TP de IMA201 : segmentation des images

Le TP utilise exclusivement des scripts **python**, que vous pouvez lancer depuis la plate-forme **spyder**, par exemple.

L'affichage des images pouvant poser des problèmes d'échantillonnage ou de quantification, n'hésitez pas à zoomer dans les images, ou à utiliser la fonction **viewimage** des premiers TP.

1 Détection de contours

1.1 Filtre de gradient local par masque

Le script **sobel.py** effectue la détection de contours en utilisant les filtres de dérivation de Sobel.

- Rappelez l'intérêt du filtre de Sobel, par rapport au filtre différence, qui calcule une dérivée par la simple différence entre deux pixels voisins,
- Est-il nécessaire de faire un filtre passe-bas de l'image avant d'utiliser le filtre de Sobel ?
- Le seuillage de la norme du gradient permet d'obtenir des contours. Commentez la qualité des contours obtenus (robustesse au bruit, continuité, épaisseur, position...) quand l'on fait varier ce seuil.

1.2 Maximum du gradient filtré dans la direction du gradient

La fonction **maximaDirectionGradient** permet de déterminer les pixels de l'image qui sont des maxima du gradient dans la direction du gradient.

- Quel critère de qualité est optimisé par ce procédé ?
- Il est possible d'éliminer les contours dont la norme est inférieur à un seuil donné. Commentez les résultats obtenus en terme de position et de continuité des contours, et de robustesse au bruit en faisant varier ce seuil.
- Cherchez à fixer le seuil sur la norme de façon à obtenir un compromis entre robustesse au bruit et continuité des contours.

1.3 Filtre récursif de Deriche

Le filtre de Deriche est un filtre récursif du deuxième ordre qui optimise les critères de Canny pour la détection de contours.La réponse impulsionnelle du filtre de Deriche est : $f(x) = k\ x\ e^{-\alpha|x|}$.

Pour un signal comportant N échantillons, la mise en œuvre récursive de ce filtre suit le schéma :

$y_1(n) = x(n-1) + 2e^{-\alpha} y_1(n-1) - e^{-2\alpha} y_1(n-2)$ pour n = 3,...,N

$y_2(n) = x(n+1) + 2e^{-\alpha} y_2(n+1) - e^{-2\alpha} y_2(n+2)$ pour n = N-2,...,1

$y(n) = ke^{-\alpha}(y_1(n) - y_2(n))$ pour n=1,...,N

avec $k = -\frac{-(1-e^{-\alpha})^2}{e^{-\alpha}}$

où $y(n)$ est la réponse du filtre à l'entrée $x(n)$.

Dans le fichier, **mrlab.py**, des erreurs ont été commises dans les fonctions **dericheGradX** et **dericheGradY**. A vous de corriger ces fonctions (uniquement au niveau des lignes indiquées) afin de mettre en œuvre la récursivité.

- Testez la détection de contours avec ce filtre sur plusieurs images. Décrivez l'effet du paramètre α sur les résultats de la segmentation (faites varier ce paramètre sur l'intervalle $0,3...3,0$).
- Le temps de calcul dépend-il de la valeur de α ? Expliquez pourquoi.
- Comment et dans quel but les fonctions **dericheSmoothX** et **dericheSmoothY** sont-elles utilisiées (cf. le filtre de Sobel).

1.4 Passage par zéro du laplacien

Testez la détection de contours par passage par zéro du laplacien avec la routine **laplacien.py**.

- Quel est l'effet du paramètre α sur les résultats ?
- Sur l'image **cell.tif**, quelles sont les principales différences par rapport aux résultats fournis par les opérateurs vus précédemment (contours, Deriche) ?
- Sur l'image **pyramide.tif**, comment est-il possible de supprimer les faux contours créés par cette approche ?

1.5 Changez d'image

- Quel opérateur choisiriez-vous pour segmenter l'image **pyra-gauss.tif** ?
- Quels seraient les pré-traitements et les post-traitements à effectuer ?

2 Seuillage avec hystérésis

Dans cette partie, vous allez tester l'intérêt du seuillage avec hystérésis sur un exemple de détection de lignes dans une image.

Le principe du seuillage avec hystérésis est de sélectionner deux seuils différents de binarisation : l'un est suffisamment bas pour que les lignes soient continues, mais risque de sélectionner du bruit dans l'image, l'autre est suffisamment haut pour que les points préservés appartiennent bien à des lignes valides, mais qui ne laisse que des lignes très incomplètes. La seconde image est utilisée pour sélectionner dans la première image les lignes les plus significatives : les objets connexes après le seuillage bas qui comportent au moins un pixel au dessus du seuil haut.

2.1 Application à la détection de lignes

- Appliquez le filtre du Chapeau haut de forme (**tophat**) à une image SPOT pour effectuer une détection de lignes :
- Modifiez le rayon de l'élément structurant utilisé pour calculer le filtre tophat, et indiquez comment évoluent les lignes détectées.
- Modifiez les valeurs des deux seuils, et examinez comment les lignes sont supprimées ou préservées. Quels sont les seuils qui donnent, à votre avis, le meilleur résultat ?
- Appliquez le seuillage par hystérésis pour améliorer la détection de contours obtenue avec un des opérateurs vus précédemment sur une image de votre choix. Précisez la mise en oeuvre que vous proposez et commentez les résultats.

3 Segmentation par classification : K-moyennes

Vous pouvez lancer un algorithme des k-moyennes sur des images à niveaux de gris (kmeans1) ou sur des image en couleur (kmeans3), en choisissant une initialisation aléatoire des germes ou en spécifiant les centres initiaux des classes.

3.1 Image à niveaux de gris

- Testez l'algorithme des k-moyennes sur l'image **cell.tif** pour une classification en 2 classes. Cette classification segmente-t-elle correctement les différents types de cellules ? Si non, que proposez-vous ?
- Testez les différentes possibilités pour initialiser les classes. Décrivez si possible ces différentes méthodes.
- La classification obtnue est-elle stable (même position finale des centres des classes) avec une initialisation aléatoire ? Testez sur différentes images à niveaux de gris et différents nombres de classes.
- Quelles sont les difficultés rencontrées pour la segmentation des différentes fibres musculaires dans l'image **muscle.tif** ?

- Expliquez pourquoi le filtrage de l'image originale (filtre de la **moyenne** ou filtre **median**) permet d'améliorer la classification.

3.2 Image en couleur

- Testez l'algorithme sur l'image **fleur.tif** pour une classification en 10 classes, les centres des classes initiaux étant tirés aléatoirement).
- Commentez la dégradation de l'image quantifiée par rapport à l'image initiale.
- Quel est le nombre minimum de classes qui donne un rendu visuel similaire à celui de l'image codée sur 3 octets ?
- Proposez une solution pour retrouver les planches-mères utilisées pour l'impression d'une carte IGN : **carte.tif**.

4 Seuillage automatique : Otsu

La méthode de **Otsu** pour seuiller automatiquement une image, consiste tout d'abord à définir un critère à optimiser, tel que minimiser la dispersion intra-classe, puis à parcourir de façon exhaustive tous les seuils possibles (256 possibilités poour un problème à deux classes, beaucoup plus pour un prolème à trois classes).

- Dans le script **otsu.py** quel critère cherche-t-on à optimiser ?
- Testez la méthode de Otsu sur différentes images à niveaux de gris, et commentez les résultats.
- Cette méthode permet-elle de seuiller correctement une image de norme du gradient ?
- Modifiez ls script **otsu.py** pour traiter le problème à trois classes, i.e. la recherche de deux seuils.

5 Croissance de régions

A l'aide du script **region_growing.py**, testez le principe de croissance de régions pour la segmentation de la matière blanche dans une image de cerveau.

- Quelles contraintes doit vérifier un pixel pour être ajouté à l'objet existant ?
- Les paramètres à fixer sont la position du point de départ $(x0, y0)$, un seuil $thresh$ et le $rayon$ qui définit le voisinage sur lequel sont estimés la moyenne et l'écart-type locaux.
- Quel est l'effet du paramètre $thresh$ sur le résultat de segmentation ?

- Quels paramètres permettent de segmnenter correctement la matière blanche ?
- Parvenez-vous à segmenter la matière grise également ?
- Quel est le prédicat mis en place dans ce script ?
- Proposez un autre algorithme qui n'utilise pas la croissance de régions, mais qui donne le même résultat.
- Proposez un prédicat qui nécessite réellement un algorithme de croissance de région.