

RESTAURATION

FILTRAGE du BRUIT

b- filtres non linéaires

FILTRES NON LINÉAIRES

Objectifs : Réduire le bruit, ou supprimer des points aberrants, en respectant les contours.

Principe : Remplacer le pixel central (P_c) par une valeur calculée dans le voisinage de P_c , en filtrant plus ou moins, suivant la situation (zone contour ou homogène)

But : ne pas combiner des pixels appartenant à des objets différents

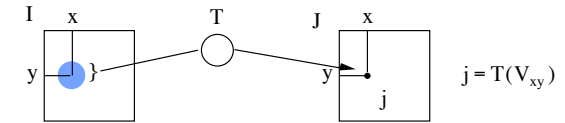
=> méthodes non linéaires :

Choix des pixels :

- Tous les pixels, avec un poids variable (inverse gradient)
- Sélection des pixels dans le voisinage (KNN, Sigma)
- Sélection d'une zone particulière (SNN, homogénéité maximum)

Valeur de substitution - Majoritaire (médian)
- Combinaison linéaire des pixels sélectionnés

TRAITEMENT LOCAL



En faisant intervenir les pixels voisins de (x, y) , permet de :

- restaurer une image dégradée
- corriger du bruit ou des parasites
- renforcer ou supprimer des petits détails
- détecter des structures locales (contours,...)

Principe : La nouvelle valeur du pixel courant P_c est calculée à partir des valeurs des pixels voisins.

Paramètres :

- forme et taille du voisinage
- poids des pixels voisins

Implantation

- séparable ou non
- problème des bords de l'image

FILTRES NON LINÉAIRES : ÉVALUATION

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20



*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20

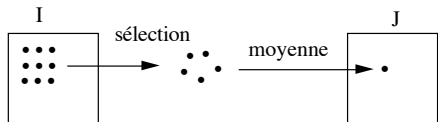


.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Zone de transition non bruitée (respect des contours)

FILTRES NON LINÉAIRES

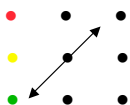
Remplacer le pixel central (Pc) par une **fonction d'une sélection des pixels voisins**



Sélection : **KNN** : les k pixels les plus proches, en valeur

Sigma : les pixels situés dans $[I(x,y) - \sigma, I(x,y) + \sigma]$

SNN : pour chaque paire de pixels symétriques par rapport à Pc, on sélectionne celui qui est le plus proche de Pc, en valeur



MinMax : si le Pc est plus grand (plus petit) que le Max (Min) des voisins, on attribue la valeur Max (Min), sinon il est inchangé

FILTRE INVERSE du GRADIENT

Remplacer le pixel central par la moyenne des voisins, pondérée par l'inverse du gradient

$$\text{gradient : } \Delta_{kl} = |I(x+k, y+l) - I(x,y)|$$

$$\delta_{kl} = \begin{cases} 1 / \Delta_{kl} & \text{si } I(x+k, y+l) \neq I(x,y) \\ 1 / 2 & \text{sinon} \end{cases}$$

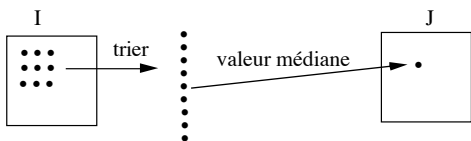
$$\text{poids : } w_{kl} = \frac{1}{2} \frac{\delta_{kl}}{\sum_{k,l} \Delta_{kl}}, \quad w_{00} = 1 / 2$$

$$\text{combinaison : } J(x,y) = \sum_k \sum_l w_{kl} \cdot I(x+k, y+l)$$

FILTRES NON LINÉAIRES

Filtre médian

remplacer un point aberrant par la valeur majoritaire



Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

non triés

51
21
14
9
9
14
19
9
19
9
22
22
27
27

triés

9
9
9
14
19
21
22
27
51

median

19

« même » nombre de pixels de chaque côté

FILTRE MÉDIAN : ÉVALUATION

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20



*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20

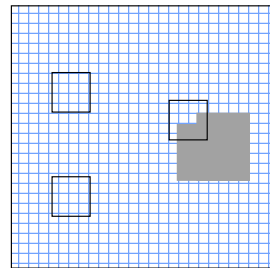
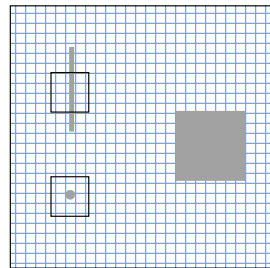
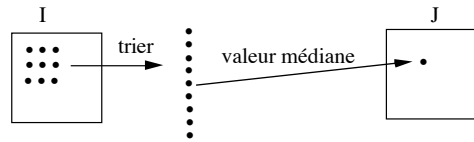


.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

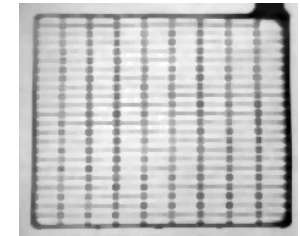
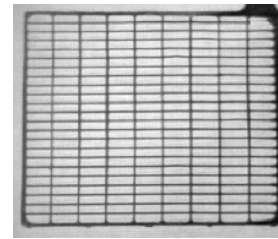
Zone de transition non bruitée (respect des contours)

FILTRE MÉDIAN

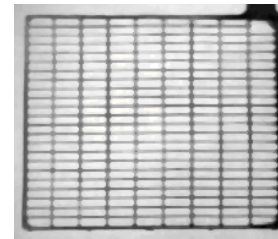
Filtre médian remplacer un point aberrant par la valeur médiane



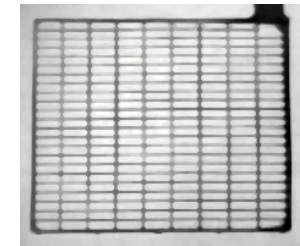
FILTRE MÉDIAN



médian
5x5



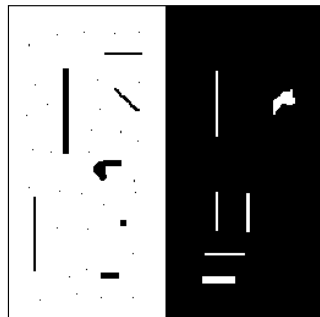
médian 3x3 (1 iter)



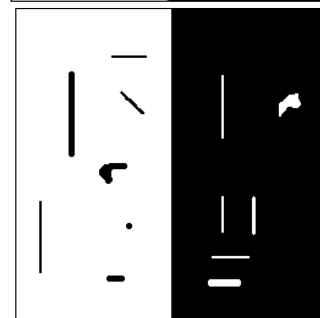
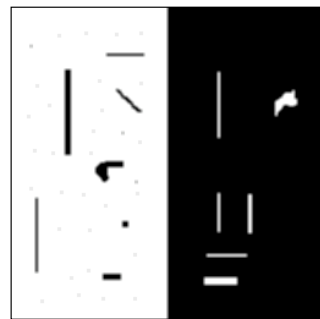
médian 3x3 (2 iter)

FILTRE Médian/Convolution

Convol 3x3

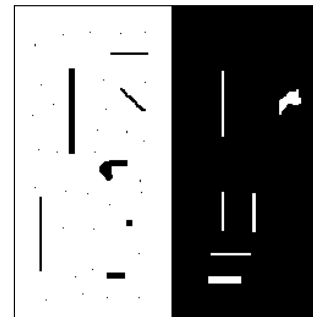


Médian 3x3

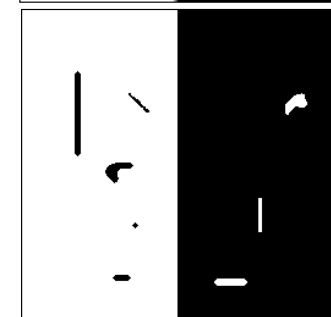


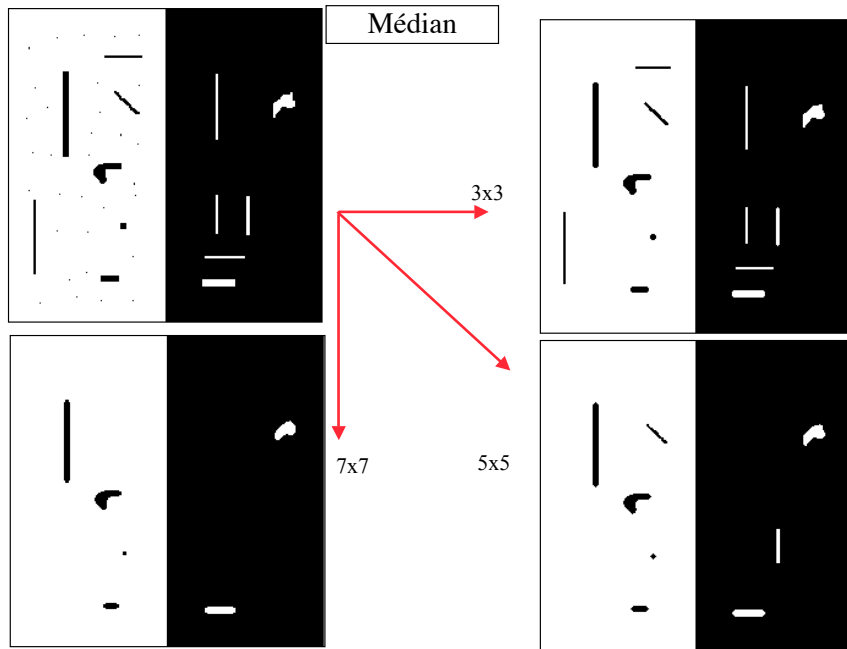
FILTRE Médian/Convolution

Convol 5x5



Médian 5x5





Filtre médian

Original bruité



Filtrage :

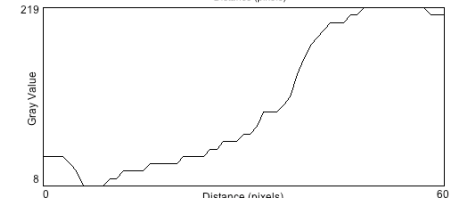
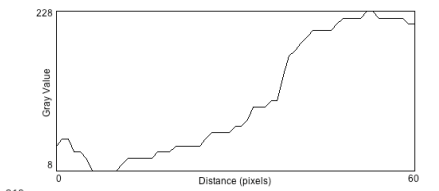
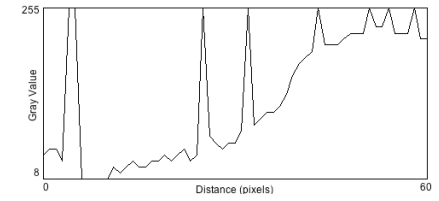
médian 3x3



médian 5x5

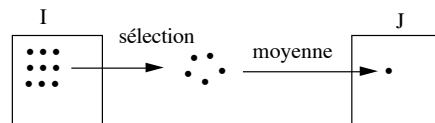


Profil de la ligne



FILTRES NON LINÉAIRES

Remplacer le pixel central (P_c) par la **moyenne d'une sélection des pixels voisins**



Objectif : sélectionner les pixels appartenant à la même classe que le pixel central

Différentes méthodes suivant le critère de sélection

FILTRES NON LINÉAIRES : KNN

Sélection : les k pixels dont les valeurs sont les plus proches de celle du P_c

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Ex : $k=4$

$\{9, 9, 14, 19\} \Rightarrow$ moyenne = 13

FILTRE KNN : ÉVALUATION

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20



*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Zone de transition non bruitée (respect des contours)

FILTRES NON LINÉAIRES : Sigma

Sélection : les pixels situés dans $[I(x,y) - \sigma, I(x,y) + \sigma]$

Si le nombre de pixels sélectionnés est inférieur à un seuil k donné, on affecte la moyenne des 8 voisins.

La valeur de k dépend de la taille du voisinage utilisé pour la sélection.

σ peut être déduit de l'écart-type de l'image.

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Ex : $\sigma = 13 \Rightarrow [0, 22]$

{ 9, 9, 14, 19, 21, 22 } \Rightarrow moyenne = 16

FILTRE Sigma : ÉVALUATION σ entre 5 et 9

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20



*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Zone de transition non bruitée (respect des contours)

FILTRES NON LINÉAIRES : SNN

Sélection : pour chaque paire de pixels symétriques par rapport à P_c , on sélectionne celui dont la valeur est la plus proche de celle de P_c



Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

{ 9, 14 } \Rightarrow 9
 { 9, 19 } \Rightarrow 9
 { 51, 27 } \Rightarrow 27
 { 21, 22 } \Rightarrow 21

Moyenne = 16

FILTRE SNN : ÉVALUATION

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20



*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20



.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20

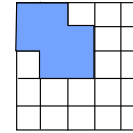


.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

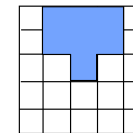
Zone de transition non bruitée (respect des contours)

FILTRES NON LINÉAIRES : NAGAO

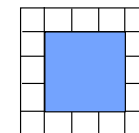
Zone : On découpe le voisinage en plusieurs zones.
On sélectionne la zone la plus homogène (σ minimal)
On calcule sa moyenne et on l'affecte au P_c



4 zones analogues par rotation

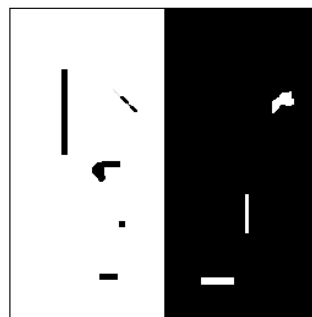
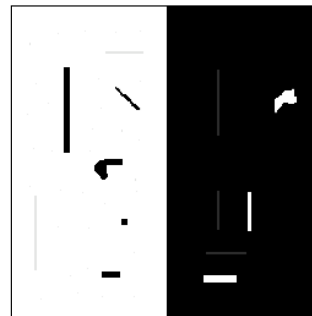
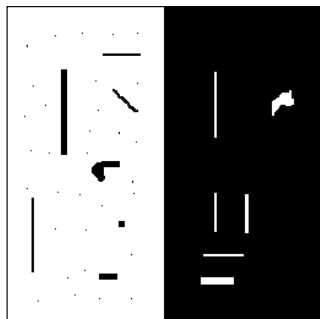


4 zones analogues par rotation

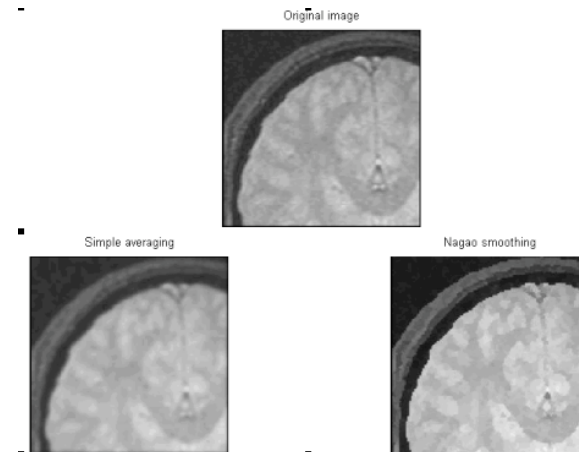


1 zone

FILTRES de NAGAO



FILTRES de NAGAO



FILTRE Min-Max (filtre conservatif)

Principe : on détermine la valeur Min et la valeur Max dans la fenêtre.
 Si le pixel central Pc a une valeur supérieure à Max, on affecte Max
 Si le pixel central Pc a une valeur inférieure à Min, on affecte Min
 Si le pixel central Pc est compris entre Min et Max, on affecte la valeur de Pc

Image

10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	3	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Min = 9
 Max = 51
 Pc = 3 → Nouvelle valeur : 9

FILTRES MIN-MAX: ÉVALUATION

21	20	20	22	21
20	20	20	21	20
20	21	15	20	22
20	20	20	22	21
20	21	20	20	20

*	*	*	*	*
*	*	*	*	*
*	*	20	*	*
*	*	*	*	*
*	*	*	*	*

Bruit dans région homogène

5	4	20	22	21
4	5	20	21	20
4	4	15	20	22
3	5	20	22	21
4	4	20	20	20

.	.	*	*	*
.	.	*	*	*
.	.	15	*	*
.	.	*	*	*
.	.	*	*	*

Bruit dans zone de transition (contour)

5	4	20	22	21
4	5	20	21	20
4	4	20	20	22
3	5	20	22	21
4	4	20	20	20

.	.	*	*	*
.	.	*	*	*
.	.	20	*	*
.	.	*	*	*
.	.	*	*	*

Zone de transition non bruitée (respect des contours)

FILTRE Min-Max



Le bruit de taille > 1 pixel n'est pas supprimé.
 Bon respect des détails

FILTRES : comparaison



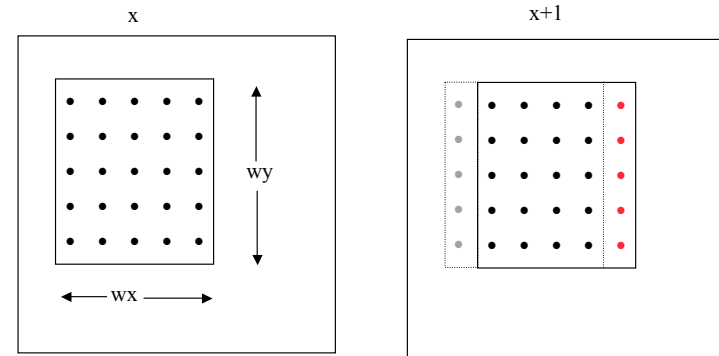
moyenne

FILTRES : comparaison



médian

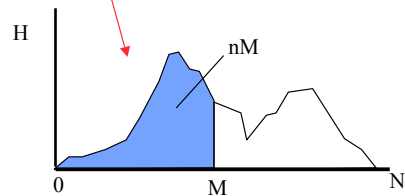
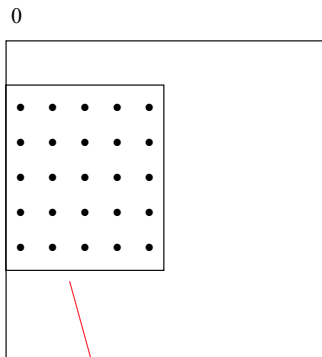
TD : MEDIAN RAPIDE



$df = wx * wy / 2$
demi-effectif de la fenêtre

départ de wy pixels
arrivée de wy nouveaux pixels
 $2 * wy$ changements au lieu de $wx * wy$

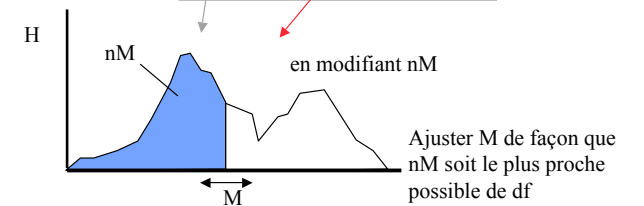
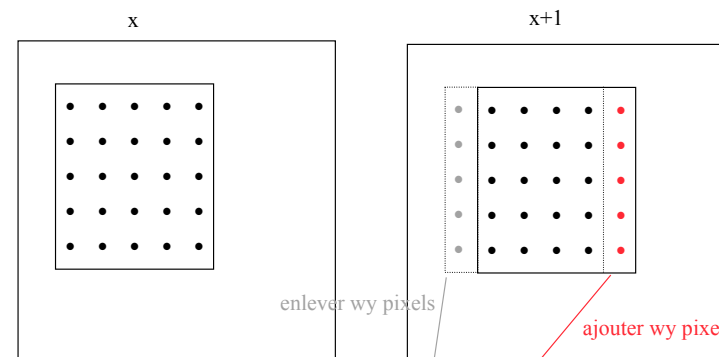
TD : MEDIAN RAPIDE



$M = \text{médiane}$

$nM = \text{nombre de pixels inférieurs à la médiane}$

TD : MEDIAN RAPIDE



Ajuster M de façon que nM soit le plus proche possible de df

TD : MEDIAN RAPIDE

```
Median rapide
{ // colG : colonne Gauche de la fenetre precedente
  // colD : colonne droite de la fenetre courante
  // H : histogramme
  // M : valeur de la mediane de la fenetre
  // nM : nombre de pixels de valeur inferieure a M
  //      dans la fenetre
  // wx, wy :taille de la fenetre
  // Ix, Iy : taille de l'image I
  df =wx*wy/2; // demi effectif de la fenetre

  for (i=wy/2+1; i<Iy-wy/2; i++) // n° de ligne
  { //1- calculer l 'histogramme de la premiere fenetre
    //2- determiner M en parcourant H
    //3- calculer nM
    for (j=wx/2+1; j<Ix-wx/2; j++) // n° de colonne
      { charger colG et colD
        calculM();
      }
  }
}
```

TD : MEDIAN RAPIDE

```
calculM () // mise a jour H et M
{ for (k=0; k<wy; k++)
  { g=colG[k]; H[g]--;
    if (g<M) nM--;

    g = colD[k]; H[g]++ ;
    if (g<M) nM++;
  } // nM = nb de pixels de la fenetre courante
  //inferieurs a la mediane de la fenetre precedente

  // trouver la nouvelle valeur de M et de nM
  if (nM > df)
    do {M--;
      nM=nM-H[M];
    }while (nM > df);
  else while (nM+ H[M] <= df)
    { nM = nM + H[M];
      M++ ;
    }
}
```