M4201Cin TP 1

Recherche de contours — Segmentation

28 février 2018 – B. COLOMBEL

Les réponses aux questios des exercices 1 à 4 seront écrites dans l'éditeur Scinote dans un seul fichier.
Les réponses autres que des fonctions Scilab seront commentées. Il sera enregistré sous le nom :

`Nom_Prenom-TP4.sci`

et déposé dans le dossier prévu à cet effet dans AMETICE (module M3201Cin, TP4)

Si le module SIVP de **Scilab** est bien installé, vous verrez au démarrage :

```
Initialisation :
  Chargement de l'environnement de travail

SIVP - Scilab Image and Video Processing Toolbox
        load macros
        load gateways
        load help
        load demos
```

Sinon, installez le :

Applications » Gestionnaire de modules - ATOMS » Traitements des images » SIVP

Cliquer sur [installer] puis redémarrer **Scilab**.

1 Filtres de Sobel : Exemple de recherche de contours

Exercice 1 : **Filtre de Sobel.**
On récupère le filtre de Sobel horizontal S_x avec :

```
--> S = fspecial('sobel')
```

Pour obtenir S_y, on détermine sa transposée `S'`.

```
--> S2 = S'
```

1. Définir le filtre de Sobel horizontal S_x puis le visualiser :
   ```
   --> n = 3; plot3d(1:n, 1:n, S);
   ```
2. Écrire une fonction `res = grad(image0)` en **Scilab** qui calcule le module du gradient obtenu à partir des convolutions $I_x = I * S_x$ et $I_y = I * S_y$.
3. Écrire une fonction `res = seuil(image)` en **Scilab** permettant le seuillage de ce module afin de produire un filtre détecteur de contours du premier ordre.

⚠ Pour visualiser le résultat, les coefficients du résultat doievnt être des entiers compros entre 0 et 255. Pour cela, on peut utiliser la fonction `mat2gray(X)` qui normalise l'image `X` entre 0 et 1 puis multiplier le résultat par 255 :

```
--> res = uint8(255*mat2gray(X));
```

4. Tester votre fonction sur l'image `lena.pgm`.

2 Segmentation

La segmentation d'image est une opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères pré-définis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond. Il existe de nombreux types de méthodes de segmentation d'image : certaines sont basées sur les contours, d'autres sur un seuillage des pixels en fonction de leur intensité, d'autres découpent l'image en régions connexes , etc.

Nous allons ici considérer une image composée de pièces de monnaie sur un fond relativement uniforme, et tenter de segmenter cette image avec différentes techniques.

2.1 Segmentation par seuillage de valeurs

Nous allons dans un premier temps réaliser une segmentation grâce à un seuillage des valeurs des pixels. Notre but ici est que les pièces apparaissent en blanc et que le fond devienne noir. Nous allons donc créer une image binaire booléenne, composée uniquement de 0 et de 255. Nous allons donc définir deux seuils T_{min} et T_{max} : tous les pixels dont la valeur sera comprise entre T_{min} et T_{max} seront mis à 255 (blanc), tandis que tous les autres seront mis à 0 (noir). La question restante est donc : comment choisir ces seuils ?

Exercice 2 :

1. Ouvrir l'image `coins.pgm`, la stocker dans une matrice X. Afficher l'image.
2. Tracer l'histogramme de l'image et identifier les différents pics de l'histogramme.
   Quelles sont les zones correspondant au fond ? aux pièces ?
3. En observant l'histogramme, déterminer les seuils T_{min} et T_{max} à utiliser.
4. Créer une fonction `Xbin = bin1(X, Tmin, Tmax)` qui retourne une image binaire booléenne X_{bin} de même taille que X, valant 255 pour les pixels de X compris entre T_{min} et T_{max} et 0 sinon. La zone correspondant au fond sera donc noire, et les zones correspondant aux pièces seront blanches. Afficher X et X_{bin} sur la même figure.
5. Tester plusieurs valeurs de T_{min} et T_{max} jusqu'à avoir une segmentation de bonne qualité.
6. S'il reste des points isolés qui ne sont pas de la bonne couleur, on peut appliquer un filtrage médian sur l'image X_{bin}.
   Réaliser cette opération : vous devriez obtenir une segmentation parfaite.

2.2 Segmentation par détection de contours

Nous allons maintenant essayer de réaliser une autre segmentation qui va se baser non pas sur les valeurs des pixels, mais sur une détection de contours. Nous n'allons donc pas travailler sur l'image originelle, mais sur une image filtrée ayant des contours rehaussés. Nous allons définir un seuil T : tous les pixels de l'image réhaussée dont la valeur est supérieure à T seront mis à 255, tandis que tous les autres seronts mis à 0. Les contours des pièces seront donc en blanc et le reste en noir.

Exercice 3 :

1. Ouvrir l'image `coins.pgm`, la stocker dans une matrice Y. Afficher l'image.

2. Appliquer la fonction `grad` de l'exercice 1 pour déterminer le gradient Y_1 de Y.
3. Tracer l'histogramme de Y_1 afin de déterminer le seuil T à utiliser.
4. Créer fonction `Ybin = bin2(Y, T)` qui retourne une image Y_{bin} de même taille que Y valant 255 pour les pixels de Y_1 supérieurs à T et 0 sinon. Les contours des pièces seront donc blancs et tout le reste de l'image sera noir. Afficher X et Y_{bin} sur la même figure.
5. Tester plusieurs valeurs de T jusqu'à voir apparaître tous les contours des pièces.

2.3 Fusion des deux méthodes

Nous avons étudié deux méthodes de segmentation : la première méthode nous a permis de faire une image noire avec les pièces de couleur blanche, alors que la deuxième nous a permis de trouver les contours des pièces.
En revanche la deuxième méthode a donné de moins bons résultats : alors comment obtenir des contours de pièces de bonne qualité ? Une des façon de résoudre le problème est de fusionner les résultats. Au lieu de détecter les contours sur l'image originelle, nous allons détecter les contours sur l'image déjà segmentée par la première méthode.
Exercice 4 :

1. Ouvrir l'image `coins.png`, la stocker dans une matrice Z.
2. Générer l'image Z_1 obtenue par la méthode de la partie 2.1.
3. Générer l'image Z_{bin} obtenue par la méthode de la partie 2.2 en prenant en entrée l'image Z_1 convertie en double (et pas Z).
4. Afficher Z et Z_{bin} sur une même figure : le résultat doit être parfait.