Traitement d'images, TP6
Détection de contours

ENSEEIHT 2EN

`http://oberlin.perso.enseeiht.fr/teaching.html`

1. Télécharger les images du jour à l'adresse `http://oberlin.perso.enseeiht.fr/cours/images_TP6.zip`. Lire et afficher l'image **circles.png**. Calculer son gradient discret, avec la formule centrée vue en cours. Afficher le module des composantes horizontales et verticales du gradient, ainsi que sa norme.
2. Pour réduire la sensibilité au bruit, on peut faire un lissage dans la direction perpendiculaire à la dérivation. Tester cette méthode, en implémentant les filtres de Prewitt (lissage uniforme de taille 3) et de Sobel (lissage binomial de taille 3). Tester également un lissage plus fort. Comparer les images des normes du gradient obtenues par ces différentes méthodes : pour mettre en évidence les différences, on zoomera sur le patch $(330 : 400, 240 : 310)$.
3. Sélectionnez à présent des points de contour en faisant un seuillage de la norme du gradient (calculé avec le filtre de Sobel), et affichez-les. Tester différentes valeurs de seuil, et commenter les résultats.
4. On s'intéresse à présent aux passages par 0 du Laplacien. Comme on l'a vu en cours, le Laplacien étant très sensible au bruit il faut auparavant lisser l'image avec un filtre Gaussien h_σ. La méthode revient donc à estimer les passages par 0 de $g = f \star \Delta h_\sigma$.
   - ◇ Écrire $\hat{g}$ dans le domaine de Fourier. On pourra négliger les constantes, puisqu'on s'intéresse seulement aux passages par 0 de g.
   - ◇ Mettre en oeuvre le filtrage en Fourier sur **Lena**, et afficher les contours correspondants. Pour estimer les passages par 0, on utilisera la fonction **contour(g,[0 0])**.
   - ◇ Afficher les contours de Lena pour différents niveaux de lissages ; quel est l'intérêt de cette détection multi-échelle ?
5. Mettez-en oeuvre la méthode de détection de Cany en utilisant la fonction **edge** de Matlab. Tester sur l'image Lena. Expliquer brièvement le rôle des deux paramètres.
6. (Bonus) Tester la méthode des contours actifs en Matlab, avec la commande **imageSegmenter**.