UPMC

Master SESI

PERI

Filtrage

Introduction

Le but de ce TP est de d'implémenter différents filtres en flottant et en virgule fixe pour ensuite évaluer la qualité du filtrage. Les codes source et cet énoncé sont disponible sur le trac du LIP6 : `https://www-soc.lip6.fr/trac/sesi-peri`. Un makefile basique permet de compiler l'ensemble des fichiers avec gcc.
Les fonctions principales se trouvent dans le fichier `test_filterNR.c`. Les filtres à coder se trouve dans le fichier `filterNR.c`. Les différentes fonctions de calcul sont paramétrables pour laisser à chacun la possibilités de tester un grand nombre de cas. Par défaut, les paramètres sont réglés à des valeurs représentatives de la *réalité*.

1 Signal de base et bruit

1.1 Signal de base

Plusieurs formes de signaux de taille n (paramètre) sont disponibles (fichier `sampleNR.c`) :

- constant (`constant`) signal constant de valeur $m = 100$
- front montant (`step_up`) d'amplitude a, passant de $m - a$ à $m + a$, la transition se faisant au milieu du signal en $n/2$
- front descendant (`step_down`) d'amplitude a, passant de $m - a$ à $m + a$, la transition se faisant au milieu du signal en $n/2$
- échelon (`step_updown`) d'amplitude a, passant de $m - a$ à $m + a$, la première transition se faisant au premier quart en $n/4$ et la seconde au troisième quart en $3n/4$

Les signaux seront composés d'échantillons codés sur 8 bits (`uint8`) stockés dans des tableaux dynamiques (`ui8vector`).

1.2 bruit

Plusieurs bruits sont disponibles (fichier `noise.c`), mais nous n'utiliserons que le bruit blanc gaussien (bruit additionnel)

- bruit gaussien `gaussian_noise_ui8vector`, d'écart type `sigma`
- bruit impusionnel `impulse_noise_ui8vector` avec une probabilité `percent`

Ce même fichier contient les fonctions d'analyse qualitative `mean_square_error_ui8vector` et `psnr_ui8vector`.
L'erreur quadratique moyenne :

$$mse(x, \tilde{x}) = \frac{1}{n^2} \sum (x - \tilde{x})^2 \tag{1}$$

Le PSNR :

$$psnr(x, \tilde{x}) = 10 \times log_{10} \frac{max(x)^2}{mse(x, \tilde{x})} \tag{2}$$

2 Filtrage

2.1 Filtrage non récursif

Le filtre non récursif fourni est un filtre moyenneur codé en flottant (`float32`) `fir_average_f32`. Son paramètre est le `radius` r :

$$y(n) = \frac{1}{2r+1} \sum_{i=-r}^{i=+r} x(n+i) \tag{3}$$

En vous inspirant du code `fir_average_f32`

1. Codez la version `fir_average_i16` où les calculs seront fait dans des variables de types `uint16` et où la division sera réalisée via une division entière. Que faut il ne pas oublier de faire ?
2. Codez la version `fir_average_q16` où les calculs seront fait dans des variables de types `uint16` et où la division sera remplacée par le calcul d'une fraction équivalente. Le paramètre supplémentaire de cette fonction est q le nombre de bits de codage. Observez les résultats pour $q = 8$ et $q = 10$.
3. La fonction implémentant le filtre gaussaient étant déjà codée : `fir_gauss_f32`, comparez les performances de ces 4 filtres : faites un tableau indiquant, en fonction du niveau de bruit (`sigma_noise`) et de la largeur du filtre (pour les filtre moyennes) et de q (pour le filtre en virgule fixe), le $psnr$ résultat.

2.2 Filtrage récursif

Le filtre récursif fourni est le filtre de FGL codée en flottant `iir_f32`. Son paramètre est α et ses coefficients sont b_0, a_1 et a_2 :

$$b_0 = (1-\gamma)^2, \quad a_1 = 2\gamma, \quad a_2 = -\gamma^2, \quad \text{avec } \gamma = e^{-\alpha} \tag{4}$$

Le filtre a pour formule :

$$y(n) = b_0 x(n) + a_1 y(n-1) + a_2 y(n-2) \tag{5}$$

1. Codez la version `iir_q16` où les coefficient sont multipliés par 2^q et les calculs se font dans des variables de type `sint16`. Si on pose $Q = 2^q$, $B_0 = b_0 \times Q$, $A_1 = a_1 \times Q$, $A_2 = a_2 \times Q$, et si on utilise des variables temporaires pour stocker les différentes valeurs de $y(n)$: $x_0 = x(n)$, $y_0 = y(n)$, $y_1 = y(n-1)$, $y_2 = y(n-2)$ alors l'équation du filtre devient :

$$y_0 = B_0 x_0 + A_1 y_1 + A_2 y_2 \tag{6}$$

2. Afin d'augmenter la précision des calculs, on décide d'augmenter le nombre de bits pour coder les échantillons des x et y. Pour cela on pose $X_0 = x_0 \times Q$, $Y_0 = y_0 \times Q$, $Y_1 = y_1 \times Q$, $Y_2 = y_2 \times Q$. Codez la version `iir_q32` de ce filtre. Pour debugger et mettre au point le code, il est conseillé d'observer le contenu des variables et d'afficher leur valeurs à chaque tour de boucle. Le calcul devient :

$$Y_0 = B_0 X_0 + A_1 Y_1 + A_2 Y_2 \tag{7}$$

3. Comparer les trois implémentations du filtre récursif à celles des filtres non récursifs. Conclure sur l'utilisation de la virgule fixe.