TP IMA 3 : Segmentation

ANTOINE ANDURAO

1. Détection de contours

   1.1 Filtre de gradient local par masque

   - Le filtre de Sobel est un filtre 3x3, il est donc un peu moins précis qu'un simple filtre à différence (c'est-à-dire que les contours qu'ils détectent sont moins bien localisés et souvent épais), mais les images ainsi obtenues sont généralement plus fiables et permettent des post-traitements plus poussés.
   - Il ne faut pas appliquer de filtre passe-bas avant le filtre de Sobel car celui-ci détecte les hautes fréquences (contours).
   - Plus le seuil est grand, plus les contours sont propres, mais aussi moins il y a d'information. Par exemple avec seuil = 0.5 les contours sont bien définis, mais on ne retrouve que les zones ayant un très grand gradient. Avec un seuil trop petit, le bruit passe et l'image reste bruitée, on distingue mal les contours. La valeur de seuil idéale semble être autour de 0., avec un bon compris contours définis/bruit

   1.2 Maximum du gradient filtré dans la direction du gradient

   - Le maximum du gradient dans sa direction permet de récupérer au mieux les contours puisqu'ils correspondent à un maximum de "discontinuité" au niveau des valeurs de l'image.
   - De même que pour la méthode précédente, il s'agit de trouver un compromis entre perte d'information (donc contours mal définis) et suppression du bruit. La valeur 0.1 semble à nouveau bien convenir.

2. Seuillage avec hystérésis

   - Quand on augmente le rayon, on augmente l'intensité des lignes plus fines - on perçoit plus de lignes, et avec une intensité plus forte
   - L : 1/ H : 10 : Certains contours ne sont pas fermés
   - L : 5/ H : 10 : Des valeurs trop proches donnent des observations moins précises/pertinentes, il faut donc garder un bon écart entre les 2 seuils
   - L : 5/ H : 15 : on a augmenté les 2 seuils, il reste des contours non fermés
   - L : 1/ H : 5 : On obtient des résultats très satisfaisants, les contours sont fermés sans qu'il reste trop de bruit

- Grosse différence entre rayon=5 (Fig. 1) (avec L : 20 / H : 25 → pas de bruit, mais manque de contours par exemple dans le coin gauche inférieur) et rayon=6 (Fig. 2) (L : 40 / H : 45 → on a tous les contours, mais beaucoup de bruit)

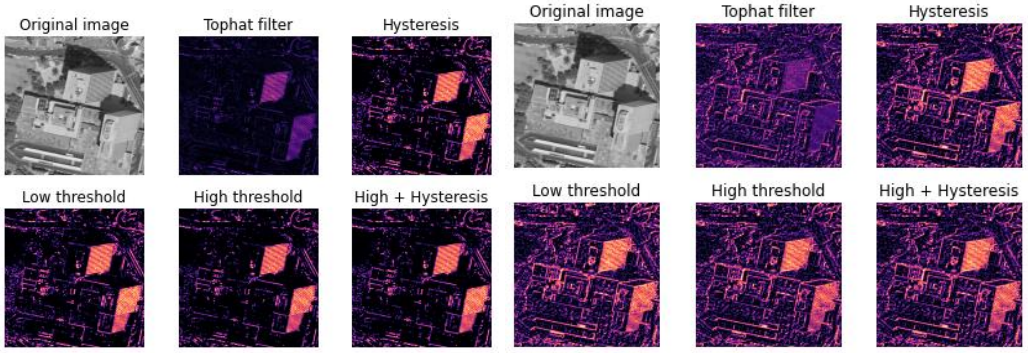


Figure 1

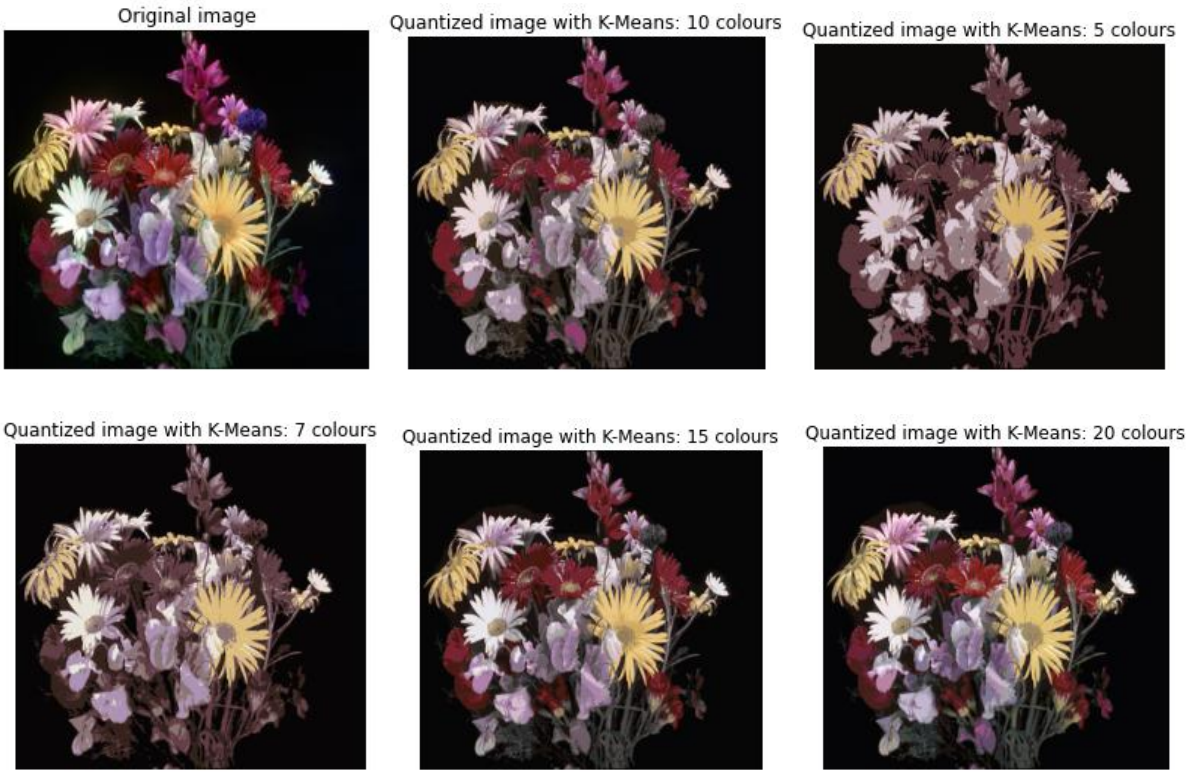
Figure 2

3. Segmentation par classification : k-moyennes
    1. Image à niveaux de gris

        - Pour 2 classes : on perd l'information sur les noyaux noirs, mais on a une bonne restitution des cellules et des noyaux clairs
        - Avec 3 classes, on récupère l'info manquante dans les noyaux noirs
        - Random-state = None → initialisation aléatoire → Classes stables dans tous les cas
        - « muscle.tif » : 2 classes → trop peu car on garde que les cellules noires. Si le nombre de classe est trop faible, le dégradé de couleurs au sein d'une cellule amène à son découpage en plusieurs classes
        - Avec le filtre médian, perte de la texture des cellules → cellules homogènes (1 seule couleur). Il n'y pas besoin de beaucoup de classes : on a réglé le problème précédent

    2. Image en couleur

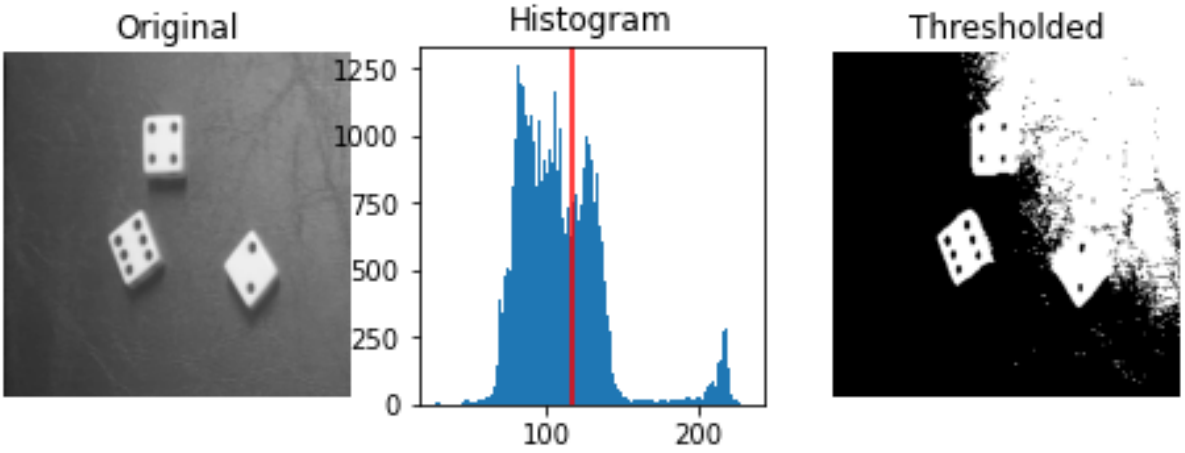
        - On obtient des couleurs plus fades et on perd la texture et le dégradé des pétales et quelques formes
        - Avec 5 classes : on garde que le jaune, blanc, marron/rouge
        - Avec 7 classes : on retrouve le rouge, mais il manque encore des formes (pétales violet foncé en bas à droite)
        - Avec 15 classes : ça va mais il manque toujours des formes
        - Enfin avec 20 classes : on a le détail des différentes couleurs d'une même pétale



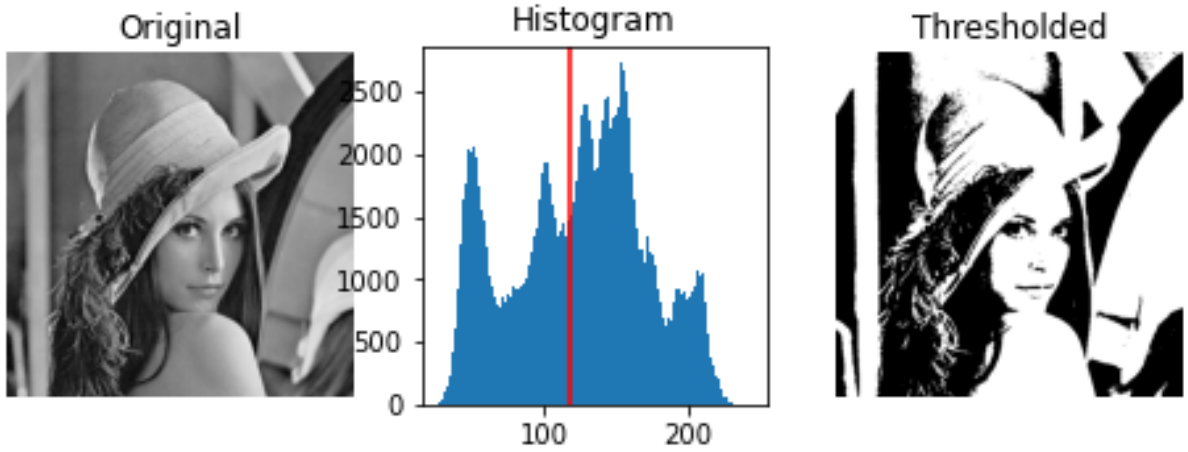
- La solution est d'augmenter progressivement le n_colors (soit le nombre de classes de couleurs) jusqu'à retrouver l'image originale de « carte.tif »

4. Seuillage automatique : Otsu

- On cherche à optimiser avec le critère suivant : La + grande différence d'intensité moyenne entre les classes, telle que les pixels soient le mieux répartis entre les deux classes. Cela revient à minimiser la dispersion intra-classe comme indiqué dans l'énoncé
- Sur « cell.tif: » : pas de valeur ajoutée par rapport à la méthode K-Means
- La méthode Otsu ne marche pas pour « dice.tif » car il faudrait au moins 3 classes



- Avec « pyra.tif » : utilisable si on découpe l'histogramme en plusieurs histogrammes car la méthode Otsu sépare très bien
- Pour « lena.tif » : On retrouve très bien les contours



5. Croissance de régions
    - Il faut que la moyenne d'intensité de son voisinage de taille rayon diffère de la moyenne d'intensité de la zone de départ de moins de threshold*S0
    - Thresh est le critère de sélectivité : plus on l'augmente, moins la sélection est rude (so>1 car valeurs entières d'intensité des pixels)
    - (Xo,Yo) est bien initialisé sur un point correspondant à de la matière blanche. En passant threshold de 2 (Fig. 3) à 5 (Fig. 4), on obtient un bien meilleur résultat.
      Avec rayon=6 → on observer l'apparition de pâtés. Avec rayon=3 (Fig. 5) : l'algo est plus lent, mais la forme de la zone sélectionnée plus précise : on obtient un très bon résultat.

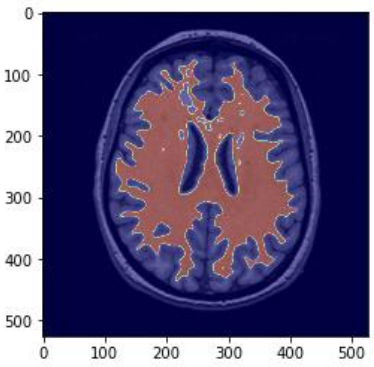


Figure 3

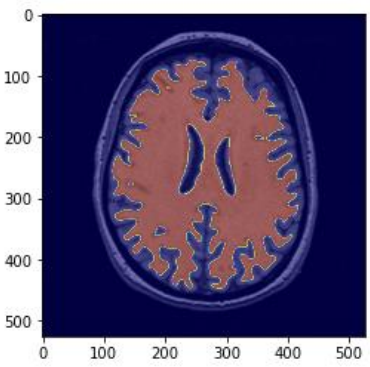


Figure 4

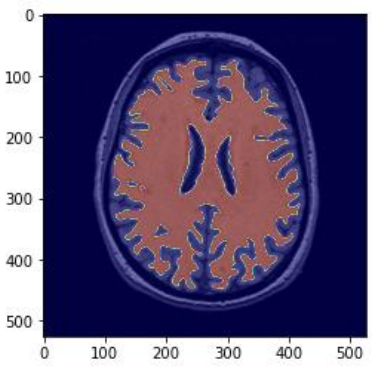


Figure 5

- Si on augmente le threshold, on finit par accepter la matière grise, mais sans la distinguer de la blanche
- Le prédicat ne dépend pas de l'évolution de la région, mais uniquement du voisinage autour du pixel étudié et du voisinage du pixel de départ. Il compare

la moyenne d’intensité du voisinage du pixel étudié à la moyenne d’intensité du voisinage du « centre » (pixel de départ), qui reste inchangée. (On pourrait la calculer pour tous les pixels d’un coup)

- On enlève l’usage de perimeter et on calcule directement la différence la moyenne d’intensité du voisinage de chaque pixel de l’image à la moyenne d’intensité du voisinage du « centre » (pixel de départ), et s’il confirme le prédicat, est ajouté à la zone de sélection
- Il faudrait remplacer la moyenne d’intensité du voisinage du « centre » (pixel de départ) dans le prédicat par la moyenne d’intensité sur toute la région sélectionée à chaque étape.