TP4 : Par ici la monnaie

Développer une application capable de déterminer automatiquement le montant en Euros d'un ensemble de pièces de monnaies posées sur une table.

La reconnaissance des différentes pièces de monnaies sera basée sur le fait que chacune d'entre elle est caractérisée par un diamètre normalisé.

Étape 1 : Détecter les pièces de monnaie dans l'image
Fichiers image de test : *pieces1.jpg, pieces2.jpg, pieces3.jpg*

L'image RGB de la scène contient trop d'informations pour être directement exploitable, il est donc nécessaire de la transformer en image à niveaux de gris (`rgb2gray`), puis de la seuiller (`graythresh`, `im2bw`) afin d'obtenir une image binaire dont les `1` correspondent aux pixels des pièces de monnaie et les `0` au fond de la scène.

L'image binaire obtenue fait apparaître de nombreux 'trous' du fait de la non homogénéité des surfaces des pièces. En prenant en compte des considérations topologiques (connections entre pixels), il est possible de les combler (`imfill`) afin d'obtenir des régions pleines. Ainsi tout ensemble de pixels connexes à `0` ne pouvant être connectés aux pixels à `0` situés aux bords de l'image sera mis à `1`.

```
% Exemple de remplissage
Ibw1 = logical([1 1 1 0 0 0 0 0
                1 0 1 0 0 0 1 1
                1 1 1 1 0 1 0 1
                1 1 1 1 0 1 0 1
                1 0 0 1 0 1 1 1
                1 0 1 1 0 1 1 1
                1 0 1 0 0 0 1 1
                1 1 1 0 0 0 0 0])
Ibw2 = imfill(Ibw1,'holes')
```

La texture des objets de la scène, les conditions d'acquisition ainsi que la valeur des paramètres des différentes étapes de notre chaîne de traitement conduisent inévitablement à la génération de régions de très petite taille ne correspondant pas aux pièces de monnaie que nous avons choisi de détecter. Afin de les supprimer, appliquer une ouverture morphologique (`imopen`) avec un élément structurant (`strel`) adapté à la taille des objets que l'on souhaite supprimer.

On peut alors rechercher pour chaque région les pixels lui correspondant en analysant les composantes connexes de l'image binaire (`bwlabel`). Un affichage satisfaisant de l'image des régions détectées peut se faire grâce à `label2rgb`.

```
% Exemple de recherche des composantes connexes
Ilab1 = bwlabel(Ibw2,4)
Ilab2 = bwlabel(Ibw2,8)
```

Afin d'identifier les différents types de pièces présentes dans l'image, nous allons calculer deux caractéristiques géométriques (`regionprops`) pour chaque région, à savoir son aire et les coordonnées 2D de son barycentre.

```
% Exemple de calcul de caractéristiques géométriques de composantes connexes
stats = regionprops(Ilab2,'Area','centroid')
```

Afin de valider cette première étape, afficher (`text`) sur l'image des régions détectées un '+' au centre de chaque région susceptible de correspondre à une pièce de monnaie ainsi que son indice dans la structure retournée par `regionprops` et son aire en pixels.

Étape 2 : Introduire un facteur d'échelle

Il s'agit de déterminer l'aire théorique en pixels des huit types possibles de pièces de monnaie. L'utilisateur va saisir l'indice d'une pièce de monnaie puis en indiquer la valeur en euros (`input`).

Pour la pièce sélectionnée, estimer son diamètre en pixel à partir de la valeur de son aire. Connaissant la valeur de la pièce sélectionnée et donc son diamètre en mm (cf. tableau normalisé donné ci-dessous), calculer la taille en mm d'un pixel. En déduire le diamètre en pixels des huit types possibles de pièces ainsi que leur aire théorique en pixels.

Valeur (€)	2	1	0.50	0.20	0.10	0.05	0.02	0.01
∅ (mm)	25.75	23.25	24.25	22.25	19.75	21.25	18.75	16.25

Étape 3 : Identifier individuellement chaque pièce de monnaie

Il ne reste plus qu'à déterminer pour chaque région l'aire théorique qui lui est la plus proche afin d'identifier la valeur de la pièce de monnaie correspondante.

Vérifier le bon fonctionnement du système en affichant sur l'image RGB la valeur en euros au centre de chaque pièce.

Tester sur les 3 fichiers image.

Étape 4 : Des disques venus d'ailleurs …

Que se passe-t-il si vous appliquez votre code aux fichiers *disques1.jpg* et *disques2.jpg* ? Pourquoi ?

On pose alors comme hypothèse que les régions d'une aire suffisamment inférieure à celle d'une pièce de 1 centime ou suffisamment supérieure à celle d'une pièce de 2 euros ne doivent pas être considérés comme des pièces mais comme des intrus. On suppose que l'utilisateur choisira toujours une pièce de monnaie lors de l'étape 2.

Modifiez votre code en conséquence et testez-le sur les 5 fichiers image.

Étape 5 : Encore des intrus !

Cette fois ci, il ne s'agit plus de disques mais d'objets disparates : *intrus1.jpg, intrus2.jpg, intrus3.jpg, tronquees.jpg*. Pourquoi votre code détecte-t-il des pièces là où il n'y en a pas (et se trompe lorsqu'elles sont en bordure d'image) ?

Afin d'éliminer ces régions non circulaires, il peut être judicieux de recourir à deux caractéristiques géométriques supplémentaires (longueur du grand et du petit axe de l'ellipse approchant la région) afin de calculer un rapport représentatif de la circularité de la région.

Modifiez votre code en conséquence et testez-le sur les 9 fichiers.

Étape 6 : Trop proches.

Les ennuis continuent, voilà que certaines pièces sont bord à bord : *contact1.jpg, contact2.jpg, contact3.jpg, contact4.jpg*. Quels soucis ce phénomène pose-t-il ?

Pour dissocier les régions en contact, nous allons recourir à une érosion à l'aide d'un élément structurant adapté aux dimensions des pièces de monnaie. Puis, une fois les régions étiquetées individuellement, il est possible de retrouver sensiblement les caractéristiques géométriques initiales de chacune d'entre elle en réalisant une dilation avec l'élément structurant ayant été utilisé pour l'érosion.

```
% Exemple d'érosion suivi d'une dilation individuelle par région pour le calcul
des caractéristiques géométriques de composantes connexes
Ibw = logical([1 1 1 0 0 0 0 0
               1 1 1 0 0 0 1 1
               1 1 1 1 0 1 1 1
               1 1 1 1 1 1 1 1
               1 1 1 1 1 1 1 1
               1 1 1 1 0 1 1 1
               1 1 1 0 0 0 1 1
               1 1 1 0 0 0 0 0])
E=strel('disk',1);
Ie = imerode(Ibw,E)
stats = regionprops(bwlabel(Ie,8),'Area','centroid');

for ind=1:length(stats)
      stats(ind).Area
      Iregion=bwselect(Ie, stats(ind).Centroid(1), stats(ind).Centroid(2));
      Iregion=imdilate(Iregion,E)
      stats(ind)=regionprops(bwlabel(Iregion),'Area','Centroid');
      stats(ind).Area
end
```

Modifiez votre code en conséquence et testez-le sur les 13 fichiers.

Étape 7 : Je vais prendre un peu de tout finalement ...

Si tout va bien votre code devrait être capable de traiter les images *vrac1.jpg, vrac2.jpg, vrac3.jpg, vrac4.jpg*. Mais qu'en est-il avec *bonus.png* ? Proposez une solution (si elle existe).