TD IV : Segmentation en contours et régions

IV. 1/ Détection de contours

a/ Ecrire un programme qui calcule l'image du gradient `Ig` grâce aux opérateurs de Sobel et Prewit pour l'image `'carte.png'`. Convertir cette image RGB en niveaux de gris (`rgb2gray()`). Visualiser l'image du gradient.

Seuiller cette image du gradient afin d'obtenir l'image des contours `Ic`. Afficher `Ic`. Observer l'effet du seuil sur les contours. Quel est le format des données dans l'image `Ig` et dans l'image `Ic` ?

b/ Grâce à l'instruction `edge()` ci-dessous, calculer l'image de contours en utilisant différents opérateurs de calcul du gradient (Canny, Sobel, Prewitt, Roberts, …) et différentes valeurs pour le seuil.

```
Ic = edge(I,'canny',.1);
```

IV.2/ Approximation polygonale de droites

On cherche à sélectionner uniquement contours qui ressemblent à des segments de droites. On utilisera pour cela la transformée de Hough (annexe 2).

a/ Ecrire le programme permettant de remplir la table de Houh `H` pour l'image de contours `Ic` :

```
[H,T,R] = hough(Ic,'RhoResolution',1,'ThetaResolution',1);
```

Visualiser `H` en ajustant les valeurs comme indiqué ci-dessous :

```
figure(2), imshow(imadjust(mat2gray(H)),'XData',T,'YData',R);
title('Hough transform table');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
```

b/ Les droites suffisamment longues correspondent à des *maxima* locaux de la table de Hough. Détecter par exemple les segments les plus longs grâce à l'instruction ci-dessous qui sélectionne par exemple les 20 plus hauts pics de la table :

```
P  = houghpeaks(H,20);
```

Afficher ces points sur la table de Hough (Les coordonnées des points décrivant chaque droite sont les éléments des matrices `T` et `R` indexées par `P`).

b/ Détecter les segments de droites correspondant aux maximas de la table de Hough grâce à l'instruction ci-dessous. Il convient de choisir les segments ayant une longueur minimale donnée. Les segments plus petits peuvent être agrégés par un comblement des lacunes inférieures à un nombre de pixels spécifié.

```
lines = houghlines(Ic,T,R,P,'FillGap',5,'MinLength',50);
```

Les données de la structure `lines` représentent la liste des segments ainsi que les extrémités de ces segments. Pour afficher le $k^{ème}$ segment on tape :

```
xy = [lines(k).point1; lines(k).point2];
plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
```

c/ Tracer les segments détectés sur l'image puis faire le lien entre chaque point de la table de Hough et la droite correspondante dans l'image.

IV.3/ Approximation polygonale de cercles

L'approche précédente peut être généralisée à la détection de cercles. La table de Hough dans ce cas recueillera, pour chaque pixel, les votes de tous les cercles d'un rayon donné auxquels il peut appartenir. Le seuillage des pics de la table obtenue après le vote permet ainsi de détecter les cercles présents dans l'image de contours.

a/ Charger l'image `' monnaie.png'`, détecter ces contours `Ic`, puis créer grâce à la fonction la `circle_hough()`, la table de Hough `H` pour les cercles de rayon `r` dans l'intervalle [20,30] pixels (prendre une marge de plus ou moins 5 pixels) :

```
r = 15:1:40;
H = circle_hough(Ic,r,'same','normalise');
```

Avec `H(i,j,k)` contenant le nombre de votes pour le cercle centré sur `(i,j)`, avec un rayon `r(k)`.

b/ Sélectionner les 10 cercles les ayant le plus de votes grâce à la fonction `circle_houghpeaks()` :

```
peaks = circle_houghpeaks(H,r,'nhoodxy',15,'nhoodr',21,'npeaks',10);
```

c/ Afficher les cercles dans l'image grâce à la fonction `circlepoints()`. Cette fonction retourne les points d'un cercle avec un rayon donné :

```
[x,y] = circlepoints(peak(3));
```

Refaire le même travail sur l'image `'oeil.png'` en réglant les paramètres de sorte à localiser l'iris et la pupille.

IV. 4/ Segmentation en régions

a/ Ecrire un programme capable de détecter les zones homogènes (régions). La segmentation consiste à parcourir l'image et pour chaque pixel :

- Si non déjà labellisé, ajouter un label (nouveau numéro de région).
- Donner le même label à tous ses voisins non labellisés vérifiant un critère d'homogénéité du niveau de gris.

Dans une seconde étape, on parcourt l'image labellisée pour fusionner, si nécessaire, toutes les régions connexes (c'est-à-dire ayant au moins deux pixels voisins).
Tester le code sur les images `'piece.png'`,`'monnaie.png'` et `'oeil.png'` en adaptant les paramètres.