Thème Image - TP3 - Introduction au filtrage

Objectifs du TP

— Aborder concrètement la notion de filtrage, d'abord en 1D puis en 2D
— Etudier deux filtres de lissage : filtre moyenne et filtre gaussien
— Application : lissage d'une image bruitée

Image originale

Image bruitée

— Comparer l'effet des deux filtres

1 - Filtrage 1D

On s'intéresse dans un premier temps au filtrage de signaux 1D. Dans le cadre du filtrage numérique, on travaille sur des vecteurs de nombres. L'idée du filtrage est de promener une "fenêtre" sur chacun des éléments du vecteur, et obtenir une nouvelle valeur à partir des éléments de cette fenêtre.

1.1 - Construction de la fenêtre et parcours du signal

Exercice 1 :

Soit `s` un vecteur de taille `N`.

1. Etant donné un indice `u` et un entier `m`, comment construire avec `scilab` le sous-tableau de `s` contenant les `2m+1` éléments suivants :

   `s(u-m), ..., s(u), ..., s(u+m)` ?

   Pour quelles valeurs de `u` est-il possible de construire ce sous-tableau ?

2. Dans le fichier `filtres1D.sci`, modifiez la fonction `afficher_fenetre(s, m)` pour qu'elle parcourt les éléments de `s` et affiche quand c'est possible la fenêtre de taille `2m+1` autour de l'élément courant. Vous pouvez la tester avec le code suivant :

```
// on charge les fonctions definies dans filtres1D.sci
exec('filtres1D.sci')
// definition du signal :
s = 0:7;
// affichage des fenetres de taille 2m+1
afficher_fenetre(s, m);
```

   Vous devez obtenir les résultats suivants :

```
--> afficher_fenetre(s, 1)
   0.   1.   2.
   1.   2.   3.
   2.   3.   4.
   3.   4.   5.
   4.   5.   6.
   5.   6.   7.
```

```
--> afficher_fenetre(s, 2)
   0.   1.   2.   3.   4.
   1.   2.   3.   4.   5.
   2.   3.   4.   5.   6.
   3.   4.   5.   6.   7.
```

1.2 - Filtre moyenne

L'application du filtre moyenne consiste à remplacer la valeur d'un élément par la moyenne arithmétique des valeurs sur une fenêtre centrée autour de cet élément.

Exercice 2 :

1. En vous inspirant de l'exercice précédent, modifiez la fonction `moyenne(s1, m)` pour qu'à partir d'un tableau `s1`, elle renvoie le tableau `s2` défini de la façon suivante : pour chaque indice `u`, `s2(u)` contient la moyenne sur les éléments de la fenêtre de taille `2m+1` autour de l'indice `u` si on peut définir cette fenêtre, et sinon `0`.

   Indication : Vous pourrez utiliser la fonction `mean` pour le calcul de la moyenne.

2. Exemple :

```
exec('filtres1D.sci')
s = [zeros(1,5), ones(1,5)*3];
disp(moyenne(s,1))
```

   doit produire

```
0.   0.   0.   0.   1.   2.   3.   3.   3.   0.
```

3. Que pensez-vous de ce résultat ?

1.3 - Moyenne pondérée

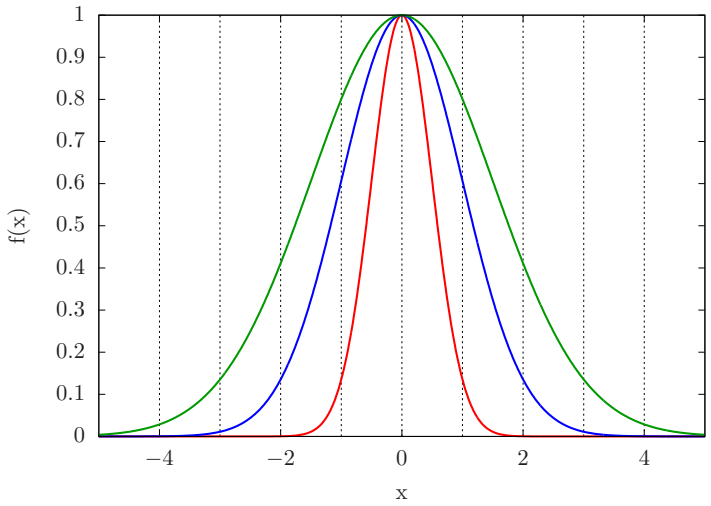
La moyenne arithmétique accorde un poids égal à chacun des pixels sur une fenêtre donnée. On considère d'autres types de moyennes, dites pondérées, où un poids différent peut être accordé à chaque pixel. Dans un cadre totalement différent, pensez à la note du baccalauréat qui est obtenue en affectant des coeficients différents selon les matières, en faisant la somme de toutes ces notes, puis en divisant par la somme des coefficients pour ramener la note sur 20. Ici, on fait de même : étant donné un tableau `W` de taille `2m+1` et un vecteur `s1`, on veut construire un vecteur `s2` tel que pour chaque indice `u`, `s2(u)` contient la moyenne pondérée par les poids `W` sur la fenêtre de taille `2m+1` autour de l'indice `u` si on peut définir cette fenêtre, et sinon `0`.

Exercice 3 :

1. Etant donné un vecteur **ligne** `W` de taille `2m+1` qu'on suppose normalisé (dont la somme des coefficients vaut 1), et un tableau `s1` contenant un signal, comment calculer en `scilab` la moyenne pondérée par les poids `W` ?
   Indications : l'opérateur `.*` réalise le produit terme à terme de deux tableaux, et la fonction `sum` renvoie la somme des éléments d'un tableau.
2. Modifiez la fonction `moyenne_W(s1, W)` dans le fichier `filtres1D.sci` pour qu'elle réalise l'opération de moyenne pondérée par les poids `W` sur les éléments du tableau `s1`, et renvoie le résultat dans un tableau `s2`.
3. Testez avec les poids `W = 1/3*[1, 1, 1]`, vous devez retrouver le même résultat qu'à l'exercice précédent.

1.4 - Filtre gaussien

La fonction gaussienne $f(x) = e^{-\frac{x^2}{2\sigma^2}}$ a une forme de chapeau qui est intéressante pour construire une moyenne pondérée avec une valeur plus importante sur l'élément central de la fenêtre et une décroissance rapide et régulière sur les autres éléments. De plus la largeur du chapeau est contrôlée par le paramètre σ, ce qui permet d'obtenir facilement une famille de filtres de taille plus ou moins grande.



Fonction gaussienne pour $\sigma = 0.5$ (rouge), 1 (bleu), 1.5 (vert).

L'idée est prendre les valeurs de $f(x)$ pour x parcourant des ensembles de la forme $\{-L, \ldots, 0, \ldots, L\}$ où L est choisi tel que $f(L+1)$ soit négligeable devant $f(0) = 1$. Par exemple pour $\sigma = 0.5$ (courbe rouge ci-dessus), on a

$$f(0) = 1, \;\; f(1) \approx 0.135, \;\; f(2) \approx 0.0003,$$

donc $f(2)$ est négligeable devant $f(0)$, on prend $L = 1$ et les poids correspondant à $\sigma = 0.5$ sont `[0.135, 1, 0.135]` avant normalisation. De manière générale, on se fixe un seuil ε et pour une valeur

de σ donnée, et on choisit la taille `L` du filtre de telle manière que $f(L) > \varepsilon$ et $f(L+1) < \varepsilon$.

Exercice 4 :

1. Montrez que $f(L) > \varepsilon \iff L < \sigma\sqrt{-2\log(\varepsilon)}$.
2. On fixe $\varepsilon = 10^{-3}$. Dans le fichier `filtres1D.sci`, modifiez la fonction `W_gauss(sigma)` pour qu'elle construise le tableau `[f(-L), ..., f(0), ..., f(L)]` où `L` est le plus grand entier vérifiant $f(L) > \varepsilon$ en fonction du paramètre `sigma`, puis qu'elle renvoie le tableau normalisé (la somme des éléments doit être égale à 1).
3. Vous pouvez tester votre fonction avec le script suivant :

   ```
   exec('filtres1D.sci');
   sigma = 0.5;
   G = W_gauss(sigma);
   ```

   Le tableau `G` doit contenir les valeurs `[0.106507 0.786986 0.106507]`. Remarquez qu'on a bien `sum(G) = 1`.
4. Remplissez le tableau suivant qui indique la taille du tableau en fonction de quelques valeurs de `sigma` :

`sigma`	0.5	1	1.5	2
taille				

Il est important que vous ayez terminé cet exercice et obtenu le résultat demandé à la question 3 car vous en aurez besoin pour le reste de cette séance et (au moins) la séance suivante.

1.5 - Lissage d'un signal bruité

On obtient un signal bruité à partir de l'image `papillon_noise.bmp`

```
exec('init_tp_image.sce');
im_noise = lire_imageBMPgris('papillon_noise.bmp');
// on extrait la 128-eme ligne de l'image
signal_noise = im_noise(128, :);

// affichage sur la figure 0
scf(0);clf(0);plot(signal_noise);

// comparaison avec le signal non bruite
im = lire_imageBMPgris('papillon.bmp');
// on extrait la 128-eme ligne de l'image
signal = im(128, :);

// affichage sur la figure 1
scf(1);clf(1);plot(signal);
```

Exercice 5 :

1. Testez l'effet du filtre moyenne sur le vecteur `signal_noise` en augmentant progressivement la taille de la fenêtre : 3, 5, 11, 21.
2. Testez l'effet du filtre Gaussien sur le vecteur `signal_noise` en augmentant progressivement la valeur de `sigma` : 0.5, 1, 3, 5.

2 - Filtrage 2D

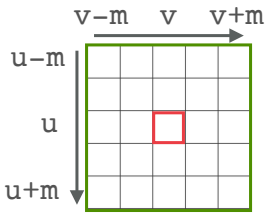
On se place maintenant dans le cadre du filtrage appliqué aux images. Le signal ici est constitué des valeurs des pixels dans un tableau à deux dimensions et les fenêtres que l'on va déplacer sur l'image sont des carrés centrés sur le pixel considéré.

2.1 - Construction de la fenêtre et parcours de l'image

Exercice 6 :

Soit `im` un tableau de taille `MxN` contenant une image.

1. Etant donné un pixel désigné par un couple d'indices `(u,v)` et un entier `m`, comment construire avec `scilab` le sous-tableau de `im` contenant les $(2m+1)^2$ éléments autour du pixel ?



2. Pour quelles valeurs de `(u,v)` est-il possible de construire ce sous-tableau ?
3. Dans le fichier `filtres2D.sci`, modifiez la fonction `afficher_fenetre2D(im, m)` pour qu'elle parcourt les éléments du tableau `im` et affiche quand c'est possible la fenêtre de taille $(2m+1)^2$ autour de l'élément courant. Vous pouvez la tester avec le code suivant :

```
// on charge les fonctions definies dans filtres2D.sci
exec('filtres2D.sci')
// definition de l'image
im = diag([1,2,3,4]);
// affichage du tableau
disp(im);
// affichage des fenetres de taille 2m+1 = 3
afficher_fenetre2D(im, 1);
```

Vous devez obtenir le résultat suivant (l'ordre des fenêtres peut varier selon votre choix de parcours) :

```
--> afficher_fenetre(im, 1 )

   1.   0.   0.
   0.   2.   0.
   0.   0.   3.

   0.   0.   0.
   2.   0.   0.
   0.   3.   0.

   0.   2.   0.
   0.   0.   3.
   0.   0.   0.

   2.   0.   0.
   0.   3.   0.
   0.   0.   4.
```

2.2 - Filtre moyenne

De même qu'en 1D, l'application de ce filtre consiste à calculer la nouvelle valeur d'un pixel comme étant la moyenne des valeurs sur une fenêtre centrée autour de lui.

Exercice 7 :

1. En vous inspirant de l'exercice précédent, modifiez la fonction `moyenne2D(im, m)` pour qu'à partir d'une image `im1`, elle renvoie l'image `im2` défini de la façon suivante : pour chaque couple d'indices `(u,v)`, `im2(u,v)` contient la moyenne de la fenêtre de taille $(2\mathtt{m}+1)^2$ autour de `im1(u,v)` si la fenêtre est définie, et sinon `0`.
2. Vous pouvez tester avec une image simple contenant un carré noir sur fond blanc :

   ```
   exec('init_tp_image.sce');
   exec('filtres2D.sci')
   // on initialise l'image avec un fond blanc
   im1 = ones(60,60)*255;
   // on construit le carre noir
   im1(25:35, 25:35) = 0;
   // on affiche l'image
   afficher_image(im1);
   // application du filtre moyenne
   im2 = moyenne2D(im1, 1);
   afficher_image(im2);
   ```

   Vous pouvez faire varier la taille de la fenêtre et observer le résultat.
3. Faites de même avec l'image `papillon.bmp` ;

   ```
   exec('init_tp_image.sce');
   exec('filtres2D.sci')
   im1 = lire_imageBMPgris('papillon.bmp');
   afficher_image(im1);
   // application du filtre moyenne
   m = 1;
   im2 = moyenne2D(im1, m);
   afficher_image(im2);
   ```

   Que pensez-vous du résultat pour `m=5` ?

2.3 - Moyenne pondérée

On étend en 2D la moyenne pondérée définie dans la section 1.3.

Exercice 8 :

1. Etant donné une matrice `W` de taille $(2\mathtt{m}+1)^2$ qu'on suppose normalisée, et une image `im1` contenant un signal, comment calculer en `scilab` la moyenne pondérée par les poids `W` ?
   Indications : l'opérateur `.*` réalise le produit terme à terme de deux tableaux, et la fonction `sum` renvoie la somme des éléments d'un tableau.
2. Modifiez la fonction `moyenne2D_W(im1, W)` dans le fichier `filtres2D.sci` pour qu'elle réalise l'opération de moyenne pondérée par les poids `W` sur les éléments de l'image `im1` et renvoie le résultat dans une image `im2`.
3. Testez avec les poids `W = 1/9*ones(3,3)`, vous devez retrouver le même résultat qu'à l'exercice précédent.

2.4 - Filtre gaussien

On peut étendre à deux dimensions la fonction gaussienne vue auparavant : $f_{2D}(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$. Elle possède les mêmes bonnes propriétés pour obtenir des poids : valeur plus importante sur l'élément central de la fenêtre et décroissance régulière ; de plus elle ne possède pas de direction privilégiée. Par ailleurs, on peut observer que $f_{2D}(x,y) = e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} = f(x)f(y)$, ce qui implique qu'on peut construire le tableau `G2D` correspondant à f_{2D} à partir du tableau `G` construit dans l'exercice 9 :

$$\forall(\mathtt{i},\mathtt{j}), \mathtt{G2D(i,j)d} = \mathtt{G(i)} * \mathtt{G(j)}$$

Exercice 9 :

1. Dans le fichier `filtres2D.sci`, modifiez la fonction `W_gauss_2D(sigma)` pour qu'elle renvoie le tableau `G2D` décrit ci-dessus.
2. Vous pouvez tester votre fonction avec le script suivant :

   ```
   exec('filtres2D.sci');
   sigma = 0.5;
   G2D = W_gauss_2D(sigma);
   ```

   Le tableau `G2D` doit contenir les valeurs

   ```
   0.0113437   0.0838195   0.0113437
   0.0838195   0.619347    0.0838195
   0.0113437   0.0838195   0.0113437
   ```

   Remarquez qu'on a bien `sum(G2D) = 1`.
3. Vous pouvez maintenant utiliser la fonction de l'exercice précédent pour appliquer le filtre gaussien :

   ```
   exec('init_tp_image.sce');
   exec('filtres2D.sci')
   im1 = ones(60,60)*255;
   im1(25:35, 25:35) = 0;
   afficher_image(im1);
   // application du filtre gaussien
   sigma = 0.5;
   im2 = moyenne2D_W(im1, W_gauss_2D(sigma));
   afficher_image(im2);
   ```

   Vous pouvez faire varier `sigma`.
4. Faites de même avec l'image `papillon.bmp` ; que pensez-vous du résultat pour `sigma = 2.5` en comparaison avec le résultat avec un filtre moyenne de grande taille (question 3 de l'exercice 7) ?

2.5 - Lissage d'une image bruitée

On reprend l'image bruitée `papillon_noise.bmp`.

Exercice 10 :

- Appliquez le filtre moyenne à l'image bruitée pour plusieurs tailles de fenêtres.
- Faites de même avec le filtre gaussien pour plusieurs valeurs de `sigma`.
- Avec quel filtre obtient-on le meilleur résultat ?