Traitement d'images TP3 : Convolution spatiale

Filtrage Passe Bas : Réduction de Bruit additif

Filtrage Passe haut : Réhaussement de contraste

Line POUVARET, Hamdi BENAOUN

2015-2016

Travail préliminaire

1°) La somme en ligne notée s_n des éléments du triangle est égale à 2n.

2°) Par analogie avec la formule binomiale : $(x+y)^n = (x+y)^{n-1} * (x+y)$, on constate que $(h_1)^n[k] = (h1)^{*n-1}[k] * (h_1)[k]$

Par exemple, pour l'ordre n=2, la convolution de (1 1) avec (1 1) donne bien (1 2 1) et ainsi de suite pour les ordres supérieurs.

Dans MatLab :

```
conv2([ 1 1], [1 1]) = [1 2 1]
conv2([1 2 1], [1 1]) = [ 1 3 3 1]
```

3°) Moy(n) = 0 pour chaque n.

- n=0 → Var(n) = 0
- n=1 → Var(n) = ($\frac{1}{4}$)*(1/2) + (1/4)*(1/2) = 1/4
- n=2 → Var(n) = (1)*(1/4)+(1)*(1/4) = 1/2
- n=3 → Var(n) = (9/4)*(1/8)+(1/4)*(3/8) = 3/4
- n=4 → Var(n) = 1

  On remarque que la Variance pour un n donné est égale à n*(1/4)

Fichier TP_reduction_bruit.m

1°) Dans MatLab :

```
%--------------------------------------
%CALCULER LA VARIANCE DES NIVEAUX DE GRIS SUR IMA_GRIS
%variance des niveaux de gris dans l'image
%var_ima=.......
var_ima = var(ima_gris(:));

%FIXER LA VARIANCE DU BRUIT
%Parametrage du niveau de bruit
%rsb=..... \%en dB entre 10 et 20
%var_b= .......
```

```
rsb=10;
var_b=var_ima/(10*(rsb/10));
%----------------------
%FORMER l'IMAGE BRUITEE
%Bruit gaussien additif
%bruit= ................;
bruit=randn(size(ima_gris))*sqrt(var_b);

%Generation de l'image bruitee
%ima_bruit= ...................;
ima_bruit=ima_gris+bruit;
```

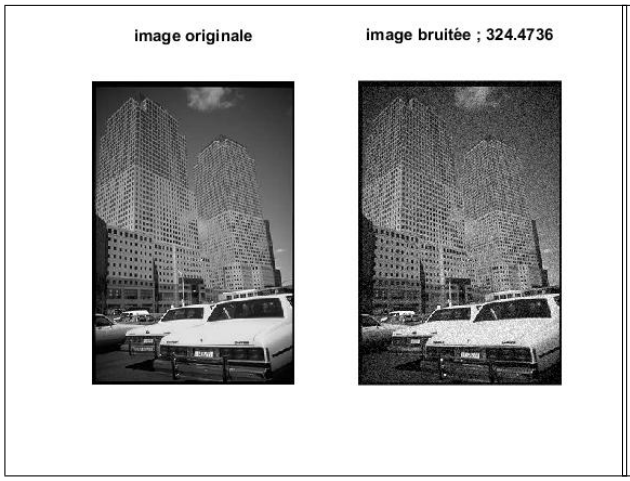

FIGURE 1 – Image bruitée avec RSB = 10

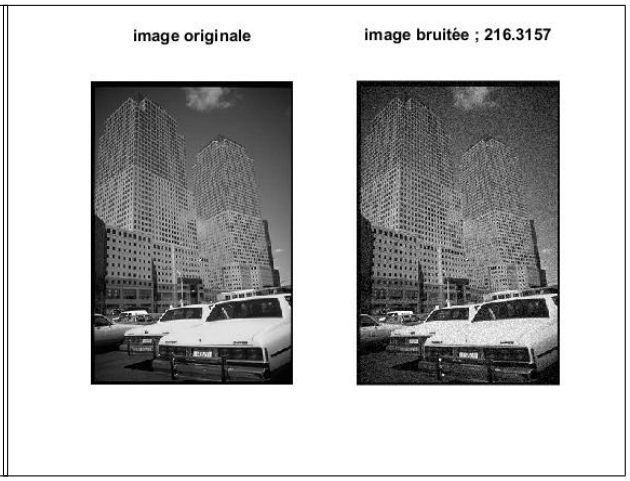

FIGURE 2 – Image bruitée avec RSB = 15

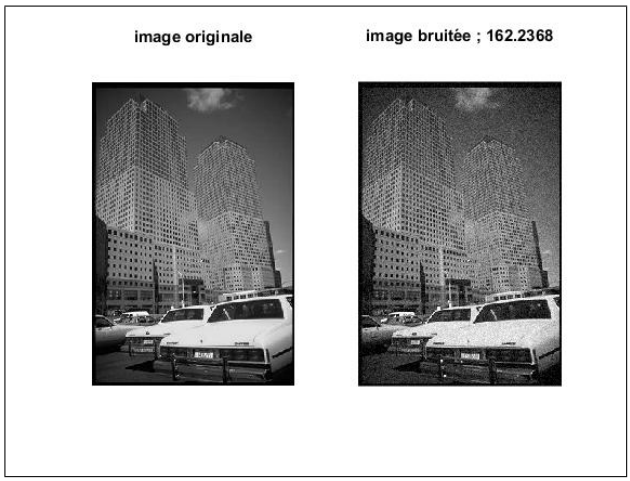

FIGURE 3 – Image bruitée avec RSB = 20

On constate que plus on diminue le RSB, plus l'image sera bruitée (beaucoup plus de grains sur l'image).

2°) Dans MatLab :

```
%Initialisation de variable si necessaire pour la generation du filtre
%binomial
%.....
for i=1:length(taille_kernel)

```

```
    %Convolution avec le filtre moyenne
    %-------------------------------------------------------------
    %FORMER LE NOYAU DU FILTRE MOYENNE ET EFFECTUER LA CONVOLUTION
    % kernel_moy=.......................;
    kernel_moy=ones(taille_kernel(i)).*(1/power(taille_kernel(i),2));
    % ima_moy=..........................;
    ima_moy=conv2(ima_bruit,kernel_moy, 'same');
    %Convolution avec le filtre gaussien
    %-------------------------------------------------------------------------
    %FORMER RECURSIVEMENT LE NOYAU DU FILTRE GAUSS ET EFFECTUER LA CONVOLUTION
    % kernel_gauss = ........................;
    kernel_gauss = [1 1];
    for j=1:taille_kernel(i)-2
        kernel_gauss = conv2(kernel_gauss, [1 1]);

    end
    kernel_gauss=kernel_gauss'*kernel_gauss;
    kernel_gauss = kernel_gauss.*(1/(sum(sum(kernel_gauss))));
%Rmq : Verifier que la somme des coefficients des noyaux est egale a 1

    % ima_gauss=........................;
    ima_gauss=conv2(ima_bruit,kernel_gauss, 'same');
```

3°) Analyse des résultats : On considère RSB = 10 pour analyser les résultats sur une image très bruitée (différence plus notable entre les deux filtres)

Comparaison entre les filtres Moyenne et Gaussien pour une taille donnée de noyau

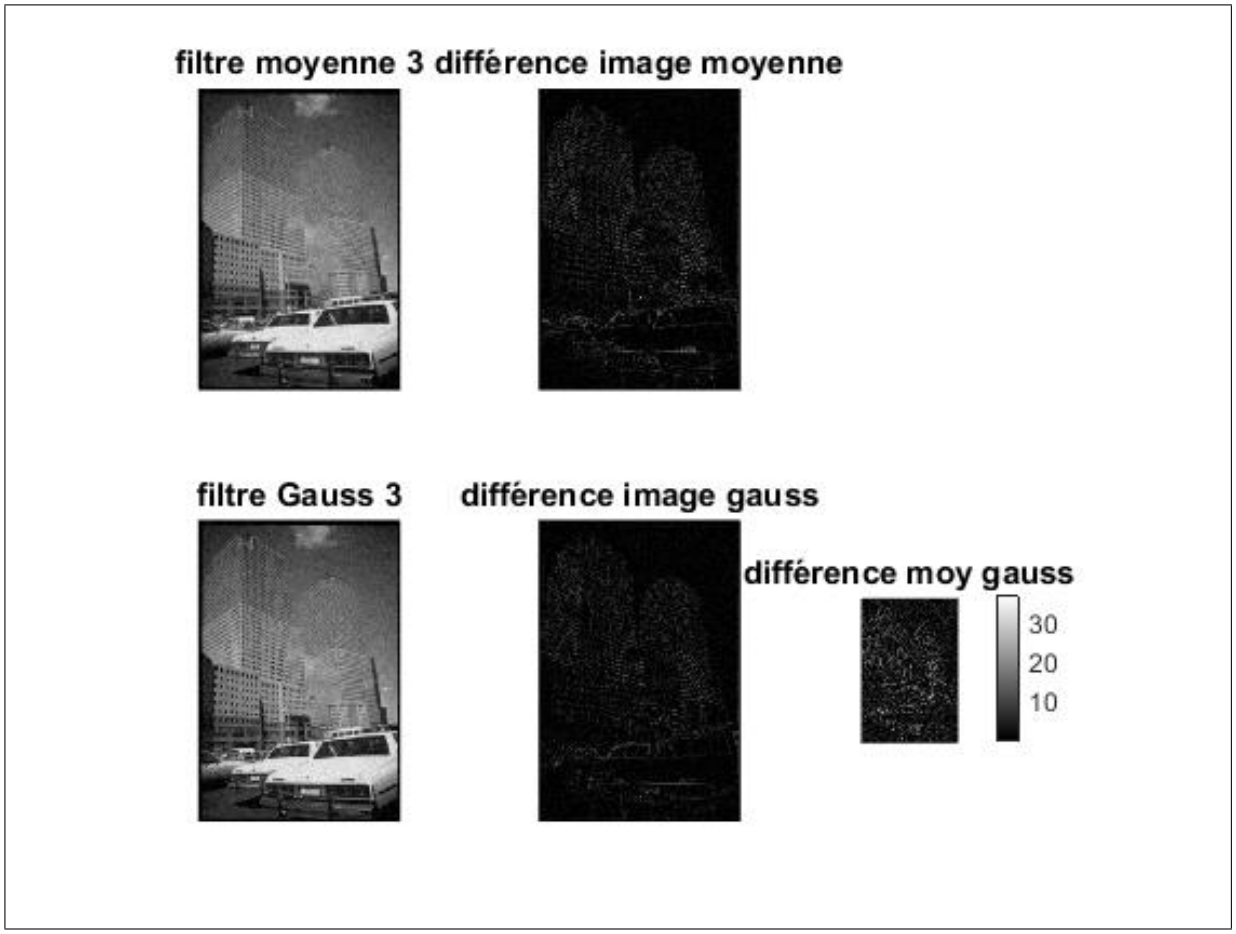

FIGURE 4 – Moyenne VS Gaussien, K = 3

La différence entre les deux filtres n'est pas notable pour cette taille de noyau.
On peut à la limite voir que le filtre moyenne perd un peu les contours dans l'image par rapport au filtre gaussien.

Le bruit est légèrement réduit.

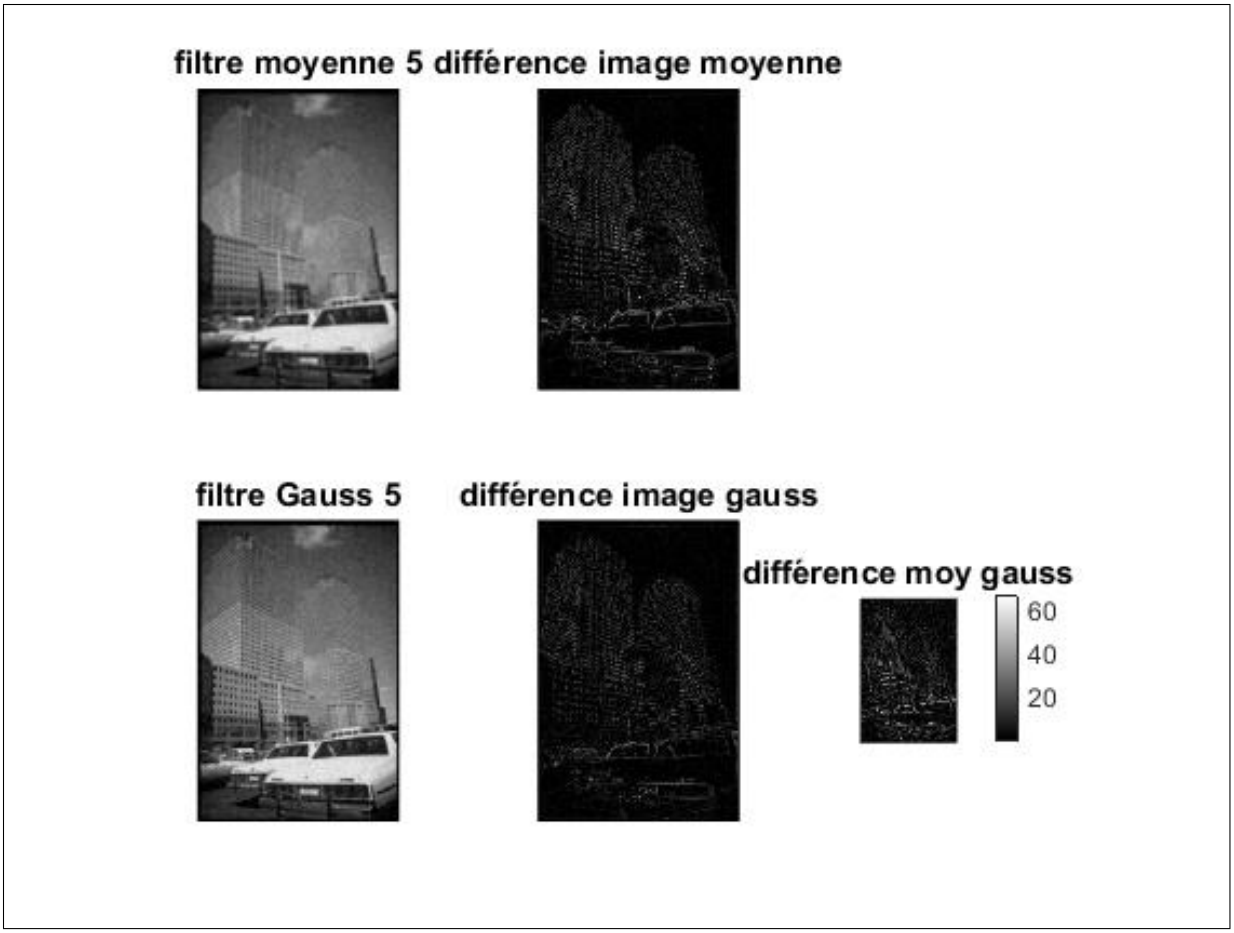

FIGURE 5 – Moyenne VS Gaussien, K = 5

Pour cette taille de noyau, on voit que le filtre moyenne réduit bien le bruit mais a tendance à trop flouter l'image et perdre les contours par rapport au filtre gaussien qui paraît plus efficace.

Le bruit est un peu plus réduit dans les deux cas.

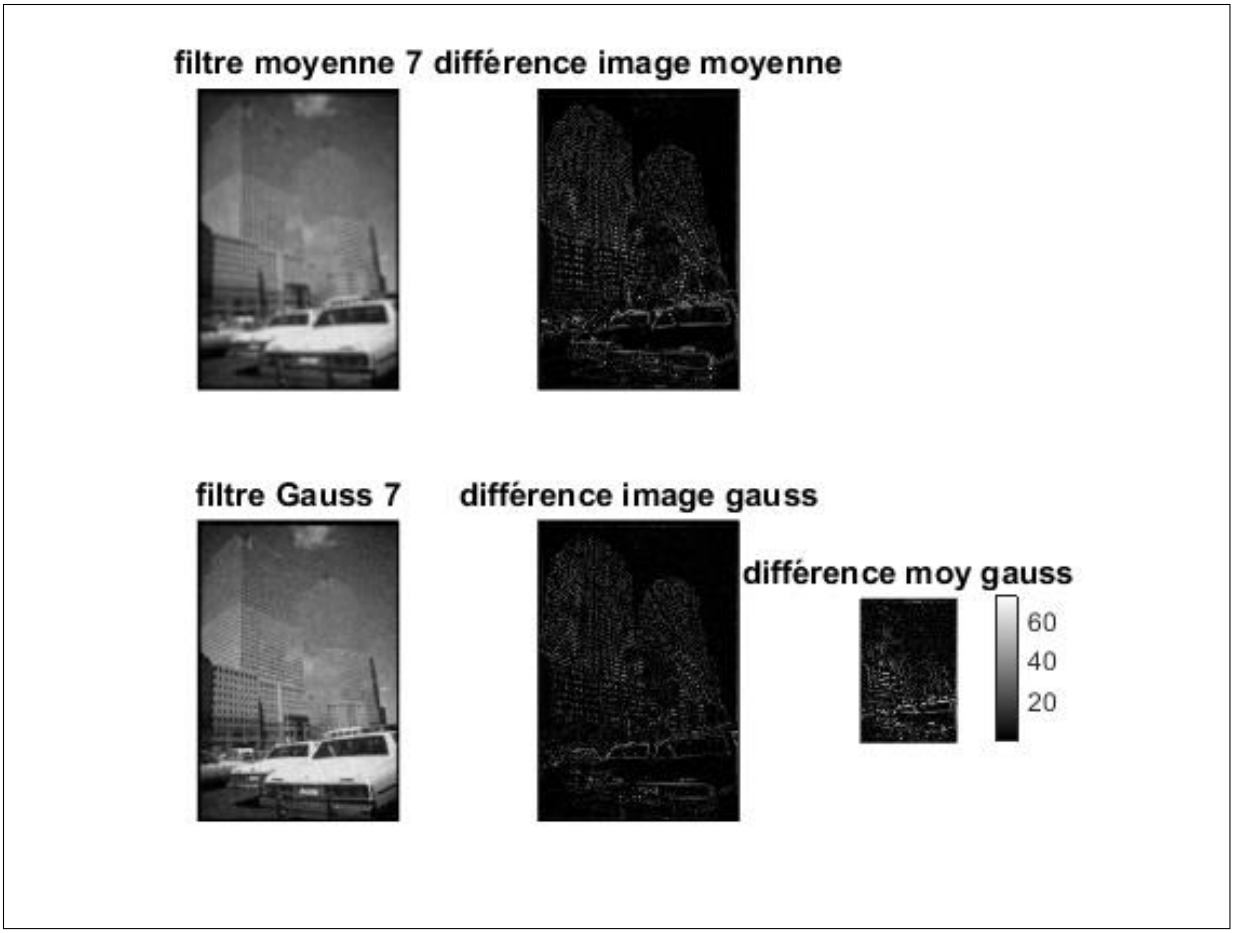

FIGURE 6 – Moyenne VS Gaussien, K = 7

Finalement, pour une taille de noyau de 7, le filtre moyenne floute énormément l'image, on perd quasiment tous les contours des objets et beaucoup d'information.
Le bruit est réduit mais étant donné que l'image est très floutée, on a du mal à bien mettre en évidence la réduction du bruit.

Le filtre gaussien conserve beaucoup mieux les contours, les détails sont malgré tout dégradés (les fenêtres de l'immeuble par exemple) mais l'information est beaucoup mieux conservée et le bruit assez bien réduit.

Comparaison entre les tailles des noyaux pour chaque filtre

Filtre moyenne

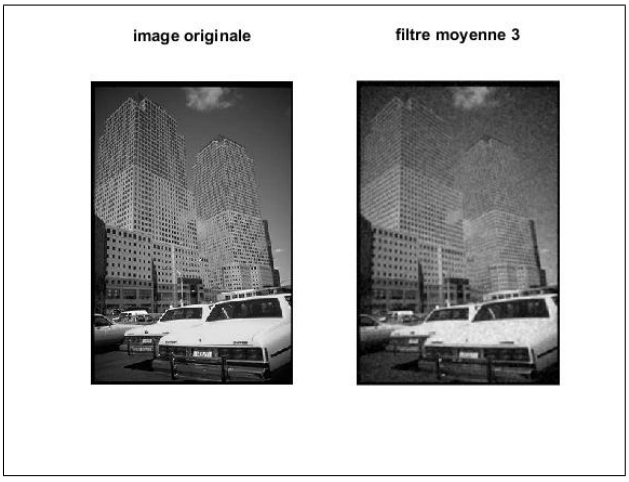

FIGURE 7 – Filtre moyenne, K=3

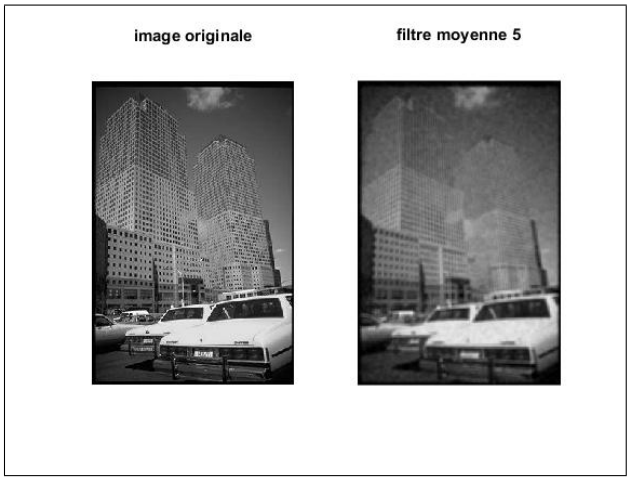

FIGURE 8 – Filtre moyenne, K=5

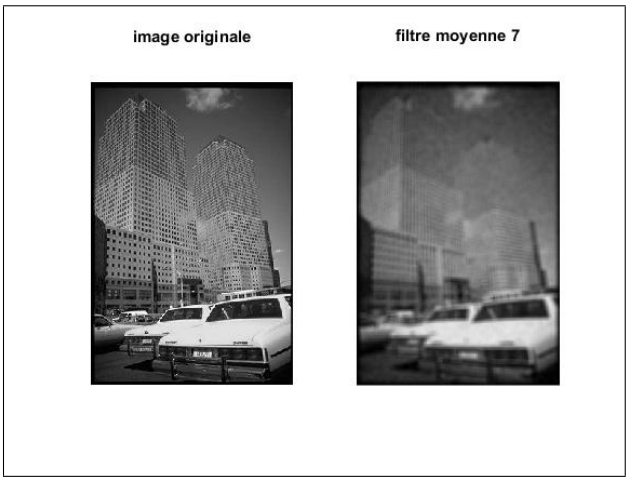

FIGURE 9 – figure
Filtre moyenne, K=7
Plus la taille du noyau de convolution va augmenter et plus l'image va se retrouver floutée avec le filtre moyenne.

Le bruit est de plus en plus réduit plus la taille du noyau est grande mais la perte des contours augmente aussi.

Filtre gaussien

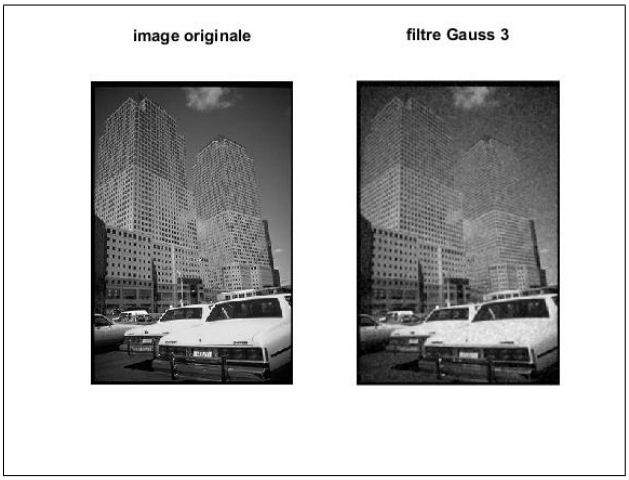

FIGURE 10 – Filtre gaussien, K=3

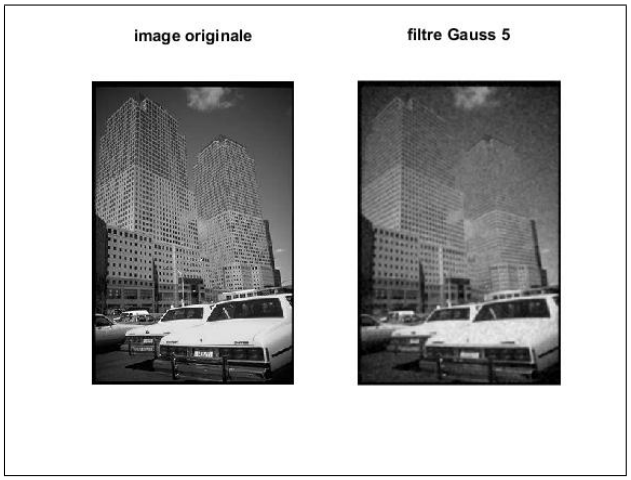

FIGURE 11 – Filtre gaussien, K=5

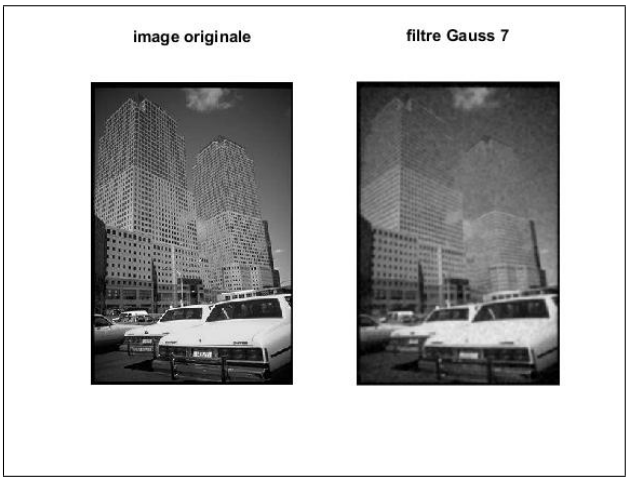

FIGURE 12 – Filtre gaussien, K=7

Plus la taille du noyau de convolution va augmenter et plus le bruit va être réduit.

On remarque aussi que les détails les plus sombres sont de plus en plus perdus (comme les fenêtres de l'immeuble), il y a quand même un léger flou mais les contours sont globalement bien conservés.

Fichier TP_rehaussement.m

1°) Dans MatLab :

```
%Filtrage Passe Haut 3 x 3
%Image filtree Passe Haut : Image originale - Image filtree Passe Bas
%Former le noyau du filtre Passe Haut a partir de son oppose Passe Bas
%---------------- A FAIRE----------------------%
%Ker_PB = .....;
taille_kernel = 7;
choix_filtre = 2;
if(choix_filtre == 1)
    Ker_PB = [1 1];
    for j=1:taille_kernel-2
```

```
        Ker_PB = conv2(Ker_PB, [1 1]);
    end
    Ker_PB=Ker_PB'*Ker_PB;
    Ker_PB = Ker_PB.*(1/(sum(sum(Ker_PB))));
elseif(choix_filtre == 2)
    Ker_PB=ones(taille_kernel).*(1/power(taille_kernel,2));
end
%Ker_PH = ......;
Ker_imp = zeros(taille_kernel, taille_kernel);
Ker_imp((taille_kernel+1)/2, (taille_kernel+1)/2) = 1;
Ker_PH = Ker_imp - Ker_PB;

%Convolution avec l'image
ima_PH = conv2(ima, Ker_PH, 'same');
```

```
%Filtrage Passe Haut de rehaussement
%Image rehaussee = image originale + alpha * image filtree passe haut
%Former le noyau de convolution du filtre equivalent
%alpha = ... ;
%Ker_RH = ..... ;
alpha = 5;
Ker_RH = Ker_imp + alpha*Ker_PH;

%Convolution avec l'image
ima_RH = conv2(ima, Ker_RH, 'same');
```

2°) Variations

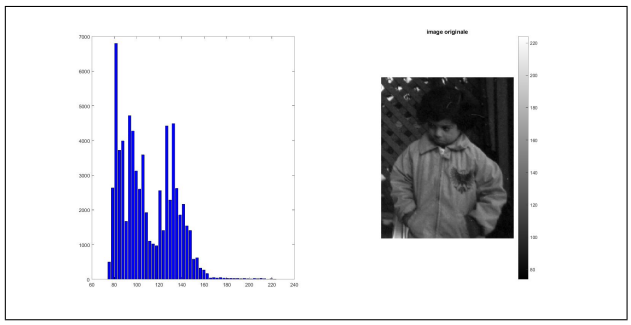

FIGURE 13 – Image originale

Réhaussements (filtre binomial)

$\alpha = \mathbf{2}$

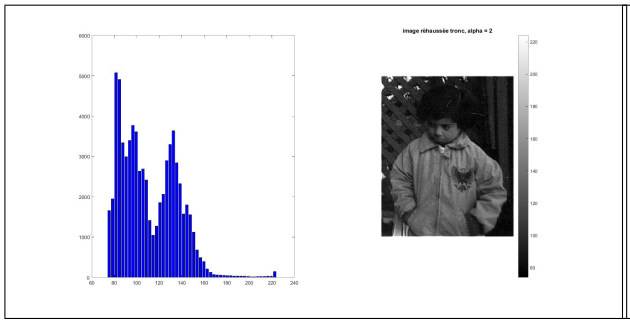

FIGURE 14 – Image réhaussée 1 fois, α=2

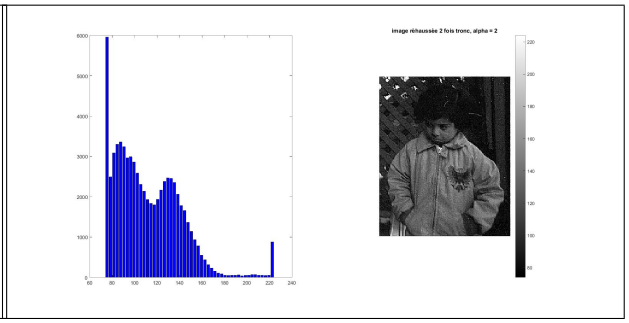

FIGURE 15 – Image réhaussée 2 fois, α=2

$\alpha = \mathbf{5}$

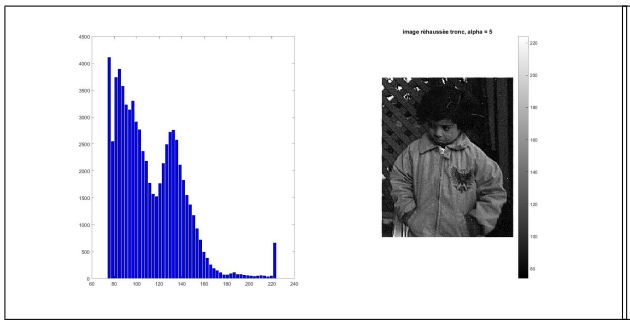

FIGURE 16 – Image réhaussée 1 fois, α=5

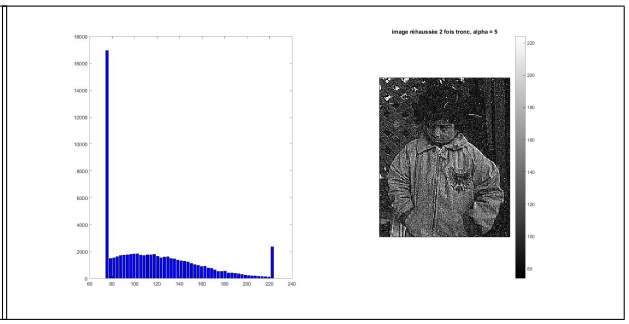

FIGURE 17 – Image réhaussée 2 fois, α=5

On remarque que plus on va augmenter α et plus on va réhausser le contraste mais avoir tendance à perdre de l'information sur l'image.

Réhaussements (filtre LaPlacien)

On remplace notre noyau binomial par un noyau LaPlacien :

```
1 alpha = 0.5;
2 Ker_L = [-1 -1 -1;-1 8 -1;-1 -1 -1];
3 Ker_RHL = Ker_imp + alpha*Ker_L;
4 %Convolution avec l'image
5 ima_RH = conv2(ima, Ker_RHL, 'same');
```

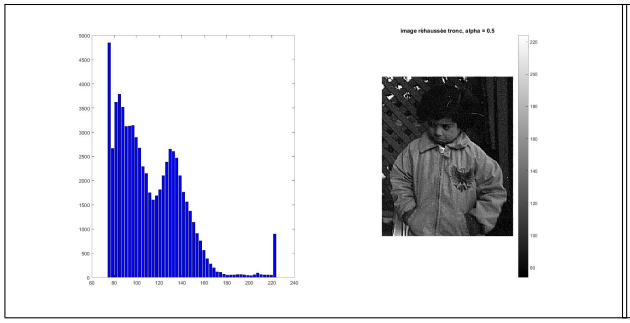

FIGURE 18 – Réhaussée 1 fois, α=0.5

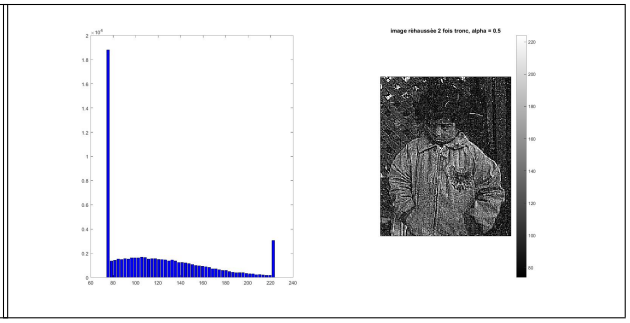

FIGURE 19 – Réhaussée 2 fois, α=0.5

Pour des valeurs α plus grandes, le filtre LaPlacien nous fait perdre beaucoup plus d'information sur l'image.

On peut imaginer d'autres types de noyau :

- en horizontal, au milieu :

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

- en vertical, au milieu :

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

- en diagonal :

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

- etc...
  Le rendu sera différent selon le filtre qu'on choisit.