

Filtrage 1D

1 Introduction

Les fichiers C nécessaires se trouvent à l'adresse www.ief.u-psud.fr/~lacas/Teaching.
Des documents sont téléchargeables à l'adresse www.ief.u-psud.fr/~lacas/Download.

2 Filtres linéaires

2.1 Filtres de taille 3

Les filtres moyenneurs (*average* en anglais) sont les plus simples des filtres passe-bas. Soit le filtre A_3 :

$$A_3 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad (1)$$

L'équation aux différences de ce filtre est :

$$y_3(n) = \frac{1}{3} [x(n-1) + x(n) + x(n+1)] \quad (2)$$

1. Codez le filtre A_3 dans la fonction `avg3_vf32vector`.
2. Validez cette fonction via la fonction de test `test_avg3_vf32vector`

3 Filtres non linéaires

3.1 Filtres *min*

Les filtres *min* sont des filtres à fenêtre glissante, comme les filtres linéaires.

1. Afin de simplifier et de rendre plus lisible les fonctions codées, écrivez les macros suivantes (voir fichier `filtre1_SSE.h` pour plus de détails) `vec_left1_ps`, `vec_left2_ps`, `vec_left3_ps`, `vec_left4_ps` pour le calcul de registres SIMD non alignés à gauche et les macros `vec_right1_ps`, `vec_right2_ps`, `vec_right3_ps`, `vec_right4_ps` pour le calcul de registres SIMD non alignés à droite.
2. Codez le filtre *min-3* dans la fonction `min3_vf32vector`. Ecrivez la macro `vMIN3`.
3. Codez le filtre *min-5* dans la fonction `min5_vf32vector`. Ecrivez la macro `vMIN5`.
4. Validez ces fonction via la fonction de test `test_min_vf32vector`

3.2 Filtres *médian*

3.2.1 médian 9 classique via algorithme de tri

La valeur médiane d'une série de n valeurs est la valeur de rang $n/2$ lors que toutes ces valeurs sont triées. Le filtrage médian est un filtre non linéaire de la famille des filtres d'ordre. Le filtrage médian est très robuste à différents types de bruit, comme le bruit gaussien ou le bruit impulsionnel. Son principal inconvénient est sa complexité mathématique lié à l'utilisation inefficace d'un algorithme de tri. L'algorithme classique du filtre médian repose sur l'utilisation d'un algorithme de tri (tri par sélection, tri par insertion, tri à bulles, tri rapide, ...).

Dans le cas où le nombre de valeurs à trier est faible ce sont les tris par sélection ou par insertion qui sont le plus efficaces. Ces tris ne sont pas vectorisables (ou tout du moins, leur version SIMD n'est pas plus rapide que la version scalaire). Le seul tri simplement vectorisable est le tri à bulles qui consiste à faire des comparaisons de paire de nombres. La figure 1 montre le fonctionnement du tri à bulles pour trier 3 valeurs.

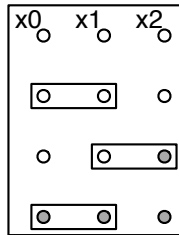


FIG. 1 – M_3 : tri de 3 valeurs par tri à bulles

1. Codez la macro `vMM` (pour `vector-min/max`) qui pour tout couple de valeurs (a, b) renvoie un couple de valeurs (m, M) telque $m = \min(a, b)$ et $M = \max(a, b)$ (en fait, après l'appel de `vMM(a, b)`, a sera égal à m et b sera égal à M).
2. Codez la fonction `median3_vf32vector`. Codez la macro `vTRI3` qui utilisera la macro `vMM`.
3. En utilisant le même principe, codez la fonction `median5_vf32vector` et la macro `vTRI5` (qui utilisera aussi la macro `vMM`).
4. afin de faciliter l'écriture des fonctions `median7_vf32vector` et `median8_vf32vector` faire un programme généraliste (fonction du paramètre n la taille du filtre) générant le code des macros de tri.

3.2.2 Médian rapide avec tri partiel

1. Codez un algorithme de médian 9 rapide, telque vu en cours et basé sur l'utilisation de la macro `vTRI3`. Codez une macro `vMEDIAN9`.
2. Comparez le nombre d'instructions utilisées pour le médian 9 classique et pour le médian 9 rapide.