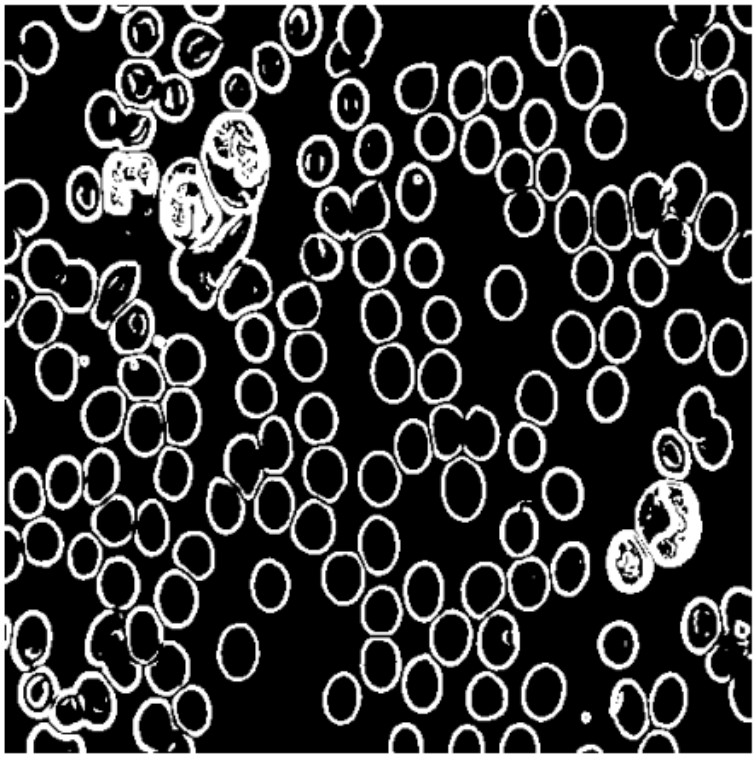
- Pour retrouver le nombre de planches-mère ayant servi à créer carte.tif on augmente simplement le nombre de classes jusqu'à obtenir l'image originale. On obtient ce résultat pour 6 classes.

IV- Seuillage automatique : Otsu

- Dans ce fichier on cherche à réaliser la meilleure séparation de l'image en deux classes en partant de son histogramme et en maximisant un critère dépendant du nombre de pixels dont la valeur est sous le threshold et au dessus.

- Cette méthode fonctionne bien sur des images telles que cell.tif mais comme soulevé précédemment, muscle.tif pose un problème, certaines cellules ne sont pas détectées.

- Utiliser la méthode otsu sur une image de norme de gradient rend la segmentation bien plus claire, voici le résultat obtenu sur une image de norme de gradient de Sobel, les contours ne sont plus flous

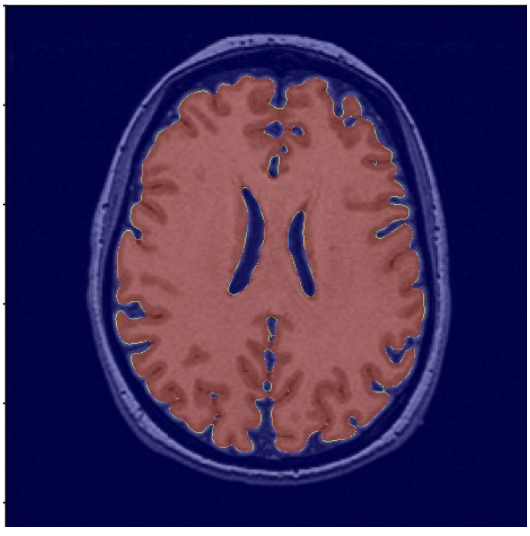


- Pour obtenir la méthode à trois classes, on effectue une boucle pour t1 allant de 0 à 256 dans laquelle on effectue une boucle sur t2 de t1 à 256 et on cherche à maximiser une quantité dépendant du nombre de pixels dont l'intensité se trouve entre 0 et t1, entre t1 et et t2 et entre t2 et 256 ainsi que des sommes pondérées par leur intensité de ce nombre de pixels en s'inspirant de la forme donnée pour deux classes.

V - Croissance de régions

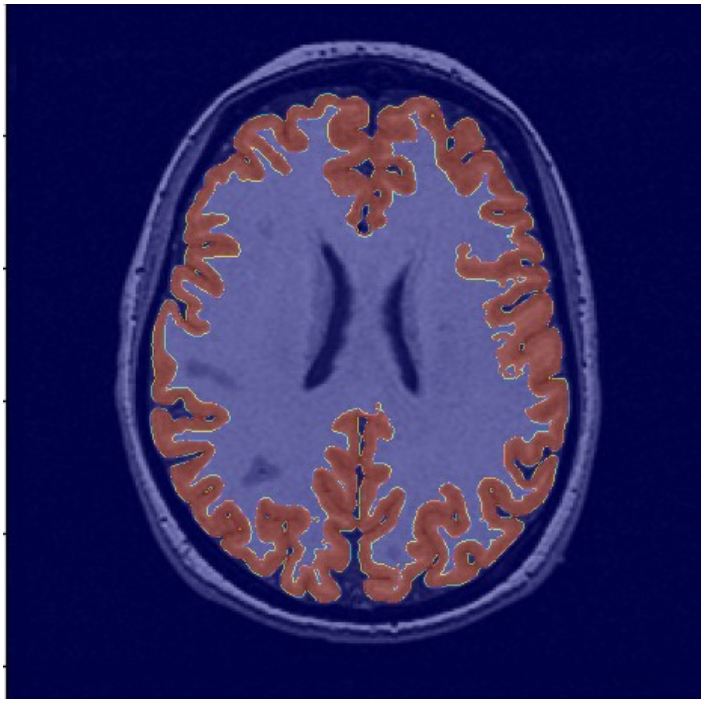
- Si la valeur moyenne des pixels autour d'un pixel donné de l'image plus une valeur choisie (thresh*s0) à l'avance est plus grande que la valeur moyenne des pixels autour du germe initial on ajoute ce pixel à l'objet.

- Le paramètre thresh permet, en augmentant, d'être plus tolérant sur les pixels à ajouter, jusqu'à trop en ajouter ce qui dénature l'objet. Avec un threshold de 10, les replis sur les bords du cerveau ne sont plus apparents.



- Afin de segmenter correctement la matière blanche il faut régler le paramètre thresh afin d'ajouter le plus de pixels possible, sans en prendre en dehors de la région à segmenter. On peut également diminuer le rayon ce qui ralentit l'algorithme mais évite ce genre d'erreurs.

- On peut tenter de segmenter la matière grise en positionnant le germe initial dans une zone grise mais il est difficile de régler les paramètres pour ne pas incorporer de matière blanche et il est impossible de récupérer la matière grise entourée de matière blanche.



Thresh = 10
Rayon = 2
Y0=346
X0=130

- Ce script suppose donc que les objets à segmenter sont connexes, ce qui ne paraît pas être très exigeant à première vue mais nous avons ici un parfait contre-exemple.

- Toutes les méthodes de segmentation de la première partie basée sur des gradients ou des annulations donnent le même résultat car étant des objets séparés, on observe alors une annulation du laplacien/une grande valeur de gradient.

- La croissance de région est nécessaire pour segmenter des objets dont la couleur est proche et étant superposés, les autres méthodes donneraient un seul et même objet : ils ont la même couleur et le gradient varie assez peu au niveau de l'endroit où ils se superposent, sauf si l'un est complètement dans l'ombre de l'autre.