Filtrage numérique par convolution

1. Introduction

Un signal numérique est une suite de nombres, obtenue par exemple par échantillonnage d'un signal analogique. Une fréquence d'échantillonnage est généralement associée au signal numérique.

Le filtrage numérique est un traitement effectué sur cette suite de nombres, pour transformer le signal. Ce calcul peut être fait sur un ordinateur à partir de signaux stockés en mémoire. Dans les circuits d'électronique numérique, il est fait en *temps réel* par des microprocesseurs spécialisés (DSP Digital Signal Processor), que l'on rencontre par exemple dans les téléphones portables pour le traitement du son.

Nous allons aborder ici un type particulier de filtrage, le *filtrage par convolution*. Les filtres obtenus sont à réponse impulsionnelle finie (filtre RIF).

Les images numérisées sont aussi des signaux numériques, mais à deux dimensions. Une image numérisée est donc une matrice de nombres. Le filtrage par convolution se pratique couramment sur les images numériques, par exemple pour rendre les images légèrement floues afin de réduire le bruit, ou bien au contraire pour accentuer les détails. Nous verrons comment se fait le filtrage par convolution sur une image.

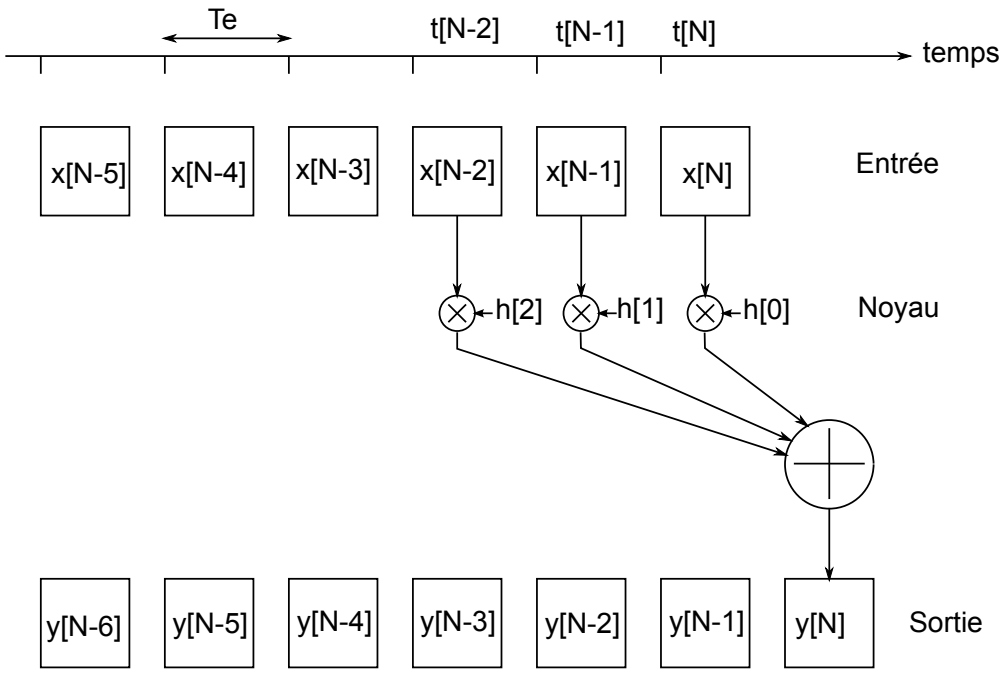
2. Filtrage des signaux

2.a. Convolution

On considère un signal numérique donné par la suite x_n, où n est un indice, compris en 0 et $N-1$.

Lorsque le signal est stocké en mémoire, le nombre d'échantillons N est fixé dès le départ. Dans un système de traitement numérique *temps réel*, un nouvel échantillon doit être traité à intervalle de temps régulier (la période d'échantillonnage). Dans ce cas, le nombre d'échantillons N augmente au cours du temps.

Pour comprendre le filtrage par convolution, nous commençons par le cas du filtrage temps réel. La figure suivante en montre le principe. Les indices sont notés entre crochets.



Seuls les 6 derniers échantillons sont représentés. La convolution se fait avec un noyau, qui dans cet exemple comporte trois termes $h[0], h[1], h[2]$. Le signal filtré est noté y_n. Pour obtenir le dernier échantillon du signal filtré, on effectue une combinaison linéaire des trois derniers échantillons du signal :

$$y_N = h_0 x_N + h_1 x_{N-1} + h_2 x_{N-2} \tag{1}$$

Dans un système temps réel, le dernier échantillon de la sortie est décalé d'une durée égale à la période d'échantillonnage par rapport au dernier échantillon de l'entrée. En effet, le calcul de cet échantillon prend un certain temps (qui doit être inférieur à la période). Sur cet exemple, le nombre d'opérations à effectuer est très réduit (3 multiplications et la somme de trois nombres), mais les cadences d'échantillonnage peuvent être très élevées, de l'ordre de plusieurs gigahertz.

Plus généralement, si M est le nombre de termes du noyau, l'échantillon n de la sortie se calcule en faisant une combinaison linéaire des M derniers échantillons de l'entrée :

$$y_n = \sum_{i=0}^{M-1} h_i x_{n-i} \tag{2}$$

Cette relation définit la *convolution discrète*, et permet donc de réaliser un filtrage par convolution. L'effet du filtre dépend évidemment des coefficients choisis pour le noyau.

2.b. Réponse impulsionnelle

Une impulsion est un signal numérique défini par :

$$x_0 = 1 \tag{3}$$

$$x_n = 0 \ (n \neq 0) \tag{4}$$

La sortie est alors :

$$y_n = 0 \ (n < 0) \tag{5}$$
$$y_0 = h_0 x_0 = h_0 \tag{6}$$
$$y_1 = h_1 x_0 = h_1 \tag{7}$$
$$\dots \tag{8}$$
$$y_{M-1} = h_{M-1} x_0 = h_{M-1} \tag{9}$$
$$y_n = 0 \ (n \geq M) \tag{10}$$

Les coefficients $h_0, h_1 ... h_{N-1}$ (le noyau du filtre) constituent donc la *réponse impulsionnelle* du filtre.

Un filtre fonctionnant par convolution est dit à *réponse impulsionnelle finie* (RIF) car le nombre de termes de la réponse impulsionnelle est fini.

Les filtres numériques RIF sont toujours stables : si l'entrée reste bornée, la sortie l'est aussi.

2.c. Exemples

Filtre moyenneur

Le filtre moyenneur effectue la moyenne arithmétique de M échantillons consécutifs du signal d'entrée. Par exemple, pour $M = 3$, le noyau est :

$$h = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \tag{11}$$

Le filtre moyenneur est un filtre passe-bas, qui réduit les hautes fréquences.

Filtre passe-bas en sinus cardinal

Le filtre moyenneur a l'inconvénient d'être très peu sélectif : si l'on augmente la taille du noyau, on réduit encore plus les hautes fréquences mais on altère aussi les basses fréquences.

Le filtre passe-bas en sinus cardinal peut être très sélectif (bien plus que les filtres électroniques analogiques). Pour le définir, il faut tout d'abord calculer le rapport de la fréquence de coupure sur la fréquence d'échantillonnage (qui doit être inférieur à 1/2) :

$$a = \frac{f_c}{f_e} \tag{12}$$

On appelle filtre idéal un filtre qui laisse passer sans atténuation toutes les fréquences inférieures à la fréquence de coupure, et qui annule les fréquences supérieures à la coupure. Malheureusement, la réponse impulsionnelle d'un filtre passe-bas idéal est infinie. Elle est donnée par :

$$g(k) = 2a \frac{\sin(2\pi\, ka)}{2\pi\, ka} \tag{13}$$

Cette réponse impulsionnelle est rendue finie par troncature. Soit P l'indice de troncature. La réponse impulsionnelle finie comporte alors $M = 2P + 1$ termes. Elle est définie par :

$$h_k = g(k - P) \tag{14}$$

l'indice k variant de 0 à $2P$.

L'indice de troncature devra être d'autant plus grand que a est faible. Il faut donc éviter une fréquence d'échantillonnage trop grande par rapport à la fréquence de coupure (le sur-échantillonnage n'est pas toujours souhaitable en filtrage numérique). Un critère simple est $Pa > 1$. Pour un coefficient a donné, la sélectivité du filtre augmente avec P. À la limite $P \to \infty$, on obtient un filtre idéal.

L'inconvénient d'un indice P grand est la perte des $M = 2P+1$ échantillons au début du signal.

Filtre dérivateur

La dérivation d'un signal numérique repose sur la formule de Taylor :

$$f(t+\epsilon) = f(t) + \epsilon f'(t) + O(\epsilon^2) \tag{15}$$

d'où l'on déduit :

$$f'(t) = \frac{f(t+\epsilon) - f(t)}{\epsilon} + O(\epsilon) \tag{16}$$

En terme de signal numérique, cela donne :

$$y_n = \frac{x_n - x_{n-1}}{T_e} \tag{17}$$

où T_e est la période d'échantillonnage. Cette évaluation de la dérivée est appelée *différence finie*. La réponse impulsionnelle est donc (on omet la période d'échantillonnage) :

$$h = (1, -1) \tag{18}$$

Il y une autre manière d'évaluer la dérivée. En considérant le développement suivant :

$$f(t-\epsilon) = f(t) - \epsilon f'(t) + O(\epsilon^2) \tag{19}$$

La différence avec le premier donne :

$$f'(t) = \frac{f(t+\epsilon) - f(t-\epsilon)}{2\epsilon} + O(\epsilon) \tag{20}$$

ce qui donne la différence finie suivante :

$$y_n = \frac{x_n - x_{n-2}}{2T_e} \tag{21}$$

dont le noyau est :

$$h = (1, 0, -1) \tag{22}$$

Dans ce cas, on considère généralement que l'échantillon de sortie y_n est simultané avec l'échantillon x_{n-1}. Cette différence finie est appelée *différence finie centrée*.

3. Filtrage des images

3.a. Convolution

Les images numériques sont des signaux numériques à deux dimensions, qui s'apparentent aux signaux de taille fixe discutés plus haut. Les échantillons sont appelés les pixels.

Le filtrage par convolution se fait sur une image en niveaux de gris. Les images en couleur sont composées de trois couches, par exemple rouge, vert et bleu. Chaque couche peut être vue comme une image en niveau de gris. Il faudra donc opérer sur chaque couche séparément pour filtrer une image en couleur (pas nécessairement avec le même filtre).

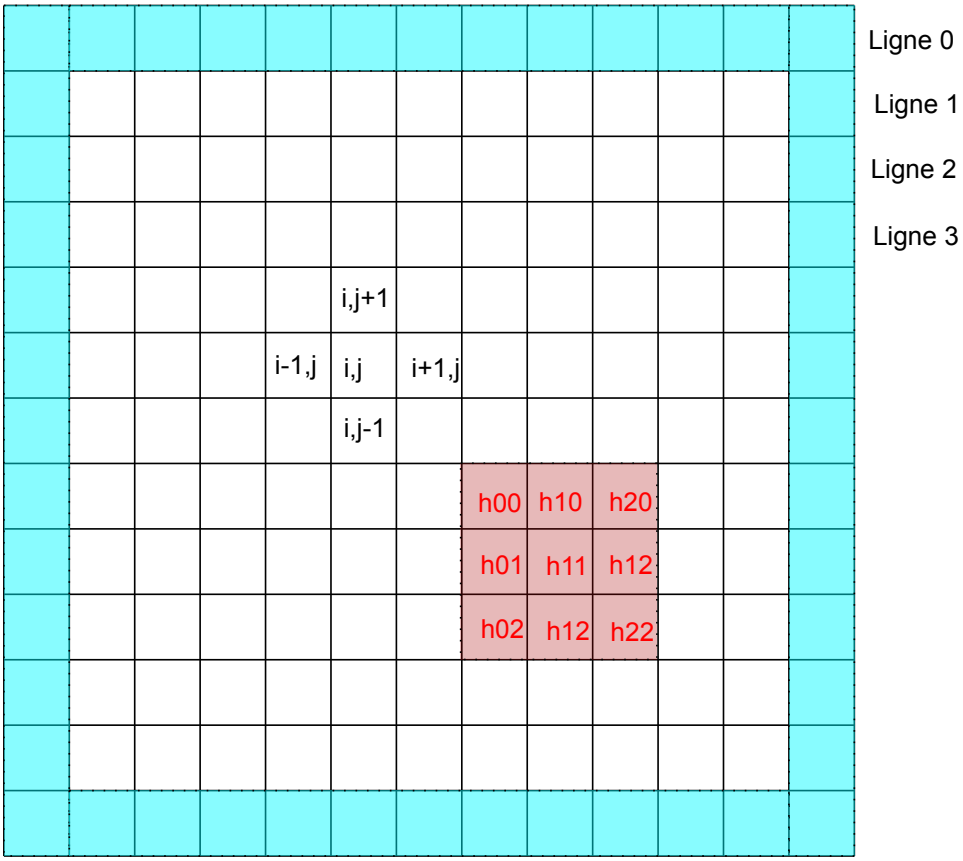
On considère une image numérique en niveaux de gris, dont la valeur du pixel (i, j) est notée $X_{i,j}$. On adopte la convention qui consiste à placer l'indice de colonne (indice i) en premier, car cet indice correspond à l'axe x sur l'image. La réponse impulsionnelle (ou noyau) dans le cas d'un signal numérique à deux dimensions est elle-même un signal à deux dimensions, c'est-à-dire une matrice, dont les coefficients seront notés $h_{m,n}$. Pour simplifier, on se limite au cas des réponses impulsionnelles carrées, avec un nombre de termes impair sur chaque dimension $M = 2P + 1$. Par exemple, pour $P = 1$:

$$H = \begin{pmatrix} h_{00} & h_{10} & h_{20} \\ h_{01} & h_{11} & h_{21} \\ h_{02} & h_{12} & h_{22} \end{pmatrix} \tag{23}$$

L'image filtrée Y s'obtient en effectuant le produit de convolution entre X et H, ce qui donne pour un pixel (i, j) de l'image finale :

$$\begin{array}{rlll} Y_{i,j} = & h_{00}X_{i-1,j-1} & +h_{10}X_{i,j-1} & +h_{20}X_{i+1,j-1} \\ & +h_{01}X_{i-1,j} & +h_{11}X_{i,j} & +h_{21}X_{i+1,j} \\ & +h_{01}X_{i-1,j+1} & +h_{12}X_{i,j+1} & +h_{22}X_{i+1,j+1} \end{array} \tag{24}$$

Le pixel $Y_{i,j}$ est donc obtenu en faisant une combinaison linéaire (ou moyenne pondérée) du pixel $X_{i,j}$ de l'image initiale et de ses 8 proches voisins. La relation ci-dessus se généralise aux noyaux de convolution $5x5$, $7x7$, etc.



Remarque : il faudra faire attention au fait que les images sont stockées dans des tableaux, dont le premier indice se réfère aux lignes horizontales. Le premier indice est donc l'indice j. On remarque aussi que les lignes des images sont numérotées en partant du haut. Ainsi la ligne 0 se trouve en haut de l'image. La correspondance entre les indices définis ci-dessus et ceux du tableau est donc

$$X_{i,j} \rightarrow image[N_y - 1 - j][i] \tag{25}$$

où N_y est le nombre de lignes de l'image.

On voit que la convolution ne peut s'appliquer sur les bords de l'image, plus précisément sur les P rangées horizontales et verticales des bords (coloriées en bleu sur la figure). On évite généralement de réduire la taille des images lors du filtrage. Il faut donc prévoir un traitement spécial pour ces rangées. La solution la plus simple consiste à ne pas modifier ces pixels.

Bien que les valeurs des pixels soient souvent des entiers (par exemple codés sur 8 bits), les calculs de filtrage sont faits avec des nombres à virgule flottante. Cela évite des erreurs d'arrondi grossières lorsqu'on applique plusieurs filtrages successifs, ou d'autres types de traitement. Les valeurs de l'image finale sont converties en entier uniquement au moment de l'affichage, ou du stockage dans un fichier.

3.b. Exemples

Filtre moyenneur

Voici la réponse impulsionnelle d'un filtre moyenneur $3x3$:

$$H = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} \tag{26}$$

Comme dans le cas des signaux à une dimension, le filtre moyenneur est un filtre passe-bas, qui atténue les fins détails de l'image, ce qui peut être utile pour réduire le bruit.

Ce filtre ne modifie pas la plage des valeurs des niveaux de gris. Par exemple pour une image codée en 8 bits (par couche), les niveaux sont compris entre 0 et 255, et restent dans cette plage après application du filtre.

Filtres dérivateurs

La dérivation est une opération courante en traitement d'image, car elle permet de faire ressortir les bords des formes. La dérivation par rapport à la direction horizontale (X) est :

$$H_x = (-1, 0, 1) \tag{27}$$

Il s'agit d'une différence finie centrée car la dérivée en un pixel est évaluée en utilisant le pixel situé à sa droite et le pixel à gauche.

La dérivation par rapport à la direction verticale (Y) est :

$$H_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \tag{28}$$

La dérivation a l'inconvénient d'amplifier le bruit. Pour réduire cet effet, on associe un filtrage passe-bas à la dérivation. C'est ce que fait le filtre de Sobel, par exemple pour la direction X :

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{29}$$

L'effet de filtrage passe-bas est obtenu ici par un moyennage pondéré dans la direction Y.

Pour en savoir plus sur les filtres dérivateurs et l'application à la détection de bords, voir Détection des bords.

Pour voir l'effet de plusieurs filtres sur des images, voir cette simulation.

Les filtres dérivateurs ne conservent pas l'intervalle $[0, 255]$ des images 8 bits. En effet, certaines dérivées peuvent être négatives. Il faut en tenir compte lorsqu'on veut afficher l'image de sortie. La fonction `matplotlib.pyplot.imshow` convertit les valeurs de manière à les faire entrer dans la plage $[0, 255]$. Une autre solution consiste à prendre la valeur absolue pour afficher la dérivée. C'est ce qui est fait dans cette simulation.