TREIZIÈME COLLOQUE GRETSI - JUAN-LES-PINS DU 16 AU 20 SEPTEMBRE 1991

TRAITEMENTS SEPARABLES POUR CALCULER LE FILTRE MEDIAN SUR LE PROCESSEUR LIGNE SYMPATI2

Pascal FERNANDEZ, Jean-Luc BASILLE

IRIT : 118 route de Narbonne 31062 TOULOUSE CEDEX
Tél: 61-55-63-08 Fax: 61-55-62-58

RÉSUMÉ

Resumé : Pour réduire le temps de calcul du filtre Médian sur le Processeur Ligne SYMPATI2 de type SIMD [JUVI88] nous proposons deux approches :
- La première est une méthode de calcul séparable qui produit la valeur de la vraie médiane de la fenêtre (2n+1)(2n+1),
- La seconde est une méthode de calcul séparable qui donne une valeur parmi les 2n+1 valeurs médianes des (2n+1)(2n+1) points de la fenêtre considérée.

ABSTRACT

Abstract : In order to reduce the computing time of the median filter on the Line Processor SYMPATI2 we suggest two approaches :
- First , a separable processing method which gives the median value of the (2n+1)(2n+1) pixels of the considered window.
- Secondly, a separable processing method which gives a value among the 2n+1 median values of the (2n+1)(2n+1) pixels of the considered window.

1. INTRODUCTION

Le but d'un filtrage est d'améliorer "la qualité de l'image", pour obtenir une image plus facile à traiter. Un grand nombre de méthodes issues de modèles mathématiques existent, elles mettent en oeuvre différents filtres sur des critères objectifs. Dans la réalité, c'est l'observateur humain qui juge subjectivement avec son oeil, c'est pourquoi, d'autres méthodes plus empiriques ont vu le jour, particulièrement le filtre médian de Tukey qui consiste à remplacer le point courant par la valeur médiane des points du voisinage considéré. Si nous travaillons sur une fenêtre 3x3, le traitement consiste à chercher la valeur médiane de 9 points. Si ce filtrage donne de bons résultats sur des images ayant un bruit impulsionnel, il est coûteux en temps d'exécution en raison du nombre élevé de comparaisons qu'il nécessite.

2. LE FILTRE MEDIAN DE TUKEY

Pour réduire le temps de traitement, une solution a été proposée, elle opère par propagation d'histogrammes [HUAN79]. Cet histogramme est mis à jour lorsqu'on déplace la fenêtre traitée. Cette fenêtre possède un nombre impair de pixels c'est-à-dire une fenêtre de taille (2n+1)x(2n+1). Cette méthode utilise le calcul de la valeur médiane d'une fenêtre pour déterminer la valeur médiane de la fenêtre suivante. Après avoir comptabilisé le nombre de pixels inférieurs à la médiane *nbinfmed*, les différentes étapes de l'algorithme sont :
- Déplacement de la fenêtre vers la droite d'une colonne : on met l'histogramme à jour et *nbinfmed* qui contient le nombre de pixels de la fenêtre courante qui ont une valeur inférieure à la médiane de la fenêtre précédente.
- L'histogramme permet d'obtenir la valeur médiane, en partant du résultat des valeurs du médian précédent. Respectivement on avance ou on recule dans l'histogramme en fonction de *nbinfmed* inférieur ou égal, ou strictement supérieur à 2n(n+1).

Les résultats des pixels de la fenêtre de filtrage NxN sont obtenus à partir de la fenêtre précédente en supprimant la colonne la plus à gauche, et en rajoutant une colonne à droite dans le cas d'un déplacement gauche-droit. Il y a donc recouvrement de la fenêtre de filtrage.
Cette méthode est intéressante s'il y a une corrélation locale des pixels. En effet si les valeurs sont peu différentes le nombre d'itérations sur l'histogramme est réduit. Cette solution n'est pas satisfaisante sur le Processeur Ligne car pour des fenêtres de petites tailles, il faut traiter un nombre important de valeurs (autant que de niveaux de gris). Pour réduire le temps de calcul nous avons envisagé deux approches que nous allons détailler.

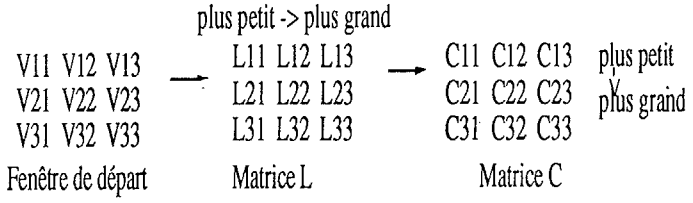
3. TRAITEMENT SEPARABLE POUR LE FILTRE MEDIAN.

Il est intéressant pour pouvoir séparer le traitement d'effectuer une décomposition ligne/colonne. Nous considérons une fenêtre 3x3. A partir de celle-ci nous trions les valeurs de chaque ligne de cette fenêtre. Nous obtenons la matrice L telle que :

$$Li,1 \leq Li,2 \leq Li,3 \quad \text{avec } i = 1,2,3$$

Nous trions alors les valeurs de chaque colonne de cette matrice L et nous obtenons la matrice C telle que :

$$C1,j \leq C2,j \leq C3,j \quad \text{avec } j = 1,2,3$$



Tout d'abord nous pouvons montrer que la matrice C 3x3 triée sur colonnes conserve le tri sur les lignes. Ce résultat provient du Lemme suivant :

Lemme

Si nous avons les p inégalités suivantes :

$$A_{p,k} \leq B_{p,k} \quad \text{avec } k = 1,p$$

et si nous ordonnons séparément les deux listes {Ap,k} et {Bp,k} nous obtenons les listes $\{A'_{p,k}\}$ et $\{B'_{p,k}\}$ telles que

$$A'_{p,k} \leq A'_{p,k+1}$$

$$\text{et} \quad B'_{p,k} \leq B'_{p,k+1} \quad \text{avec } k=1,p-1$$

Alors nous pouvons écrire A'p,k ≤ B'p,k avec k = 1,p

Démonstration

Nous pouvons démontrer ce Lemme par récurrence. En effet nous avons de façon évidente

$$A'_{p,1} = \underset{k}{\text{Min}}(A_{p,k}) \leq \underset{k}{\text{Min}}(B_{p,k}) = B'_{p,1}$$

montrons que $A_{p,k1} \leq B_{p,k0}$ avec k0 tel que $A_{p,k0} = A'_{p,1}$ et k1 tel que $B_{p,k1} = B'_{p,1}$

En effet $A_{p,k1} \leq B_{p,k1}$

or $B_{p,k1} \leq B_{p,k0}$ donc $A_{p,k1} \leq B_{p,k0}$

Nous avons donc alors (p-1) inégalités $A_{p-1,l} \leq B_{p-1,l}$ avec l = 1,p-1

où les ensembles ($A_{p-1,l}$) et ($B_{p-1,l}$) sont définis de la manière suivante, avec h0 = min(k0,k1) et h1 = max(k0,k1) :

- $A_{p-1,l} = A_{p,l}$
  $B_{p-1,l} = B_{p,l}$ pour l = 1,h0-1
- $A_{p-1,h0} = A_{p,k1}$
  $B_{p-1,h0} = Bp,k0$
- $A_{p-1,l} = A_{p,l}$
  $B_{p-1,l} = Bp,l$ pour l = h0+1, h1-1
- $A_{p-1,l} = A_{p,l+1}$
  $B_{p-1,l} = B_{p,l+1}$ pour l = h1,p-1

Ainsi nous pouvons chercher le minimum des ($A_{p-1,l}$) et celui des ($B_{p-1,l}$), l = 1,p-1 et continuer jusqu'à avoir une seule inégalité, ce qui montre le Lemme.
Ainsi en appliquant ce Lemme successivement sur les ensembles {Li,1} et {Li,2} sur {Li,2} et {Li,3} avec i =1,2,3, nous montrons que la fenêtre C reste triée sur les lignes c'est-à-dire :

C11 ≤ C12 ≤ C13
C21 ≤ C22 ≤ C23
C31 ≤ C32 ≤ C33

D'aprés ce que nous venons de démontrer, nous obtenons les inégalités suivantes :

C11 ≤ C12 ≤ C13 (1),
C21 ≤ C22 ≤ C23 (2),
C31 ≤ C32 ≤ C33 (3),

résultent du tri sur les lignes de la fenêtre,

C11 ≤ C21 ≤ C31 (4),
C12 ≤ C22 ≤ C32 (5),
C13 ≤ C32 ≤ C33 (6),
résultent du tri sur les colonnes de la fenêtre.

La médiane des 9 valeurs est alors la médiane de la deuxième diagonale (C31,C22,C13).

En effet nous voyons que C11, C12 et C21 ont chacun plus de 5 valeurs qui leur sont respectivement supérieures :

pour C11 : C12, C13, C21, C22, C32, C31, C32, C33
pour C12 : C13, C22, C32, C32, C33
pour C21 : C22, C32, C31, C32, C33
donc la médiane ne peut être une de ces trois valeurs.

De même nous voyons que C23, C32 et C33 ont chacun plus de 5 valeurs qui leur sont respectivement inférieures. Il reste donc 3 valeurs possibles : C31, C22, C13 soit la deuxième diagonale de la matrice C. La médiane de la fenêtre 3x3 initiale est donc la médiane de ces 3 valeurs.
Pour chercher la valeur médiane nous avons utilisé la méthode du tri par bulles qui est performante à cause d'une faible détérioration du taux de parallélisme sur une machine de type SIMD. Sans appliquer le Lemme, 30 comparaisons sont nécessaires pour calculer la valeur médiane de la fenêtre 3x3. En appliquant le Lemme, le nombre de comparaisons total lors du traitement de la ligne i est de 15, soit 3 comparaisons pour trier la ligne i+1, les lignes i et i-1 ayant déja été triées, 9 comparaisons pour trier les 3 colonnes de la fenêtre et 3 comparaisons pour trier la deuxième diagonale (figure 1).

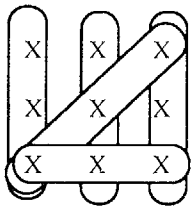


Figure 1 : Les points à trier pour une fenêtre 3x3.

3.1 Mise en oeuvre

Pour simplifier l'écriture nous avons pris 3 images de travail : l'*image 1* contient tous les plus petits points de la ligne de la fenêtre 3x3, l'*image 2* tous les points médians et l'*image 3* tous les plus grands points (figure 2).

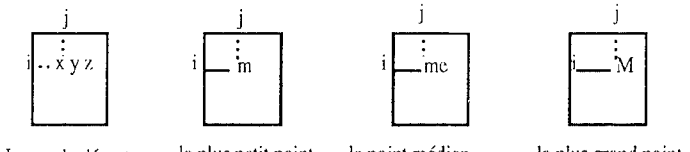


Image de départ | le plus petit point Image 1 | le point médian Image 2 | le plus grand point Image 3

m = min(x,y,z) | me = Med(x,y,z) | M = Max(x,y,z)

Figure 2 : La première phase du calcul de la valeur médiane

- Lors d'une première passe sur toute l'image de départ nous trions la ligne courante en écrivant dans les 3 images de travail.
- Durant la seconde, nous balayons les images intermédiaires, pour récupérer la fenêtre 3x3 triée sur les lignes que nous rangeons dans les 9 registres de la manière suivante (figure 3) :

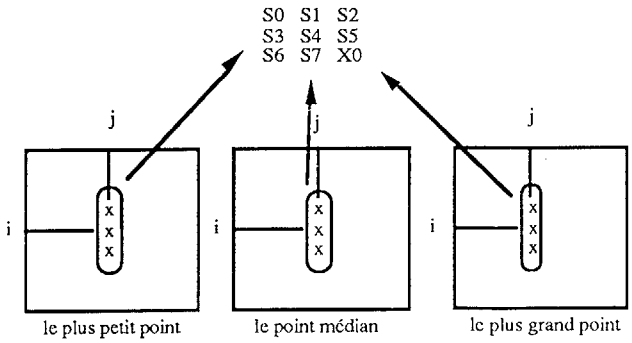


Figure 3 : la deuxième étape du calcul de la valeur médiane.

Nous pouvons généraliser cette méthode à des fenêtres de tailles plus importantes.

Fenêtre 5x5

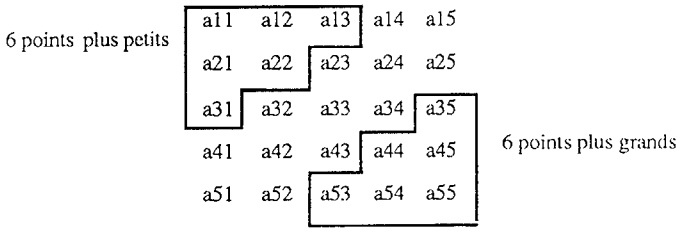


Figure 4 : points restants à trier pour une fenêtre 5x5.

Nous démontrons de la même manière qu'à l'issue d'un tri sur les lignes puis sur les colonnes, il est possible d'éliminer les deux ensembles représentés sur la figure 4. Si dans le cas d'un voisinage 3x3 nous pouvions déterminer la valeur médiane par un simple tri sur la diagonale il n'en est pas de même ici. L'ensemble qui reste à classer comprend 13 points. Il faut donc trier une ligne, les 5 colonnes et les 13 points.

1 ligne -> 10 comparaisons
5 colonnes -> 50 comparaisons
la médiane de 13 points -> 48 comparaisons
Soit 108 comparaisons.

Généralisation à une fenêtre (2n+1)x(2n+1)

Déterminons :

• le nombre de points inférieurs à la médiane INFMED (figure 5)

$$INFMED = [\ (n + 1)(n +2)\ ] / 2$$

• le nombre de points supérieurs à la médiane SUPMED (figure 5)

$$SUPMED = [\ (n +1)(n +2)\ ] / 2$$

Soit R le reste des points à trier : $R = (2n+1)^2 - (n +1)(n +2)$

$$R = n(3n + 1) -1$$

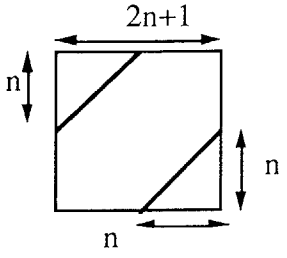


Figure 5 : Le découpage d'une image montrant le sous-ensemble de points restants à trier

• le nombre total de comparaisons pour les points restants est :
nbcomprest = x(x +1) + x avec x = (R - 1) / 2
d'où

$$\text{nbcomprest} = [\ (R - 1)\ /2]^2 + (R - 1)$$
$$= 1/4\ (R^2 - 2R + 1) + (R - 1)$$
$$= 1/4(9n^4 + 6n^3 - 11n^2 - 4n + 4) + 3n^2 + n - 2$$

Soit $\text{nbcomprest} = 9/4n^4 + 3/2n^3 + 1/4n^2 - 1$

• Calcul du nombre total de comparaisons:
Le total des comparaisons comprend le nombre de comparaisons de la ligne, de (2n +1) colonnes et des points restants, soit :

$$\text{nbcomp1} = n(2n + 1) + n(2n +1)(2n +1) + \text{nbcomprest}$$

$$\text{nbcomp1} = 9/4n^4 + 11/2n^3 + 25/4n^2 + 2n - 1$$

Taille de la fenêtre	3x3	5x5	7x7	9x9
Tri total	30	234	900	2572
Traitement séparable	15	108	392	859

Tableau 1.

4. LE PSEUDO-MEDIAN

Pour réduire le temps de calcul nous proposons un traitement séparable qui ne produit pas nécessairement la valeur médiane de la fenêtre considérée comme résultat. En effet, nous ne cherchons plus la valeur médiane de 9 points (dans le cas d'un fenêtre 3x3), mais la médiane des médianes des 3 lignes de la fenêtre. Nous obtenons un algorithme de filtrage dont le temps de calcul est nettement inférieur à celui du calcul de la médiane. L'algorithme est le suivant :

Rij = Médiane [Médiane (V11, V12, V13), Médiane (V21, V22, V23), Médiane (V31, V32, V33)]

Le traitement est effectué de la manière suivante :

$Rij = \text{Médiane}_t\ (M1_{t-2}, M2_{t-1}, M3_t)$ avec

$M1_{t-2}$ = Médiane (V11, V12, V13),
$M2_{t-1}$ = Médiane (V21, V22, V23),
$M3_t$ = Médiane (V31, V32, V33).

A l'instant t, la médiane $M3_t$ est à calculer ainsi que la médiane Rij, soit 2 médianes de 3 points à calculer contre une médiane de 9 points.
Ce traitement élimine 6 points de la fenêtre 3x3 (les 3 plus petites valeurs et les 3 plus grandes), puis on conserve un point parmi 3. Le résultat obtenu n'est pas la véritable médiane, mais l'application de ce filtre fournit un lissage satisfaisant pour l'observateur humain.
Démontrons que le traitement élimine 6 points :

M1 = Med (V11, V12, V13) implique qu'il existe a1,b1,M1 appartenant à {V11,V12,V13} tels que $a1 \leq M1 \leq b1$

M2 = Med (V21, V22, V23) implique qu'il existe a2,b2,M2 appartenant à {V21,V22,V23} tels que $a2 \leq M2 \leq b2$

M3 = Med (V31, V32, V33) implique qu'il existe a3,b3,M3 appartenant à {V31,V32,V33} tels que $a3 \leq M3 \leq b3$

Soit $an \leq Mn \leq bn$ avec $n = 1,2,3$ (1)

Rij = Med (M1 ,M2, M3) implique qu'il existe m1,m2,Rij appartenant à [M1,M2,M3] tels que $m1 \leq Rij \leq m2$ (2)

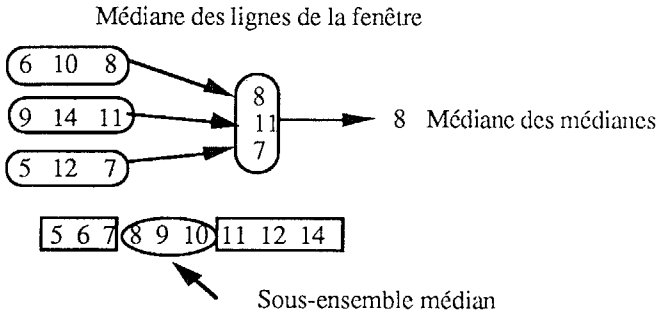
Posons m1 = Ms et Rij = Mt avec s appartenant à {1,2,3 } et t appartenant à {1,2,3} et $s \neq t$.

(1) et (2) implique $as \leq Ms \leq Mt$
(1) implique $at \leq Mt$

Les 3 points as, Ms, at sont inférieurs ou égaux à Mt égal à Rij et sont éliminés. De la même manière on montre que :

(1) et (2) implique $Mt \leq Ms \leq bs$
(1) implique $Mt \leq bt$

et donc que les 3 points Ms, bs, bt sont inférieurs ou égaux à Mt égal à Rij et sont éliminés. On prend donc un point parmi les 3 valeurs médianes des 9 points de la fenêtre 3x3.
L'exemple suivant montre le résultat obtenu par ce traitement :



L'intérêt d'un tel traitement réside dans sa facilité de parallélisation sur une structure SIMD, dans le cas présent sur un Processeur Ligne, à cause de la séparabilité du traitement en ligne.

4.2 Mise en oeuvre

Pour parvenir au résultat escompté nous allons utiliser une image de travail intermédiaire T. L'algorithme va s'exécuter en deux passes :
- Dans la première passe l'algorithme travaille sur une image de départ V et produit une image intermédiaire T,
- Dans la seconde passe l'algorithme travaille sur l'image intermédiaire T résultat de la première passe.

1 ère Passe.

On calcule la valeur médiane des points Vi,j-1, Vi,j, Vi,j+1 de l'image de départ puis on écrit le résultat dans l'image intermédiaire. Le balayage de l'image de départ se fait horizontalement (segment horizontal de processeurs, balayage linéaire).

2 ème Passe.

On calcule la valeur médiane de Ti-1,j, Ti,j, Ti+1,j de l'image de travail puis on écrit le résultat dans l'image d'arrivée. Le balayage de l'image de travail peut se faire soit verticalement (segment vertical de processeurs, balayage linéaire), soit horizontalement.

On peut généraliser le traitement à des voisinages plus grands. Jusqu'à un voisinage 5x5 cet algorithme est intéressant sur le Processeur Ligne car il nécessite peu d'accès mémoire.

5. CONCLUSION

Nous avons exposé deux méthodes de calcul séparable pour réduire le temps d'exécution. La première méthode donne la valeur médiane de la fenêtre considérée en diminuant le temps de calcul de moitié. Pour pouvoir diminuer encore plus ce temps de calcul nous avons défini un autre filtrage que nous avons appelé pseudo-médian. Le filtrage obtenu à l'issu du pseudo-médian donne un lissage qui peut être suffisant pour certain type d'applications. L'intérêt réside dans un temps de calcul qui est nettement inférieur au médian puisque nous obtenons un rapport 2,6 sur SYMPATI2. En effet pour une image 256x256 avec 32 PE le temps d'exécution du *filtre médian est de 17,72 ms pour une fenêtre 3x3*, alors qu'il est de 6,64 ms *pour le filtre pseudo-médian sur une fenêtre 3x3*.
De plus si on augmente le voisinage du filtre *pseudo-médian à 5x5* on obtient un temps de calcul de 16,24 ms qui reste inférieur à celui du filtre de Tukey 3x3.
L'intérêt de ces deux approches réside dans la facilité de parallélisation sur une structure SIMD à cause de la séparabilité du traitement en ligne et en colonne. Ces deux algorithmes ont été développés en 4LP (Low Level Language for Line Processor) [FER90] sur le Processeur Ligne SYMPATI2.

5. BIBLIOGRAPHIE

[FER90] -FERNANDEZ P., ADAM P., JUVIN D., BASILLE J-L : 4LP Low Level Language for Line Processor. EUSIPCO 90 V European Signal Processing Conference, Barcelona, September 18-21 1991.

[HUN79] -HUANG T.S, YANG G.J, TANG G.J : A fast two-dimensionnnal median filtering algorithm. IEEE trans. Acoust., Speech and Signal Process, Vol ASSP-27, n°1 1979, pp13-18.

[JUVI88] -JUVIN D., BASILLE J-L, ESSAFI H., LATIL J-Y : SYMPATI2, a 1,5 D Processor array for image application. Eusipco Signal Processing IV : Theories and applications, North-Holland, pp 311-314 1988.

[PRES89] -PRESTON K. Jr : The Abingdon Cross benchmark survey-computer july 1989 pp9-18.