TP de IMA201 : segmentation des images

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

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 Maxima du module du gradient dans la direction du gradient

La fonction **maximaDirectionGradient** (qui se trouve dans **mrlab.py**) permet de déterminer les pixels de l'image qui sont des maxima du gradient dans la direction du gradient.

— Rappeler la formule qui donne les points sélectionnés par cette fonction (cours) ?

— Il est possible d'éliminer les contours dont la norme du gradient est inférieure 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 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 ?

— Sur l'image **pyramide.tif**, comment est-il possible de supprimer les faux contours créés par cette approche ?

1.4 Adaptation en fonction de l'image

— Quel opérateur choisiriez-vous pour segmenter l'image **pyra-gauss.tif** ?

— Quels seraient les pré-traitements et les post-traitements à effectuer ?

2 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.

2.1 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 obtenue 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.

2.2 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 ?

3 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.

4 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.tif**.

— 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 segmenter correctement la matière blanche ?
— Parvenez-vous à segmenter la matière grise également ?