TP IMA201(a) – Segmentation

Jeanne Malécot

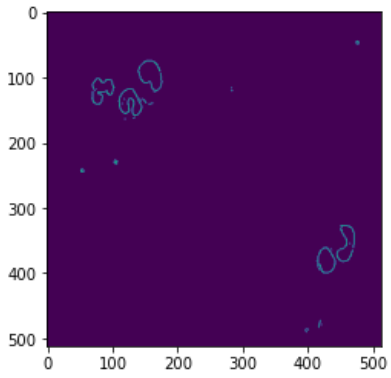
1.1 Le filtre de Sobel, contrairement au filtre différence, permet d'obtenir l'orientation du contour à parti du gradient vertical et du gradient horizontal.

Un seuil plus bas donne des contours plus épais, mais laisse aussi des informations inutiles à l'image. L'épaisseur des contours permet de toutefois de distinguer les contours qui nous intéressent du bruit. En l'augmentant, on se sépare de ces informations mais on risque d'obtenir un contour discontinu à un certain stade.

1.2 Ce procédé permet d'affiner les contours, sans risquer de perdre leur continuité. C'est une étape nécessaire pour une éventuelle vectorisation des contours.

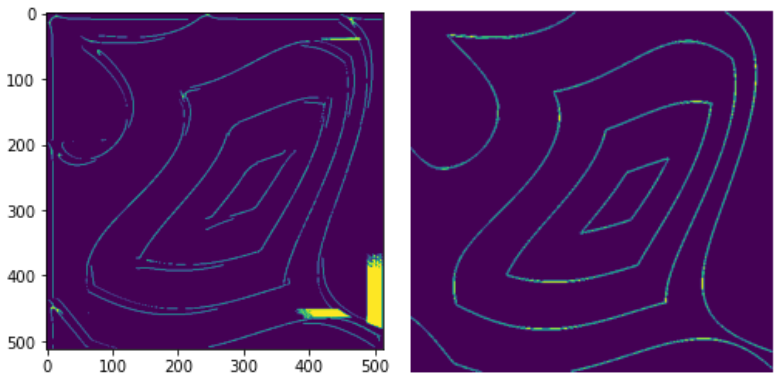
A nouveau, un seuil élevé offre une grande robustesse au bruit, puisque les contours avec une norme trop faible sont effacés, mais il peut aussi causer une discontinuité des contours, voir une quasi disparition des contours. Il est toutefois beaucoup plus important de se débarrasser du bruit avec ce procédé, puisque l'amincissement des contours empêche de distinguer les contours intéressants des autres.

Pour l'image « cell.tif », la valeur de seuil 0.5 offre un bon compromis.



1.3 La détection de contours est plus efficace avec un alpha élevé (proche de 3), et permet d'obtenir des contours plus fins et nets, tandis qu'un alpha faible donne un contour très peu précis, même dans le cas d'une image très simple.

Alpha = 0.1



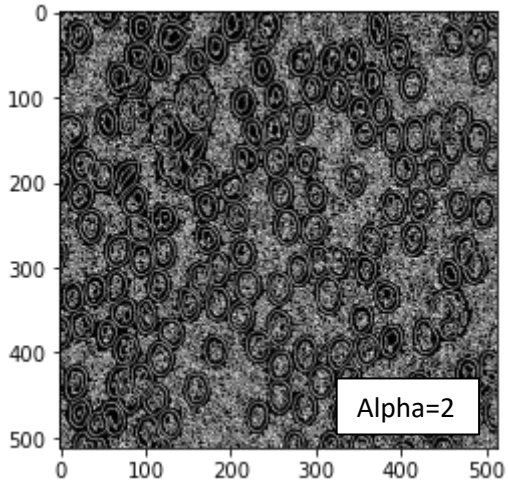
Alpha = 3

La variation d'alpha n'a pas d'influence sur le temps de calcul ; il s'agit en effet du paramètre de l'exponentielle, qui est calculée uniquement au début du programme et stockée dans la variable « ae ».

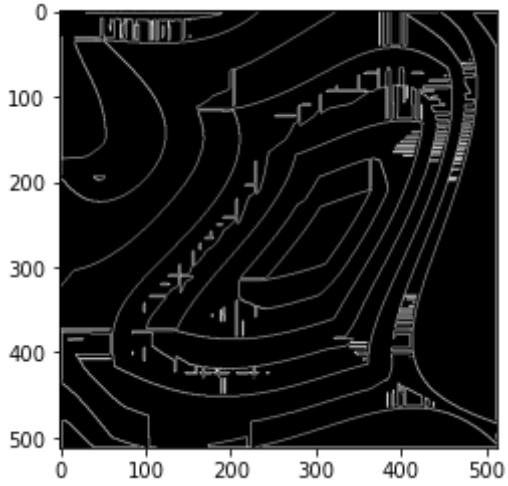
Les fonctions dericheSmoothX et dericheSmoothY sont utilisées sur l'image avant la détection des contours afin de lisser l'image, à la manière du filtre passe-bas pour Sobel. La méthode de calcul de Deriche est réutilisée, mais en ajoutant b1 et b2 plutôt qu'en les soustrayant, ce qui donne cet effet de lissage par calcul d'une moyenne.

1.4 Le paramètre alpha a à nouveau une influence sur l'épaisseur du contour détecté : plus il est élevé, plus le contour est fin.

Sur l'image « cell.tif », le résultat est moins satisfaisant qu'avec les autres méthodes : on ne distingue plus les contours intéressants des autres contours, et l'image est moins lisible (contours noirs sur fond bruité).



Pour l'image « pyramide.tif », on remarque l'apparition de faux contours :



La diminution du paramètre alpha permet d'en supprimer une partie, puisque l'image est plus lissée, mais une diminution abusive entraine également la disparition de contours important.

Aussi, il y a plus de passages à zéro du Laplacien que de maximums du gradient, donc les contours sont « dédoublés ».

1.5 Pour une image bruitée, il faut prendre un filtre peu sensible au bruit : Sobel par exemple, en choisissant des paramètres adaptés.

2. Le fait d'augmenter le rayon de l'élément structurant donne un résultat beaucoup plus proche de l'image originale ; beaucoup plus de lignes sont détectées tandis que pour un petit rayon on ne distingue presque plus aucune ligne de l'image. En contrepartie, le temps d'exécution est nettement supérieur pour un rayon élevé.

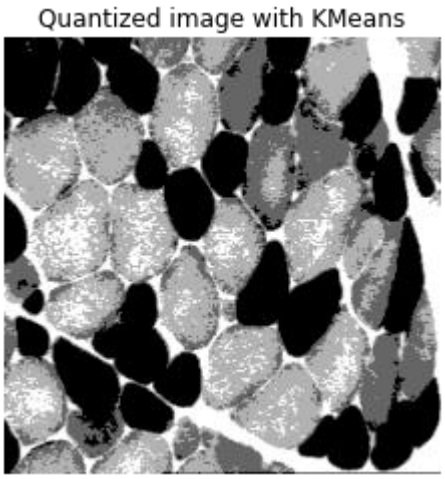
Avec un seuil haut trop élevé, on observe une disparition de beaucoup de contours ; tandis qu'en le baissant trop le filtre détecte des pixels qui ne sont pas des contours. De même, une augmentation du seuil bas trop élevé fait disparaitre des contours, mais s'il est trop bas on aura des problèmes au niveau des zones avec des variations importantes de niveaux de gris ne correspondant pas à des contours.

Des seuils à 1 et 5 semblent offrir un résultat satisfaisant : les contours sont à peu près fermés, et il reste relativement peu de bruit.

3.1 Cette classification ne segmente pas correctement les différents types de cellules ; on perd l'information des cellules noires.

Avec une segmentation en 3 classes, on récupère cette information et on obtient un résultat très satisfaisant.

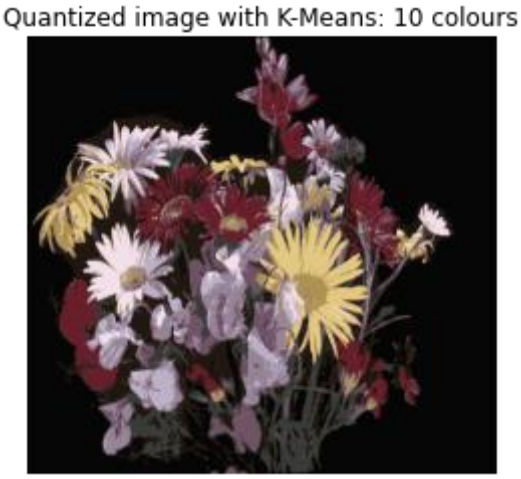
Avec une initialisation aléatoire, la classification obtenue est quand même stable.



Pour muscle.tif, une segmentation en 2 classes n'est pas suffisante puisqu'on ne voit plus les cellules grises (image de gauche) ; à l'inverse, un nombre de classes trop important ne fonctionne pas, puisque la segmentation divise des cellules en plusieurs niveaux de gris (image de droite).

Le filtre médian permet d'enlever la texture des cellules ; ainsi elles sont segmentées avec un seul niveau de gris, et il n'y a pas besoin d'un nombre de classes important pour obtenir un résultat correct.

3.2 Les couleurs de l'image segmentée sont plus fades, et on perd des informations telles que la texture des pétales, où même la netteté des bords.



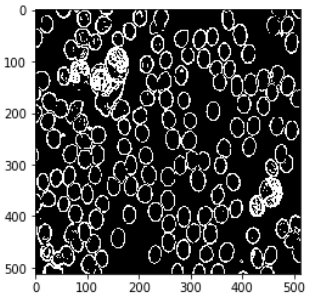
On commence à obtenir une image similaire à l'originale pour un nombre de classes égal à 20 environ : les pétales sont de plusieurs couleurs, ce qui permet de rendre même la texture.

Pour carte.tif, la solution est d'augmenter la valeur de n jusqu'à retrouver l'image originale.

4. Le script otsu.py a pour but de trouver le meilleur seuil pour séparer l'image en deux canaux : tous les pixels ayant un niveau de gris inférieur à ce seuil deviennent noir, et les autres blancs.

Les résultats sont plutôt satisfaisants, mais pour une image avec de grandes étendues de gris comme celle des dés, le fond n'est pas clairement différenciable des dés puisqu'il apparait par endroits en gris, par endroits en noir.

La norme du gradient de cell.tif est très bien seuillée, et la détection de contours est alors beaucoup plus convaincante qu'avec Sobel (seuil fixe).



5. Pour chaque pixel, il faut calculer la moyenne et l'écart type local dans un cercle de rayon r. Si cette moyenne est inférieure à l'écart type, alors le pixel est ajouté à l'objet existant.

Plus thresh est grand, plus l'algorithme est tolérant et plus la sera grande (beaucoup de pixels ajoutés.

La position du pixel initial est bien sur de la matière blanche. Pour thresh plus élevé, on obtient de meilleurs résultats, et pour un rayon supérieur à 6, des pâtés apparaissent ; avec un rayon plus faible l'algorithme devient plus lent mais les résultats sont plus convaincants.

Avec (x0,y0) = (250,100), thresh = 2 et rayon = 4, on arrive un petit peu à segmenter la matière grise. Le résultat n'est cependant pas très convaincant, il existe surement de meilleures façons d'adapter ce paramètres. En revanche, cette méthode ne segmente pas les zones grises à l'intérieur des zones blanches, ce qui présente un problème important…