Introduction au Traitement des Images (IMA201)

TP3

Éleve: Vitor de Sousa França

Email: vitor.franca@telecom-paris.fr

Septembre
2022

1 Détection de contours

1.1 Filtre de gradient local par masque

Pour cette session l'image cell.tif a été utilisée.

1.1.1 Rappelez l'intéret du filtre de Sobel, par rapport au filtre différence, qui calcule une dérivée par la simple différence entre deux pixels voisin.

Tandis que la dérivée simple fait la différence entre les pixels voisins le filtre Sobel calcule le gradient dans une direction tout en lissant avec une somme pondérée dans l'autre direction.

1.1.2 Est-il n ecessaire de faire un filtre passe-bas de l'image avant d'utiliser le filtre de Sobel ?

Normalement il n'est pas nécessaire d'utiliser un filtre passe-bas avant le filtre de Sobel parce que il fait déjà une somme pondérée dans une direction et cela rend le gradient moins sensible au bruit.

1.1.3 Le seuillage de la norme du gradient permet d'obtenir des contours. Commentez la qualité des contours obtenus (robustesse au bruit, continuit e, epaisseur, position...) quand l'on fait varier ce seuil.

Le filtre sobel est robuste au bruit car il applique un filtre passe haut tout en appliquant un filtre passe bas dans une autre coordonnée, le seuil 0.1 était suffisant pour sélectionner les contours, résister au bruit et maintenir leur continuité. Augmenter le seuil rend la norme du gradient plus robuste au bruit, mais trop l'augmenter affecte la continuité des contours

1.2 Maximum du gradient filtré dans la direction du gradient

1.2.1 Quel critère de qualité est optimisé par ce procédé ?

Le critère est bon pour déterminer la position, mais par lui-même il n'est pas robuste au bruit et aux faux contours.

1.2.2 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.

Le Seuil fonctionne de la même façon qu'avant, quand on l'augmente, il devient plus robuste au bruit, mais quand on l'augmente trop on perd de l'information. Avec l'ajout du gradient max dans le sens du gradient, la meilleure précision des contours peut être remarquée.

1.2.3 Cherchez à fixer le seuil sur la norme de façon à obtenir un compromis entre robustesse au bruit et continuité des contours.

Le seuil égale à 0.15 garantit le mieux compromis entre réduction de bruit en gardant la continuité.

1.3 Filtre récursif de Deriche

1.3.1 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).

Pour les petites valeurs alpha (autour de 0 à 1), le filtre devient résistant au bruit et, par conséquent, avec une bonne détection des contours. Cependant, avec des valeurs alpha élevées (plus proches de 3), le critère de bonne localisation du contour est améliorée.

1.3.2 Le temps de calcul dépend-il de la valeur de α? Expliquez pourquoi.

Le temps de calcul ne dépend pas de la valeur de α, parce que cette paràmetre n'est qu'une constante et n'a aucune lien avec la récursivité de l'algorithme.

1.3.3 Comment et dans quel but les fonctions dericheSmoothX et dericheSmoothY sont-elles utilisiées.

Le dericheSmooth atténue le bruit et permet donc d'utiliser des valeurs plus élevées de α, améliorant ainsi le critère de bonne localisation.

1.4 Passage par zéro du laplacien

1.4.1 Quel est l'effet du paramètre α sur les résultats?

Plus le paramètre α est élevé, meilleure est la bonne localisation des contours, par contre, l'algorithme devient plus sensible aux faux contours.

1.4.2 Sur l'image cell.tif, quelles sont les principales différences par rapport aux résultats fournis par les opérateurs vus précédemment (contours, Deriche) ?

Par rapport aux résultats fournis par les opérateurs précedent, le passage par zéro du laplacien est un bon algorithme en termes de continuité du contour. Toutefois, pour petites valuers α il ne pas très robuste à bonne localisation et pour grand valeurs α il s'aggrave aux faux contours.

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

On peut faire une condition oú il y a une limite inférieure pour la norme de gradient telle que les gradients de les régions uniforms soient plus petits que cela.

1.5 Changez d'image

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

Je choisirais l'operateur de Direch avec le dericheSmooth parce que cette operateur respect le bonne localisation du contour et avec le dericheSmooth il devient resistent au bruit.

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

Comme une pré-traitements on peut utiliser une filtre linèaire pour reduire le bruit (même que ont déjà utilise le dericheSmoth) et comme une post-traitement on peut effectuer l'hystérésis pour assurer la continuité des contours

2 Seuillage avec hystérésis

Appliquez le filtre du Chapeau haut de forme (tophat) à une image SPOT pour effectuer une détection de lignes

2.1 Modifiez le rayon de l'élément structurant utilisé pour calculer le filtre tophat, et indiquez comment évoluent les lignes détectées.

on peut vérifier que plus le rayon est grand, plus il y a de lignes détectées.

2.2 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 ?

Quand on augment le seuil bas on voir que l'image résultante devient de plus en plus proche de la réponse avec le seuil plus haut. Donc, quand on fait cela plus de lignes sont supprimées. La diminution du seuil haut a un impact plus important sur la préservation des lignes. À mon avis, le seuil qui donnent le meilleur résultat sont 1.5 pour le seuil bas et 3 pour le seuil haut, donc la plupart des lignes restent connectées.

2.3 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.

En utilisent l'image cell.tif, la fonction "tophat" avec la valeur du rayon égal a 10 et aussi en utilisent "apply_hysteresis_threshold" avec le seuil bas égale à 10 et le seuil haut égale à 15 c'est possible d'améliorer la détection de contours obtenue par rapport à un des opérateurs vus précédemment. Mais il y a encore des faux contours.

3 Segmentation par classification: K-moyennes

3.1 Image à niveaux de gris

3.1.1 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 ?

La classification en 2 classes n'est pas capable de segmenter correctmenet parce que il est nécessaire de au moins 3 classes pour faire la différentiation entre les celules.

3.1.2 Testez les différentes possibilités pour initialiser les classes. Décrivez si possible ces différentes méthodes.

Il y a la méthode des points aléatoires où k points aléatoires sont choisis comme premiers centroïdes et aussi la méthode k-means++ qui utilise d'une approche probabiliste proportionnelle à la distance au carré entre le point donné et le centroïde initialement choisi au hasard.

3.1.3 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.

La classification obtenu avec une initialisation aléatoire n'est pas stable.

3.1.4 Quelles sont les difficultés rencontrées pour la segmentation des différentes fibres musculaires dans l'image muscle.tif ?

Le bruit présent dans certains muscles rend difficile une bonne segmentation de l'image, ce qui les classe comme des espaces vides.

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

Comme le problème est un bruit interne aux muscles, l'utilisation d'un filtre passe-bas permet d'homogénéiser la zone et ainsi d'obtenir une meilleure classification.

3.2 Image en couleur

3.2.1 Commentez la dégradation de l'image quantifiée par rapport à l'image initiale.

En utilisant l'algorithme avec 10 classes on peut observer la perte de couleur dans l'image résultante, augmenter les classes permet de réduire cette perte.

3.2.2 Quel est le nombre minimum de classes qui donne un rendu visuel similaire à celui de l'image codée sur 3 octets ?

Pour représenter l'image avec fidélité, il faudrait avoir 768 couleurs et donc 768 classes. Toutefois, pour un rendu visuel similaire, le nombre minimum de 80 classes peut être considéré comme un bon choix mais il y a des pertes de contraste qui peuvent être remarquées.

3.2.3 Proposez une solution pour retrouver les planches-mères utilisées pour l'impression d'une carte IGN : carte.tif.

On peut utiliser le Tophat filter, afin de localiser les lignes est aprés

4 Seuillage automatique : Otsu

4.1 Dans le script otsu.py quel critère cherche-t-on à optimiser?

On utilise la minimisation de la variance intraclasse comme méthode d'optimisation.

4.2 Testez la méthode de Otsu sur différentes images à niveaux de gris, et commentez les résultats.

En testant la méthode de Otsu sur différentes images à niveux de gris on peut voir que il est efficient pour faire la seuillage des images qui ont l'histogramme distribué de manière binomial.

4.3 Cette méthode permet-elle de seuiller correctement une image de norme du gradient ?

Oui, il est possible d'obtenir les contours dans les images en utilisant la méthode d'Otsu.

4.4 Modifiez le script otsu.py pour traiter le problème à trois classes, i.e. la recherche de deux seuils.

```
from skimage.filters import threshold_multiotsu

ima = skio.imread('pyramide.tif')

image = np.uint8(norme)

thresh = threshold_multiotsu(image)
print(thresh)

regions = np.digitize(image, bins=thresh)

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(10, 3.5))

ax[0].imshow(image, cmap='gray')
ax[0].set_title('Original')
ax[0].axis('off')

ax[1].hist(image.ravel(), bins=255)
ax[1].set_title('Histogram')
for t in thresh:
    ax[1].axvline(t, color='r')

# Plotting the Multi Otsu result.
ax[2].imshow(regions, cmap='gray')
ax[2].set_title('Multi-Otsu result')
```

```
ax[2].axis('off')

plt.subplots_adjust()
plt.show()
```

5 Croissance de régions

5.1 Quelles contraintes doit vérifier un pixel pour être ajouté à l'objet existant ?

Pour un pixel être ajouté a l'objet existant il faut que la région du pixel soit croissance. C'est-à-dire que la différence absolue entre la moyenne de l'image d'objet existant et la moyenne de la région est inférieure au seil fois l'écart type de l'objet.

5.2 Quel est l'effet du paramètre thresh sur le résultat de segmentation ?

Le paramètre thresh est le seuil du prédicat. Plus le paramètre est élevé, plus il y a de régions intégrées dans l'image résultante.

5.3 Quels paramètres permettent de segmnenter correctement la matière blanche ?

Pour bien segmenter la matière blanche il faut augmenter le paramètre thresh à la valeur 9 et et le rayon à 3.

5.4 Parvenez-vous à segmenter la matière grise également ?

Pour segmenter la matière grise aussi il faut augmenter le rayon à 5 et le thresh à 15. Par contre, si on veut segmenter seulement le matière grise, un pixel dans cette région doit être placé comme point de départ.

5.5 Quel est le prédicat mis en place dans ce script ?

Le prédicat c'est le differences limitées.

5.6 Proposez un autre algorithme qui n'utilise pas la croissance de régions, mais qui donne le même résultat.

On peut utiliser l'algorithme K-moyennes avec 3 classes: une pour la matière blanche, une autre pour la matière grise et la dernière pour les régions sombres.

5.7 Proposez un prédicat qui nécessite réellement un algorithme de croissance de région.

L'algorithme de croissance de région serai nécessaire si on utilise la connexité comme un prédicat.