TD – Traitement d'images

2ème partie

Introduction et objectifs

Dans cette deuxième partie du TD consacré au traitement d'images, on va se concentrer sur un thème classique : la détection des contours.

On travaillera classiquement avec des images en nuances de gris, en environnement Scilab (avec SIVP) ou en environnement Python (avec les bibliothèques matplotlib (module pyplot), numpy et scipy (module ndimage).

Seuillage global simple

On se donne une image I en nuances de gris et on se fixe un seuil (threshold) s. ($0.0 < s < 1.0$ sous Scilab ou sous Python). Si on suppose que l'intensité I est normalisée (elle prend ses valeurs dans l'intervalle $[0\,;1]$), on souhaite obtenir une image J en noir et blanc telle que :

$$J(x,y) = \begin{cases} 1.0 & \text{si } I(x,y) \geq s \\ 0.0 & \text{si } I(x,y) < s \end{cases}$$

En environnement Scilab (SIVP), on utilisera la fonction `im2bw` et en environnement Python, on utilisera la fonction `where` (TRES pratique !) de la bibliothèque numpy.

Mise en œuvre des filtres

Mettre en œuvre les filtres de Prewitt, Sobel et Laplace en utilisant la console ou en écrivant des scripts simples.

Filtres de Prewitt et de Sobel

Sous Scilab (SIVP), on utilisera la fonction `fspecial` qui renvoie respectivement les filtres :

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ et } \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Ils visent donc à détecter les contours horizontaux. Pour la détection des contours verticaux, on utilisera leurs transposés.

Sous Python, les fonctions `prewitt` et `sobel` peuvent recevoir comme argument la direction de la dérivation :

`prewitt(matrice,0)` correspond à la mise en œuvre du filtre $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$.

`prewitt(matrice,1)` correspond à la mise en œuvre du filtre $\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$.

`sobel(matrice,0)` correspond à la mise en œuvre du filtre $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$.

`sobel(matrice,1)` correspond à la mise en œuvre du filtre $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$.

Filtre de Laplace

Sous Scilab (SIVP), le masque mis en œuvre via la fonction `fspecial` est :

$$\frac{1}{1+\alpha}\begin{pmatrix} \alpha & 1-\alpha & \alpha \\ 1-\alpha & -4 & 1-\alpha \\ \alpha & 1-\alpha & \alpha \end{pmatrix}$$

Le filtre vu en cours (cf. ci-dessous) correspond donc au cas $\alpha = 0$.
On pourra tester différentes valeurs du paramètre α.

Sous Python, le filtre mis en œuvre via la fonction `laplace` est le filtre vu en cours :

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Extraction des contours

On travaillera dans cette partie avec le filtre de Sobel.

On écrira un script qui effectuera les opérations suivantes :

- Lecture d'une image couleur.
- Obtention de l'image en nuances de gris correspondante.
- Lissage à l'aide d'un filtre gaussien.
- Application de deux filtres de Sobel (obtention des composantes G_X et G_Y) puis calcul du tableau des modules normalisés des gradients.
- Seuillage simple pour obtention d'une image noir et blanc des contours.
- [**Facultatif**] combinaison des images en nuances de gris et noir et blanc pour l'obtention d'une image en nuances de gris avec contours blancs.
- Affichage final.

Quelques pistes pour aller plus loin

1. Le seuillage présenté dans ce TD est le plus simple que l'on puisse imaginer ! Il en existe d'autres !

2. Afin d'améliorer les scripts de détection des contours, notamment en se souciant de la présence ou de l'absence de la première étape de lissage de l'image initiale, on pourra ajouter à celle-ci un bruit artificiel.

   Sous Scilab (SIVP), on utilisera la fonction `imnoise`.

3. Au-delà des approches relevant du domaine du traitement du signal (par exemple les filtrages dont nous avons parlé), on pourra s'intéresser à une approche plus ensembliste (et plus récente puisqu'elle date de la fin des années 70) du traitement d'image appelée « Morphologie mathématique ».

   Sous Python, on utilisera le module morphology de scipy.ndimage.

4. On pourra s'essayer à la filtration non linéaire :
   a. Filtre de Nagao.
   b. Filtre médian (revoilà le tri …).

5. Plus généralement, les environnements de traitement d'image permettent l'application de filtres personnalisés. N'hésitez pas à en chercher/inventer !