# Thème Image - TP4 - Filtres différentiels, détection de contours (1ère partie)

## 1 - Préambule

### 1.1 - Déroulement du TP

Ce TP noté comporte deux parties correspondant chacune à une séance environ. Un unique compte-rendu sera demandé pour ces deux parties, il faudra le déposer sur chamilo avant la date indiquée par votre enseignant. Le dépôt devra se faire sous la forme d'une archive (zip ou rar) qui contiendra le compte-rendu en pdf ainsi que les codes scilab. Le compte-rendu devra comporter les réponses aux exercices des deux parties.

## Instructions pour la rédaction du compte-rendu

- L'intégralité de votre code devra être fourni. Vous serez attentifs à commenter et indenter le code pour faciliter la lecture.
- Dans le corps du compte-rendu, vous pourrez recopier des portions de code si cela vous semble opportun.
- Il est important d'illustrer vos résultats avec les images produites en indiquant à chaque fois les paramètres utilisés. Ce n'est pas précisé à chaque fois dans les exercices mais c'est néanmoins ce qu'on attend.
- De manière générale, soyez attentifs à la présentation des résultats et à la rédaction.

### 1.2 - Utilisation des filtres avec `scilab`

A partir de maintenant vous utiliserez la routine `conv2(im, H)` qui est intégrée à `scilab` pour appliquer un filtre `H` à une image `im`. Par rapport à la routine `moyenne2D_W` que vous avez programmée, `conv2` s'exécute plus rapidement. Exemple d'utilisation :

```
exec('init_tp_image.sce');
im1 = lire_imageBMPgris('papillon.bmp');
// filtre moyenne
M = 1/9*ones(3, 3);
im2 = conv2(im1, M, "same");
```

L'option `"same"` est importante pour que `im2` ait la même taille que `im1`.

### 1.3 - Filtres gaussiens

Le filtrage gaussien permet d'améliorer la qualité des contours dans de nombreux cas. Vous en aurez besoin pour le dernier exercice de cette séance, ainsi qu'à la séance prochaine. La fonction `W_gauss_2D` vous est fournie dans le fichier `filtres2D.sci`.

## 2 - Dérivées de l'image

**Image continue.** On munit le plan du système de coordonnées habituel $(x, y)$. Une image continue est vue comme une fonction

$$I : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R} \\ (x,y) & \mapsto & I(x,y), \end{array}$$

où $I(x, y)$ correspond à l'intensité lumineuse au point $(x, y)$. On note $\frac{\partial I}{\partial x}$ et $\frac{\partial I}{\partial y}$ les dérivées partielles (qu'on suppose définies partout) et grad $I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ le gradient de $I$. La norme du gradient

$$\|\text{grad } I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

est donc une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$.

**Image numérique.** Soit `im` une image numérique donnée sous la forme d'un tableau à deux dimensions contenant des entiers de 0 à 255. L'équivalent numérique des dérivées partielles de l'image est obtenu par convolution avec les filtres `Dx = 0.5*[-1, 0, 1]` pour la direction $x$ et `Dy = 0.5*[1; 0; -1]` pour la direction $y$. Notez bien que `Dx` est un vecteur ligne et `Dy` un vecteur colonne. On obtient ainsi deux tableaux de nombres :

```
imx = conv2(im, Dx, "same");
imy = conv2(im, Dy, "same");
```

Pour chaque pixel `(u,v)`, le norme du gradient est définie par

$$\texttt{imn}(\texttt{u},\texttt{v}) = \sqrt{\texttt{imx}(\texttt{u},\texttt{v})^2 + \texttt{imy}(\texttt{u},\texttt{v})^2}. \tag{1}$$

**Exercice 1 :**

- Dans le fichier `contours.sci`, complétez la fonction `norme_gradient(im)` pour qu'elle renvoie la norme `imn` du gradient de l'image `im`, selon la formule (1).
- Pour tester le résultat, visualisez la norme du gradient :
```
exec('init_tp_image.sce');
exec('contours.sci');
im = lire_imageBMPgris('sweets.bmp');
imn = norme_gradient(im);
afficher_image(int(imn));
```
- Améliorez la visualisation en ramenant les valeurs de `imn` dans l'intervalle $\{0, \ldots, 255\}$. Autrement dit, corrigez le contraste de l'image `imn`.

## 3 - Détection des contours par seuillage

Une première méthode de détection des contours consiste à les définir comme les points de l'image où la norme du gradient dépasse une certaine valeur `seuil`. L'image `imc` contenant les contours est définie par

$$\forall(\texttt{u},\texttt{v}), \texttt{imc}(\texttt{u},\texttt{v}) = \begin{cases} 0 & \texttt{si} \quad \texttt{imn}(\texttt{u},\texttt{v}) < \texttt{seuil}, \\ 255 & \texttt{sinon}, \end{cases}$$

où `imn` est l'image contenant la norme du gradient de l'image `im`.

**Exercice 2 :**

— Dans le fichier `contours.sci`, complétez la fonction `contours_seuil(im, seuil)` de manière à ce qu'elle renvoie l'image `imc` telle que définie ci-dessus.

— Testez la fonction :

```
exec('contours.sci', -1);
im = lire_imageBMPgris('barque.bmp');
seuil = 20;
imc = contours_seuil(im, seuil);
afficher_image(imc);
```

Vous pouvez faire varier la valeur du paramètre `seuil` pour évaluer son influence sur le résultat.

Le seuil peut varier selon l'image, mais on peut automatiser la recherche de ce seuil en fonction de la répartition des valeurs de `imn`. Etant donné que l'on veut conserver les valeurs les plus élevées de la norme du gradient, on peut fixer le seuil de manière à ce qu'un certain pourcentage `p` des pixels les plus sombres soit éliminé. Soit `Hc` le tableau contenant l'histogramme cumulé de `imn` ; on rappelle que `Hc(v)` représente le nombre de pixels dont la valeur est inférieure ou égale à `v-1`, et que `Hc(256)` représente le nombre total de pixels de l'image. Ainsi, `Hc(v)/Hc(256)` représente le pourcentage de pixels de l'image dont la valeur est inférieure à `v-1`. Le seuil est donc choisi comme la plus petite valeur `v` telle que `Hc(v)/Hc(256) > p`.

**Exercice 3 :**

— Dans le fichier `contours.sci` complétez la fonction `trouver_seuil(im, p)` pour qu'elle mette en oeuvre la recherche automatique de seuil en fonction d'un pourcentage `p` et renvoie sa valeur. Pour la tester :

```
exec('contours.sci', -1);
p = 0.8;
im = lire_imageBMPgris('barque.bmp');
seuil = trouver_seuil(im, p);
disp(seuil);
```

Faites varier `p` et vérifiez que `seuil` varie de manière cohérente.
*Il y a une fonction* `hist_cumul(im)` *dans le fichier* `contours.sci`, *vous pouvez l'utiliser pour cette question.*

— Vous pouvez ensuite tester le résultat en terme de contours avec la fonction `contours_p(im, p)` qui vous est fournie. Elle détermine les contours à partir d'un pourcentage `p` en se basant sur les fonctions `contours_seuil` et `trouver_seuil` :

```
p = 0.8;
afficher_image(contours_p(im, p));
```

Puis vous pouvez tester avec d'autres valeurs de `p` et différentes images.

## 4 - Lissage et contours

Il est souvent intéressant d'appliquer un filtre de lissage avant de calculer la norme du gradient, de manière à ne garder que les variations les plus significatives.

**Exercice 4 :**

Pour les images du tableau ci-dessous, testez différentes valeurs du seuil `p` et du paramètre `sigma` de la gaussienne ; vous pouvez faire ces opérations et afficher le résultat en une ligne :

```
afficher_image(contours_p(conv2(im, W_gauss_2D(sigma), "same"), p));
```

Pour chacune des images, indiquez dans le tableau les valeurs testées et mettez en gras celle qui vous paraît la mieux adaptée.

| | `barque.bmp` | `barque_bruit.bmp` | `plage.bmp` | `sweets.bmp` |
|---|---|---|---|---|
| `sigma` | | | | |
| `p` | | | | |

N'oubliez pas d'illustrer votre compte-rendu avec les images obtenues !