# TP3 - Questions (William Liaw)

## 1. Détection de contours

### 1.1. Filtre de gradient local par masque

* Rappelez l’intérêt du filtre de Sobel, par rapport au filtre différence, qui calcule une dérivée par la simple différence entre deux pixels voisins.
  + Le filtre Sobel met l’accent sur les pixels les plus proches du centre du masque, car pour obtenir le opérateur de Sobel, on effectue un produit externe entre un filtre gaussien 1D et la dérivée (difference). Le filtre gaussien est utilisé pour réduire le bruit qui donne des images floues. Ainsi, l’opérateur Sobel calcule le gradient de l’image avec moins de bruit. L’opérateur Sobel est basé sur la convolution de l’image avec un petit filtre séparable et à valeurs entières dans les directions horizontale et verticale et est donc relativement peu coûteux en termes de calculs. Il en va toutefois de même pour le filtre des différences, qui nécessite encore moins de puissance de calcul. Le filtre de différence simple (2x2), qui calcule la différence entre deux pixels voisins, est centré entre les pixels, ce qui provoque un décalage de l’image. En introduisant une colonne (ou une ligne) de zéros entre les coefficients asymétriques, le filtre Sobel (3x3) est centré sur le pixel lui-même.
  + Filtre de Sobel
  + [  
     [1, 0, -1]  
     [2, 0, -2]  
     [1, 0, -1]  
    ]
  + Filtre de differences
  + [  
     [1, 0]  
     [-1, 0]  
    ]
* Est-il nécessaire de faire un filtre passe-bas de l’image avant d’utiliser le filtre de Sobel?
  + L’intérêt d’utiliser un filtre passe-bas est de débruiter l’image, améliorant ainsi la qualité des bords, ce qui, par extension, améliore la qualité de détection des contours. Par conséquent, la nécessité d’utiliser un filtre passe-bas avant d’appliquer le filtre Sobel dépend du niveau de bruit de l’image, pour des niveaux très faibles, la robustesse du filtre Sobel peut être suffisante.
* 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.
  + Lorsqu’on fixe un seuil très bas, les contours détectés sont continus, cependant ils sont très sensibles au bruit et assez épais, ce qui implique une incertitude sur leur véritable localisation, aussi on observe la détection de faux contours. En augmentant le seuil, les contours deviennent plus fins (meilleure estimation de position) et plus robustes au bruit, au détriment de la continuité. De plus, bien que le nombre de faux contours détectés soit réduit, une partie des vrais contours est également perdue.

### 1.2. Maximum du gradient filtré dans la direction du gradient

* Quel critère de qualité est optimisé par ce procédé?
  + Les critères de Canny définissent analytiquement les caractéristiques souhaitables pour la détection de contours:
    - une bonne détection, autant de bords que possible doivent être détectés avec précision, ce qui signifie que le détecteur doit avoir une réponse forte même aux contours faibles et doit maximiser le rapport signal sur bruit;
    - bonne localisation, les points détectés doivent être au centre du contour;
    - réponse unique, chaque contour ne doit être détecté qu’une seule fois et le bruit ne doit pas créer de faux contours.
  + La procédure de détection du maximum de gradient dans la direction du gradient n’est pas robuste au bruit et détecte de nombreux faux bords, elle n’optimise donc pas les premier et troisième critères. Cependant, en détectant les maximums locaux du gradient, cette procédure trouve l’emplacement précis des contours et donne des lignes fines, optimisant ainsi le deuxième critère.
* 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.
  + Lorsqu’on applique la procédure de détection du maximum du gradient sur la direction du gradient, on doit filtrer les contours détectés par seuillage selon la norme du gradient, afin d’éliminer les faux contours.
  + Plus le seuil est élevé, moins on détecte de faux contours (plus robuste au bruit), cependant on perd la continuité et une partie des vrais contours. La localisation et l’épaisseur des contours ne sont pas affectées.
* Cherchez à fixer le seuil sur la norme de fa¸con à obtenir un compromis entre robustesse au bruit et continuité des contours
  + En faisant varier le seuil utilisé pour écarter les contours en fonction de la norme du gradient, un bon compromis entre robustesse au bruit et continuité des contours semble être trouvé pour un seuil de 0,15: la plupart des contours parasites sont éliminés, même si certaines microstructures subsistent et la majorité des cellules sont détectées.

### 1.3. Filtre récursif de Deriche

* 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.
* def dericheGradX(ima, alpha):  
   nl, nc = ima.shape  
   ae = math.exp(-alpha)  
   c = -(1 - ae) \* (1 - ae) / ae  
    
   b1 = np.zeros(nc)  
   b2 = np.zeros(nc)  
    
   gradx = np.zeros((nl, nc))  
    
   for i in range(nl):  
   l = ima[i,:].copy()  
    
   for j in range(2, nc):  
   b1[j] = l[j - 1] + 2 \* ae \* b1[j - 1] - ae \* ae \* b1[j - 2]  
   b1[0] = b1[2]  
   b1[1] = b1[2]  
    
   for j in range(nc - 3, -1, -1):  
   b2[j] = l[j + 1] + 2 \* ae \* b2[j + 1] - ae \* ae \* b2[j + 2]  
   b2[nc - 2] = b2[nc - 3]  
   b2[nc - 1] = b2[nc - 3]  
    
   gradx[i,:] = c \* ae \* (b1 - b2)  
    
   return gradx  
    
  def dericheGradY(ima, alpha):  
   nl, nc = ima.shape  
   ae = math.exp(-alpha)  
   c = -(1 - ae) \* (1 - ae) / ae  
    
   b1 = np.zeros(nl)  
   b2 = np.zeros(nl)  
    
   grady = np.zeros((nl, nc))  
    
   for i in range(nc):  
   l = ima[:, i].copy()  
    
   for j in range(2, nl):  
   b1[j] = l[j - 1] + 2 \* ae \* b1[j - 1] - ae \* ae \* b1[j - 2]  
   b1[0] = b1[2]  
   b1[1] = b1[2]  
    
   for j in range(nl - 3, -1, -1):  
   b2[j] = l[j + 1] + 2 \* ae \* b2[j + 1] - ae \* ae \* b2[j + 2]  
   b2[nl - 1] = b2[nl - 3]  
   b2[nl - 2] = b2[nl - 3]  
    
   grady[:, i] = c \* ae \* (b1 - b2)  
    
   return grady
* Testez la détection de contours avec ce filtre sur plusieurs images. Décrivez l’effet du paramètre alpha sur les résultats de la segmentation (faites varier ce paramètre sur l’intervalle 0, 3…3, 0).
  + Le paramètre d’échelle alpha du filtre de Deriche correspond à l’inverse du paramètre sigma du filtre de Canny, qui indique en dessous de quelle distance deux contours parallèles sont fusionnés. Par conséquent, le paramètre alpha peut être ajusté pour filtrer le bruit en reliant les arêtes adjacentes en contours longs, lisses et continus.
  + En réduisant le paramètre alpha, nous observons une réduction du bruit et de la détection parasite du contour. Cependant, en le rendant trop petit, les bords des différents objets commencent à fusionner en un seul contour.
* Le temps de calcul dépend-il de la valeur de alpha? Expliquez pourquoi.
  + Le temps d’exécution du détecteur de contours de Deriche ne dépend pas de la valeur de alpha, puisqu’il est uniquement utilisé pour calculer les coefficients de l’expression récursive, une fois les coefficients calculés, ses valeurs peuvent être stockées. La valeur du paramètre ne modifie pas le nombre d’opérations.
* Comment et dans quel but les fonctions **dericheSmoothX** et **dericheSmoothY** sont-elles utilisiées (cf. le filtre de Sobel)
  + Le filtre Deriche est initialement défini en une dimension, il est ensuite étendu à deux dimensions par l’application croisée de deux filtres, un dans la direction x (détection de la composante verticale des arêtes), et un dans la direction y (détection de la composante horizontale des bords). De plus, dans la direction du contour est définie une fonction de lissage qui permet le filtrage du bruit, cette fonction correspond aux procédures dericheSmoothX et dericheSmoothY.
  + En conclusion, un filtre détecteur de bord est composé de deux estimateurs de dérivée, l’un dans la direction x et l’autre dans la direction y. Chacun de ces détecteurs est composé du produit de deux fonctions, en prenant par exemple le détecteur dans la direction x, nous avons un filtre passe-bas selon Oy (fonction de lissage) et un filtre passe-haut selon Ox.

### 1.4. Passage par zéro du laplacien

* Quel est l’effet du paramètre alpha sur les résultats?
  + En réduisant le paramètre alpha, nous observons une réduction du bruit et de la détection parasite du contour. Cependant, en le rendant trop petit, les bords des différents objets commencent à fusionner en un seul contour.
* Sur l’image **cell.tif**, quelles sont les principales différences par rapport aux résultats fournis par les opérateurs vus précédemment (contours, Deriche)?
  + Les filtres à gradient local, tels que le filtre Sobel, génèrent uniquement des données de bord locales, au lieu de récupérer la structure globale d’une frontière et sont très sensibles au bruit. La procédure de maximisation du gradient dans la direction du gradient ignore également les structures globales, mais pas autant que les masques locaux, et est sensible au bruit, mais optimise au moins le critère de localisation. D’un autre côté, le filtre de Deriche et les passages par zéro de la procédure laplacienne sont capables d’abstraire les structures globales, grâce à l’ajustement du paramètre alpha, et le filtre de Deriche en particulier optimise tous les critères de Canny. L’approche laplacienne présente l’avantage de détecter des contours fermés.
  + En comparant les résultats obtenus sur l’image cell.tif pour la méthode laplacienne et le filtre de Deriche, nous observons que pour des valeurs plus élevées de alpha le premier est moins robuste au bruit. Ceci s’explique par le fait que la méthode laplacienne ne prend pas en compte la norme du gradient.
* Sur l’image **pyramide.tif**, comment est-il possible de supprimer les faux contours créés par cette approche?
  + L’approche laplacienne présente l’inconvénient que le laplacien des points d’inflexion d’une fonction est également égal à zéro, ce qui entraîne la détection de faux contours sur des images en escalier, comme pyramide.tif. Pour éliminer de tels contours, il est possible d’éliminer les contours dont la norme est inférieure à un seuil donné car comme ils ne correspondent pas aux bords réels, la norme du gradient sera très petite dans de telles régions.

### 1.5. Changez d’image

* Quel opérateur choisiriez-vous pour segmenter l’image **pyra-gauss.tif**?
  + Puisque l’image **pyra-gauss.tif** est bruitée et en escalier, je choisirais l’opérateur de Deriche tel qu’il est le plus robuste au bruit et optimise les critères de Canny, la détection des faux contours dus au bruit peut également être contrôlée grâce à l’ajustement du paramètre alpha.
* Quels seraient les pré-traitements et les post-traitements à effectuer?
  + L’image **pyra-gauss.tif** étant bruitée, le pré-traitement doit consister en une étape de débruitage (convolution avec un filtre passe-bas). Et le post-traitement comprendrait la détection du maximum du gradient dans la direction du gradient combinée à une suppression de contour basée sur un seuillage de norme de gradient.

## 2. Seuillage avec hystérésis

### 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.
  + En augmentant le rayon, les lignes détectées deviennent plus épaisses et plus continues, mais on constate également une augmentation de la détection des faux contours.
* 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?
  + À mon avis, pour un rayon de 5, les seuils haut et bas qui donnent les meilleurs résultats pour **spot.tif** sont respectivement 2 et 15. Ce sont ces valeurs qui donnent le meilleur compromis entre la détection des routes sur l’image et la réduction du nombre de contours parasites.
* 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.
  + Un seuil d’hystérésis sera appliqué pour améliorer la détection de contour sur l’image **cell.tif** en utilisant l’opérateur Sobel combiné avec le maximum du gradient dans la procédure de direction du gradient. Tout d’abord, comme étape de pré-traitement, un filtre gaussien passe-bas est utilisé pour débruiter l’image. L’opérateur Sobel est ensuite appliqué à l’image pour calculer le gradient (norme et direction) et les contours sont détectés comme le maximum du gradient dans la direction du gradient.
  + Ensuite, un seuillage par hystérésis est utilisé pour améliorer la qualité de la détection: un seuil est choisi suffisamment bas pour que les raies détectées lors du seuillage de la norme de gradient soient continues, ce seuil est très sensible au bruit; un deuxième seuil est choisi suffisamment haut pour ne conserver que les points appartenant à des contours valides. La deuxième image est ensuite utilisée pour sélectionner les contours de la première image qui contiennent au moins un point conservé par le seuil haut. L’image résultante est utilisée comme masque appliqué aux contours détectés par le maximum du gradient dans la procédure de direction du gradient.
  + Le maximum du gradient dans la procédure dans le sens du gradient garantit une bonne localisation, tandis que le seuillage par hystérésis permet la suppression des contours parasites sans sacrifier la continuité.

## 3. Segmentation par classification: K-moyennes

### 3.1. Image à niveau 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?
  + L’algorithme des k-moyennes, pour k = 2, ne fait pas la différence entre les deux types de cellules différents, car l’une des classes est attribuée aux cellules et l’autre à l’arrière-plan. Autrement, pour classer différents types de cellules, nous pouvons augmenter le nombre de classes à 3.
* Testez les différentes possibilités pour initialiser les classes. Décrivez si possible ces différentes méthodes.
  + Dans l’implémentation du clustering k-means de Scikit Learn, les différentes possibilités d’initialisation des classes sont:
    - sélectionner manuellement les centres initiaux et les transmettre comme arguments à l’algorithme des k-moyennes;
    - choisir des observations au hasard parmi les données comme centroïdes initiaux;
    - utiliser la technique « k-means++ » : un centre initial est choisi uniformément au hasard parmi les points de données, le centre suivant est choisi parmi les points de données avec une probabilité proportionnelle à la contribution du point au potentiel global, cette étape est répétée jusqu’à k Des centres ont été choisis.
* 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.
  + En exécutant plusieurs fois l’algorithme k-means, avec une initialisation aléatoire, nous observons que les centres de classes sont stables.
* Quelles sont les diﬃcultés rencontrées pour la segmentation des différentes fibres musculaires dans l’image **muscle.tif**?
  + L’image **muscle.tif** a été segmentée en trois classes, dans le but de différencier le fond blanc, les fibres de couleur plus claire et celles de couleur plus foncée. En appliquant l’algorithme k-means, nous constatons qu’il ne parvient pas à segmenter avec précision les fibres de couleur plus claire, cela se produit parce qu’elles ont une texture granulaire, donc l’algorithme ne parvient pas à les identifier comme un objet unique.
* Expliquez pourquoi le filtrage de l’image originale (filtre de la moyenne ou filtre median) permet d’améliorer la classification.
  + Le filtrage de l’image avant la classification améliore les performances de l’algorithme k-means car les objets deviennent plus homogènes; le bruit et la texture sont lissés.

### 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.
  + Après quantification, l’image est codée avec seulement 10 couleurs, donc, même si l’on peut encore distinguer les fleurs, certaines informations sont perdues par rapport à l’image originale, cela est évident là où nous avions des dégradés de couleurs, qui se sont discrétisés. De plus, les couleurs les plus fréquentes ont été conservées, tandis que les plus rares, comme le bleu, ont été perdues.
* Quel est le nombre minimum de classes qui donne un rendu visuel similaire à celui de l’image codée sur 3 octets?
* En testant l’algorithme pour un nombre croissant de classes, nous obtenons un résultat visuellement similaire à l’image originale pour 𝑘 = 9 classes.
* Proposez une solution pour retrouver les planches-mères utilisées pour l’impression d’une carte IGN: **carte.tif**.
  + Pour retrouver les planches-mères de la carte **carte.tif**, on pourrait d’abord appliquer un filtre moyen pour lisser les textures hachurées et pointillées, rendant homogènes les zones appartenant à une même classe. Ensuite, l’algorithme k-means pourrait être utilisé pour segmenter la carte en trois classes.

## 4. Seuillage automatique: Otsu

* Dans le script **otsu.py** quel critère cherche-t-on à optimiser?
  + Dans le script python, le critère optimisé est la minimisation de la variance intra-classe.
* Testez la méthode de Otsu sur différentes images à niveaux de gris, et commentez les résultats.
  + En appliquant la méthode de segmentation d’Otsu pour deux classes à différentes images en niveaux de gris, nous constatons qu’elle ne parvient pas toujours à distinguer les objets de l’arrière-plan, cela est évident pour l’image représentant les fibres musculaires.
* Cette méthode permet-elle de seuiller correctement une image de norme du gradient?
  + La méthode d’Otsu peut être utilisée avec succès pour détecter les contours en choisissant un seuil adéquat pour la norme du gradient.
* Modifiez ls script **otsu.py** pour traiter le problème à trois classes, i.e. la recherche de deux seuils
* def otsu\_thresh\_modified(im):  
   h = histogram(im)  
    
   m = 0  
   for i in range(256):  
   m = m + i \* h[i]  
    
   mint1 = 0  
   mint2 = 0  
   minv = np.inf  
    
   for t1 in range(1, 256 - 2):  
   for t2 in range(t1 + 1, 256 - 1):  
   w0 = 0  
   w1 = 0  
   w2 = 0  
   m0 = 0  
   m1 = 0  
   m2 = 0  
   v0 = 0  
   v1 = 0  
   v2 = 0  
    
   for i in range(t1):  
   w0 = w0 + h[i]  
   m0 = m0 + i \* h[i]  
   if w0 > 0:  
   m0 = m0 / w0  
   for i in range(t1):  
   v0 = v0 + h[i] \* (i - m0) \*\* 2  
   if w1 > 0:  
   v0 = v0 / w0  
    
   for i in range(t1, t2):  
   w1 = w1 + h[i]  
   m1 = m1 + i \* h[i]  
   if w1 > 0:  
   m1 = m1 / w1  
   for i in range(t1, t2):  
   v1 = v1 + h[i] \* (i - m1) \*\* 2  
   if w1 > 0:  
   v1 = v1 / w1  
    
   for i in range(t2, 256):  
   w2 = w2 + h[i]  
   m2 = m2 + i \* h[i]  
   if w2 > 0:  
   m2 = m2 / w2  
   for i in range(t2, 256):  
   v2 = v2 + h[i] \* (i - m2) \*\* 2  
   if w2 > 0:  
   v2 = v2 / w2  
    
   v = w0 \* v0 + w1 \* v1 + w2 \* v2  
    
   if v < minv:  
   mint1 = t1  
   mint2 = t2  
   minv = v  
    
   return (mint1, mint2)

## 5. Croissance de régions

* Quelles contraintes doit vérifier un pixel pour être ajouté à l’objet existant?
  + Dans le script python **region\_growing.py**, un pixel à proximité d’un objet existant y est ajouté si la moyenne locale calculée sur le voisinage du pixel est comprise dans un seuil donné de la moyenne locale du germe d’origine.
* 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.
  + Plus le seuil est petit, plus le prédicat est strict, donc seuls des pixels très similaires au germe d’origine sont ajoutés à l’objet, de cette façon, la région résultante pourrait être plus petite que l’objet réel et pourrait afficher des trous indésirables, car seules de légères dishomogénéités seront assez pour supprimer un pixel. En augmentant le seuil, le prédicat devient plus doux, permettant une plus grande croissance de la région. Si le seuil est trop grand, la région pourrait dépasser les frontières du véritable objet.
* Quel est l’effet du paramètre *thresh* sur le résultat de segmentation?
  + Le paramètre **thres** correspond à l’écart acceptable entre la couleur de la zone considérée et celle de la zone en expansion. Ainsi, plus **thres** est grand, plus l’objet final sera étendu, étant donné que l’on permet un plus grand écart de couleur.
* Quels paramètres permettent de segmenter correctement la matière blanche?
  + Les paramètres suivants permettent la segmentation correcte de la substance blanche :
    - (x0, y0) = (300, 300)
    - seuil = 4
    - rayon = 5
* Parvenez-vous à segmenter la matière grise également?
  + En choisissant le bon germe, nous pouvons également segmenter la matière grise, même si seules les parties liées au germe d’origine sont détectées. Les zones disjointes de matière grise, comme celles proches du centre du cerveau, ne sont pas détectées.
* Quel est le prédicat mis en place dans ce script?
  + Le prédicat implémenté est « la région est homogène », et l’argument utilisé est que la moyenne locale du voisinage de chaque pixel doit être inférieure au temps sigma de la moyenne locale du voisinage du germe, où sigma est la norme déviation du voisinage du germe.
* Proposez un autre algorithme qui n’utilise pas la croissance de régions, mais qui donne le même résultat.
  + Étant donné le germe initial et les valeurs moyennes et standards de son voisinage, l’algorithme pourrait simplement vérifier pour chaque pixel de l’image si sa moyenne locale se situe dans le seuil souhaité de la moyenne locale du germe d’origine.
* Proposez un prédicat qui nécessite réellement un algorithme de croissance de région.
  + Une adaptation du prédicat qui exigerait l’utilisation de l’algorithme de croissance de région serait « la région est homogène, connectée et contient le point initial ».