TP TI: transformations locales des images

Kasperek Gautier
Sparrow Alice

Masque de convolution

Le tp porte sur la définition et le comportement de filtres moyenneurs simples. Pour cela nous allons apporter des modifications à l'image de la figure 1 ci-dessous.

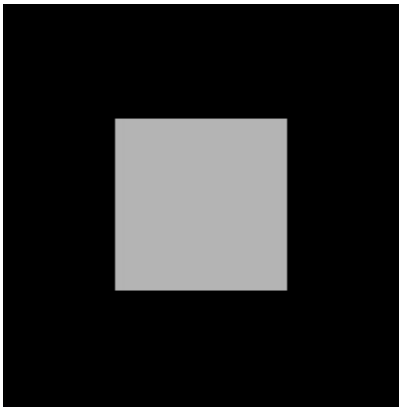

Figure 1 - Image I_2

Nous allons appliquer à cette image le filtre H_1 suivant :

$$H_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Ce filtre n'est pas normalisé car la somme de ses coefficients ne fait pas 1.

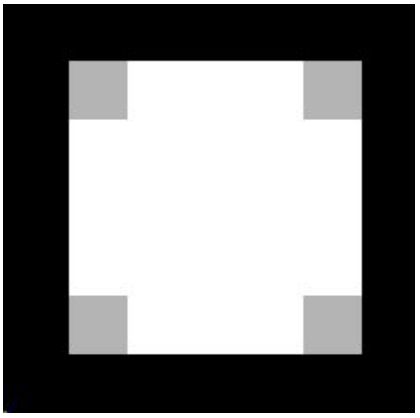

Figure 2 - Résultat de l'application de H1 sur l'image I_2

Nous allons maintenant normaliser ce filtre pour l'affecter de nouveau à l'image I_2

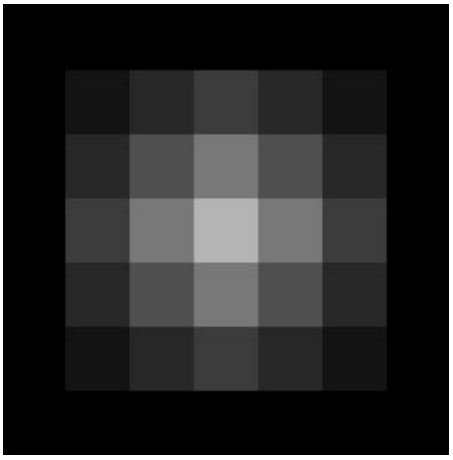

Figure 3 -*Résultat de l'application de H1 normalisé sur l'image* I_2

Afin de normaliser le filtre H_1, il nous a fallu multiplier tous les éléments de la matrice par $\frac{1}{9}$. En effet cela permet d'obtenir une somme des éléments de la matrice valant 1.

Effet de bords

Nous entamons maintenant la partie concernant les effets de bords. Il nous faut tout d'abord mettre le niveau 180 à tous les pixels de la première colonne de l'image I_2 pour obtenir l'image I_3 .

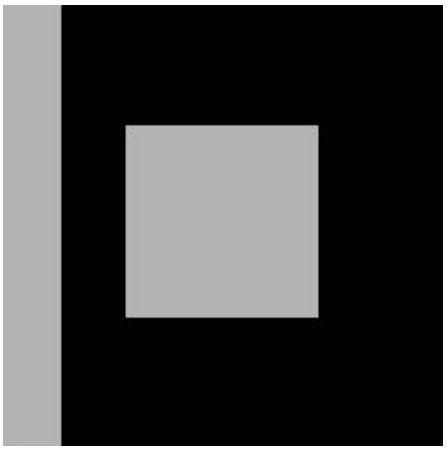

Figure 4 - Image I_3

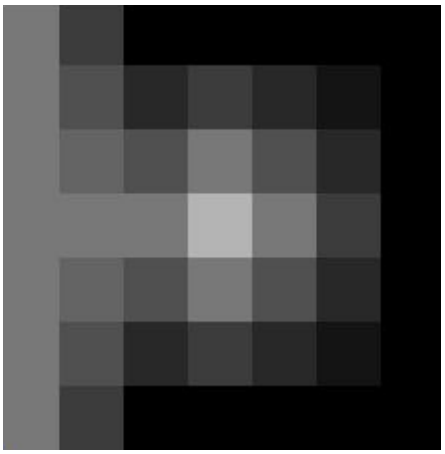

Figure 5 - Résultat de l'application de H1 normalisé sur l'image I_3

Après examination des niveaux de gris de la première colonne de l'image I_3 filtrée, nous pouvons déduire que ImageJ utilise la technique du miroir pour traiter les pixels du bord de l'image lors d'une convolution.
On le démontre comme suit :
Le niveau de gris du pixel aux coordonnées (0,0) de l'image I_3 filtrée est de 120.

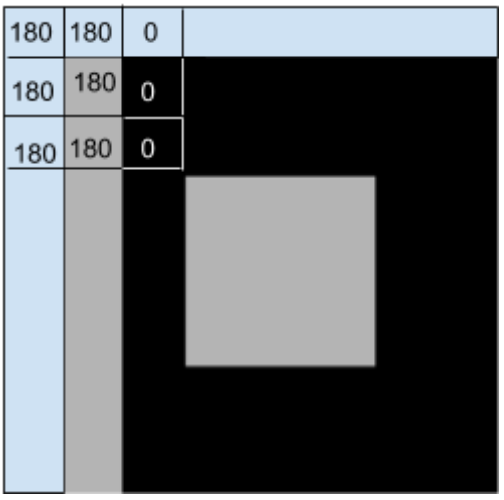

Figure 6 - Image I_3 filtrée avec la méthode miroir

On en déduit le calcul suivant :

$$\frac{180*6 + 0*3}{9} = 120$$

On retrouve bien la valeur relevée sur l'image filtrée.

Bruits et débruitage d'images

Nous allons maintenant tester la faculté des filtres à atténuer le bruit présent dans une image.

Bruit Gaussien d'une image synthétique

Pour traiter cette partie, nous avons créé une image I_4 *de taille* ***256x256****, de* ***8*** *bits de profondeur, et dont les pixels ont tous un niveau égal à* ***127****.*

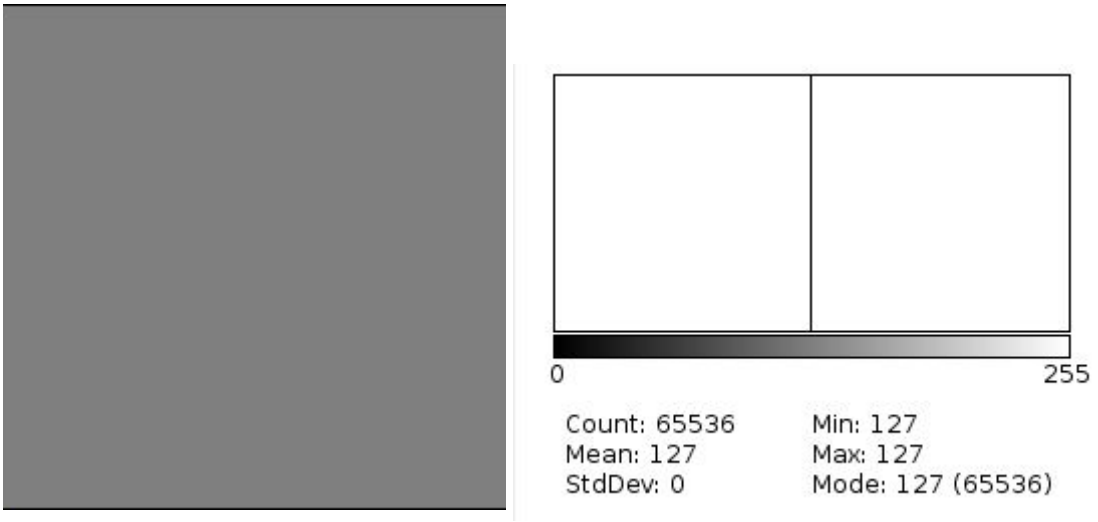

Figure 7 - Image I_4 *avec les pixels en niveau 127 et son histogramme*

On remarque que l'histogramme de cette image est composé d'une unique barre positionnée sur la valeur 127. Ce qui est normal car tous ses pixels on pour niveau de gris 127.

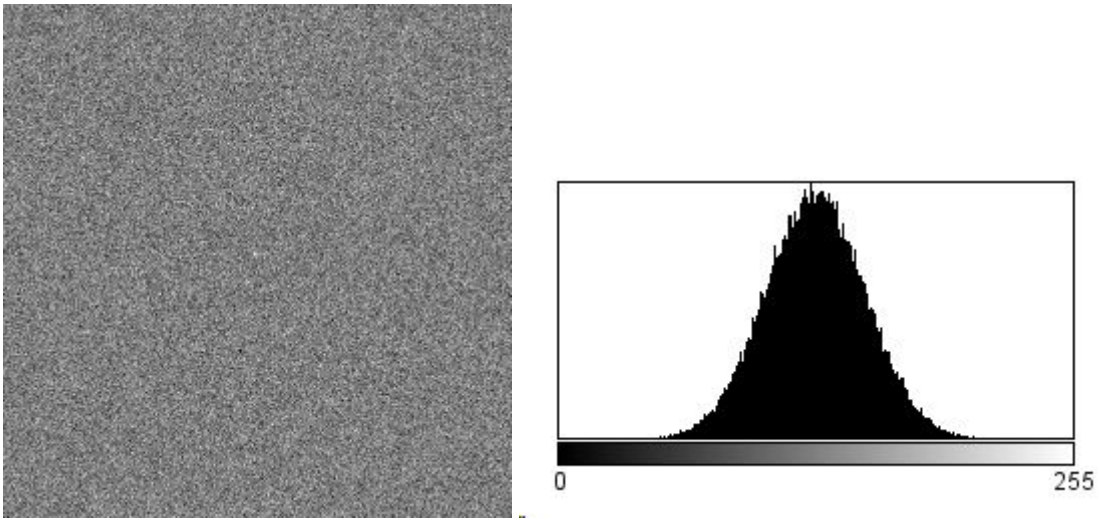

Figure 8 - Image I_4 *avec ajout de bruit et son histogramme*

On remarque que l'image bruité n'a plus rien à voir avec celle de départ. Son histogramme n'est plus caractérisé par une unique barre mais par une multitude de barres suivant une loi Gaussienne. L'histogramme nous permet de définir le bruit comme étant un bruit additif à distribution normale. En effet son histogramme est représenté par une gaussienne ce qui est caractéristique de ce type de bruit.

Le PSNR permet de mesurer la similarité entre deux images codées sur 8 bits et de définition M x N pixels. Il peut être utilisé pour quantifier la puissance du bruit ajouté à l'image ou pour mesurer la qualité du débruitage. En effet plus le PSNR de deux images est élevé plus les deux images se ressemblent.

PSNR(I_4 , I_4 bruitée) = 20.1704 dB

On peut caractériser le bruit précédent comme un bruit additif à distribution normale.

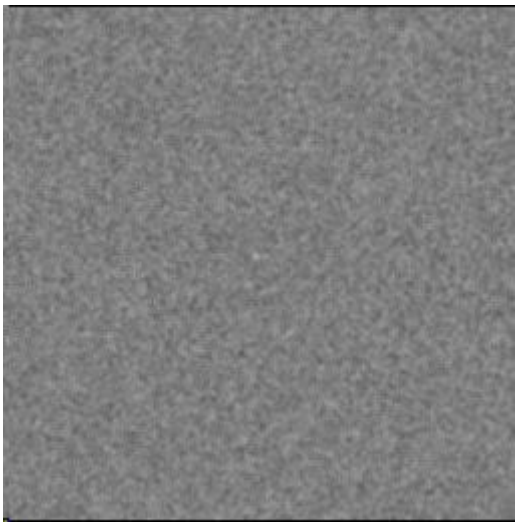

Figure 9 - Image I_4 bruitée et filtrée avec H1

PSNR(I_4 , I_4 bruitée et filtrée) = 29.5981 dB

On remarque clairement que cette image n'a rien à voir avec l'image de départ I_4 . On peut déduire de ce résultat que les filtres moyenneurs n'ont pas beaucoup d'effet sur le bruit additif à distribution normale.

Filtres moyenneurs sur une image naturelle

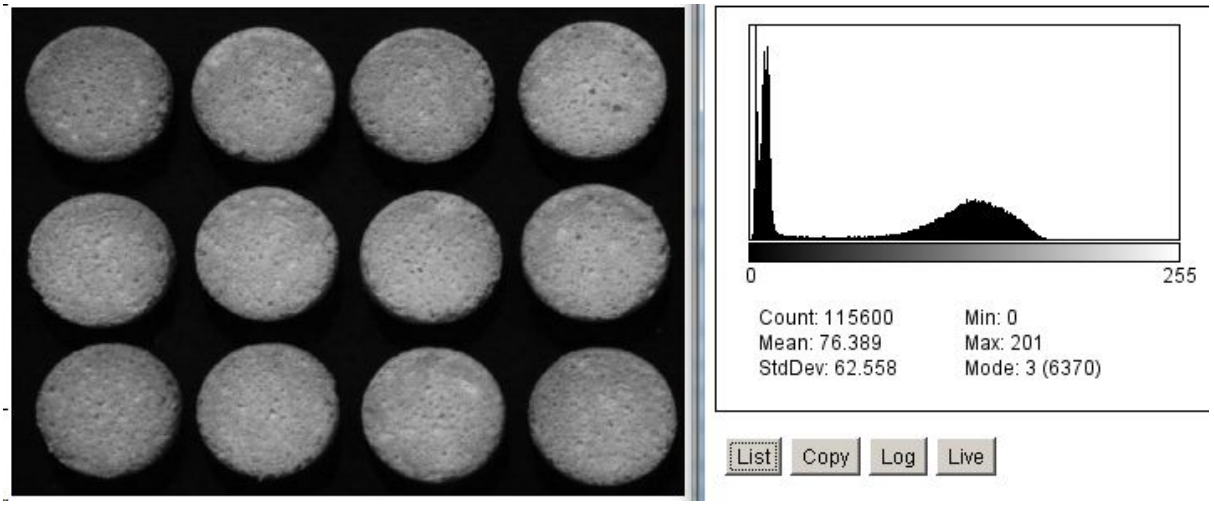

Figure 10 - Image gateaux1.png et son histogramme

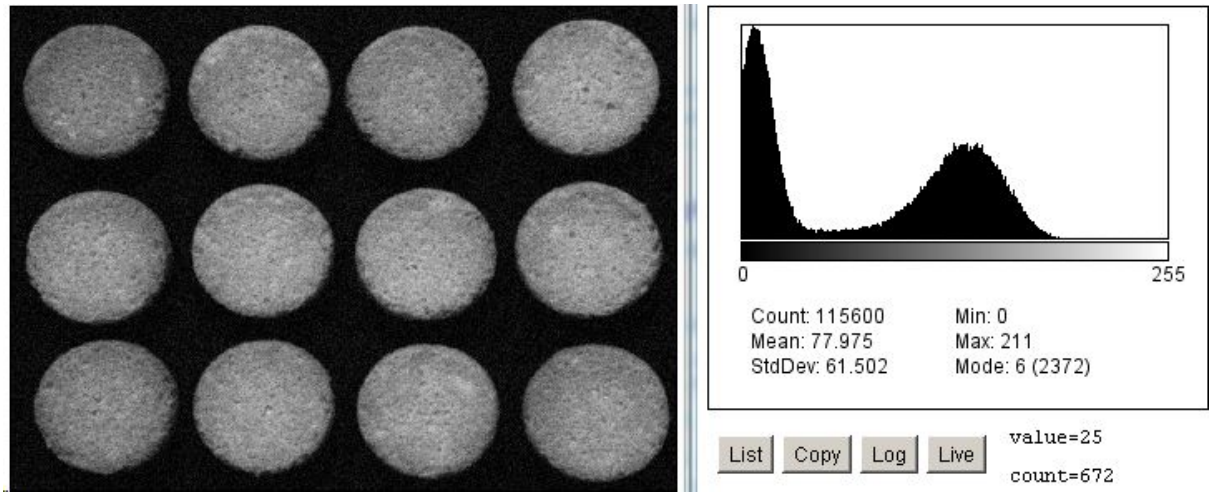

Figure 11 - Image gateaux1.png bruitée avec un bruit Gaussien d'écart-type σ=10

L'image *gateaux1.png* est bruitée avec un bruit Gaussien d'écart type *σ=10* . On peut noter que l'histogramme de l'image bruitée ressemble à celui de l'image d'origine. En effet celui-ci possède les mêmes variations que l'histogramme d'origine mais de manière plus importante.

PSNR(gateaux1.png,gateaux1.png bruitée Gaussien) = 28.6462 dB.

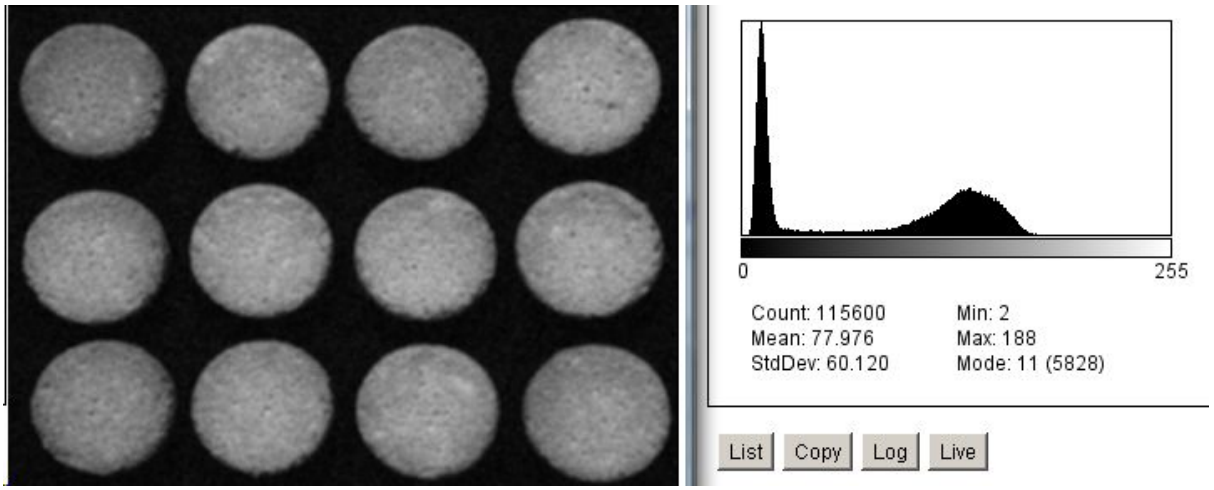

Figure 12 - Image gateaux1.png bruitée avec le filtre moyenneur 3x3 (H1 normalisé) et son histogramme.

PSNR(gateaux1.png,gateaux1 bruitée et filtrée 3x3) = 31.3983 dB

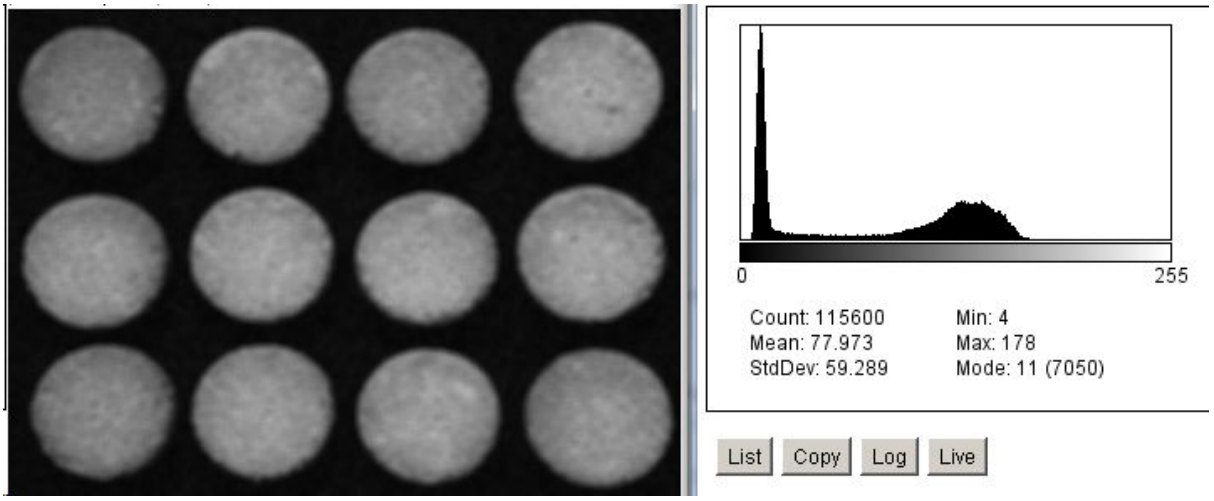

Figure 13 - Image gateaux1.png bruitée avec le filtre moyenneur 5x5 normalisé

PSNR(gateaux1.png,gateaux1.png bruitée avec le filtre moyenneur 5x5 normalisé) = 28.8783 dB

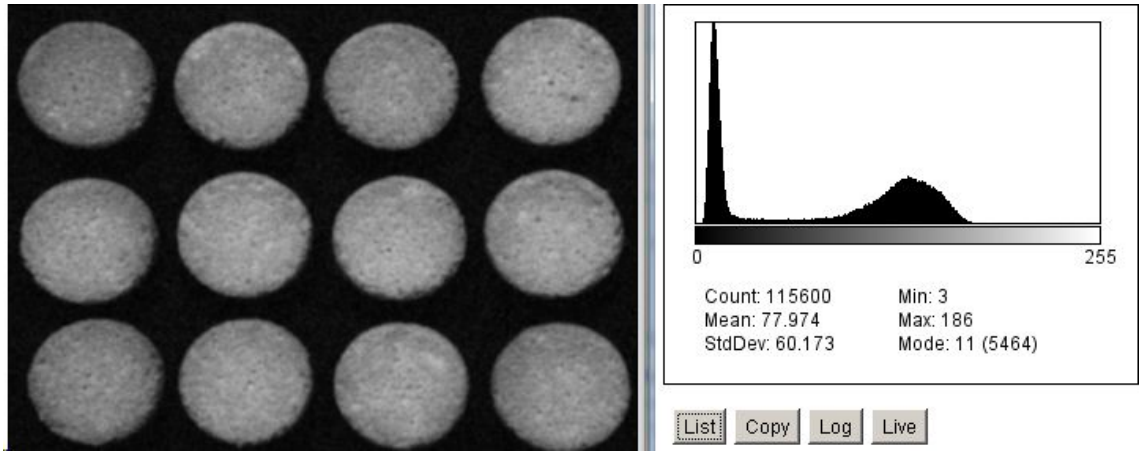

Figure 14 - Image gateaux1.png bruitée avec un filtre Gaussien d'écart-type σ=0.08

PSNR(gateaux1.png,gateaux1.png bruitée avec un filtre Gaussien d'écart-type σ=0.0*8*) = 32.2004 dB

Les histogrammes ci-dessus sont semblables cependant on remarque un décalage des courbes.

Bruit impulsionnel et filtre médian

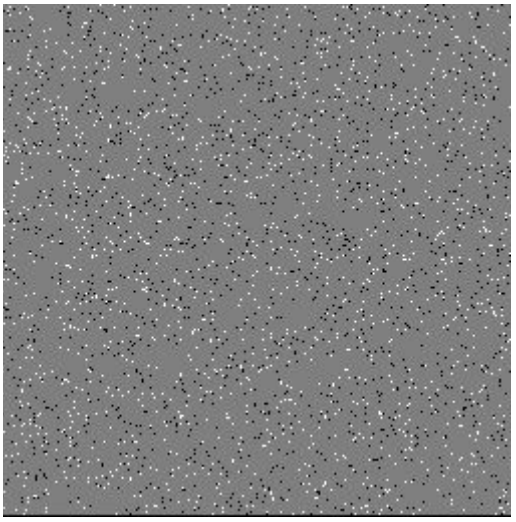

Figure 15 - Image I_4 bruitée

La figure 15 nous montre l'image I_4 bruitée. Visuellement, nous pouvons déduire de cette figure qu'il s'agit d'un bruit impulsionnel de type "poivre et sel". On remarque l'affectation de 3205 pixels sur l'image bruitée. Ce qui correspond à un pourcentage de 4,8%.

PSNR(I_4 , I_4 salt and pepper) = 19.1271 dB

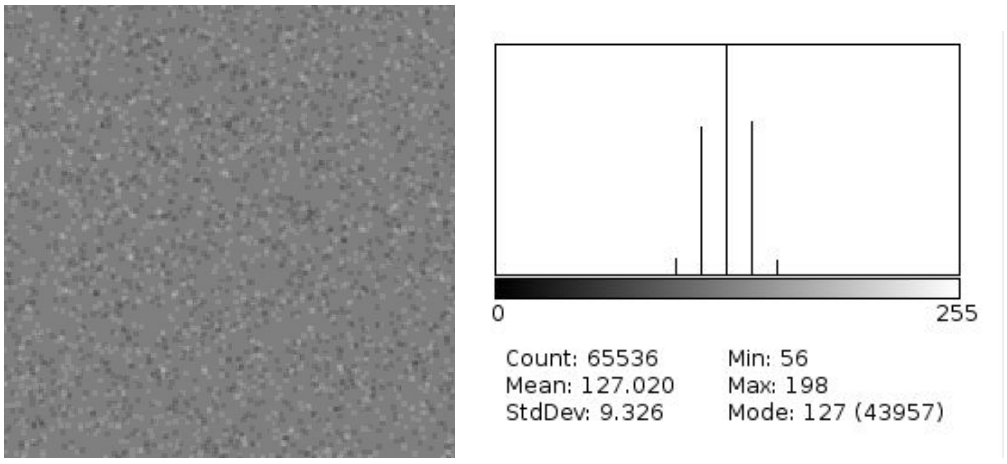

Figure 16 - Image I_4 bruitée et filtrée avec un moyenneur 3x3 et son histogramme

Visuellement l'image de la figure 16 ne ressemble pas à l'image Image I_4 d'origine. On peut de plus constater que l'histogramme de cette image n'est pas semblable à celui de l'image I_4 .

Nous avons traité le cas d'un pixel blanc. L'application d'un filtre moyenneur 3x3 nous renvoie une valeur de niveau de 141 pour ce pixel alors que dans l'image d'origine il avait pour valeur 127.

Moyenneur H1 :

127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	141	141	141	127	127	127	127
127	141	141	141	141	141	141	127	127	127	127
127	141	141	141	141	141	141	127	127	127	127
127	141	141	141	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	141	141	141	127	127	127	127	127	127

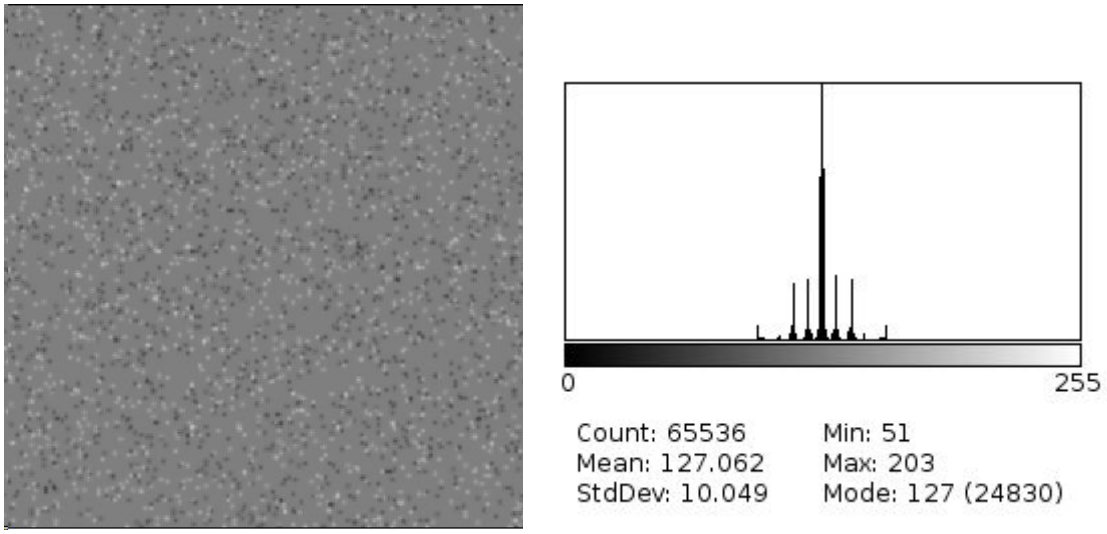

Figure 17 - Image I_4 bruitée et filtrée avec un filtre gaussien (0.8) et histogramme

On remarque également que le filtre gaussien ne produit pas un bon résultat car l'image produite ne se rapproche pas de l'image I_4. En effet ce filtre produit la valeur 159 pour le même pixel blanc que précédemment.

Filtre Gaussien (0.8)

127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	127	127	127	127	127	127	127
127	127	127	127	128	128	128	127	127	127	127
127	128	128	128	134	142	134	128	127	127	127
128	134	142	135	142	159	142	128	127	127	127
128	142	159	142	135	142	134	128	127	127	127
128	134	142	134	128	128	128	127	127	127	127
127	128	128	128	127	127	127	127	127	127	127
127	127	128	128	128	127	127	127	127	127	127
127	128	134	142	134	128	127	128	128	128	127

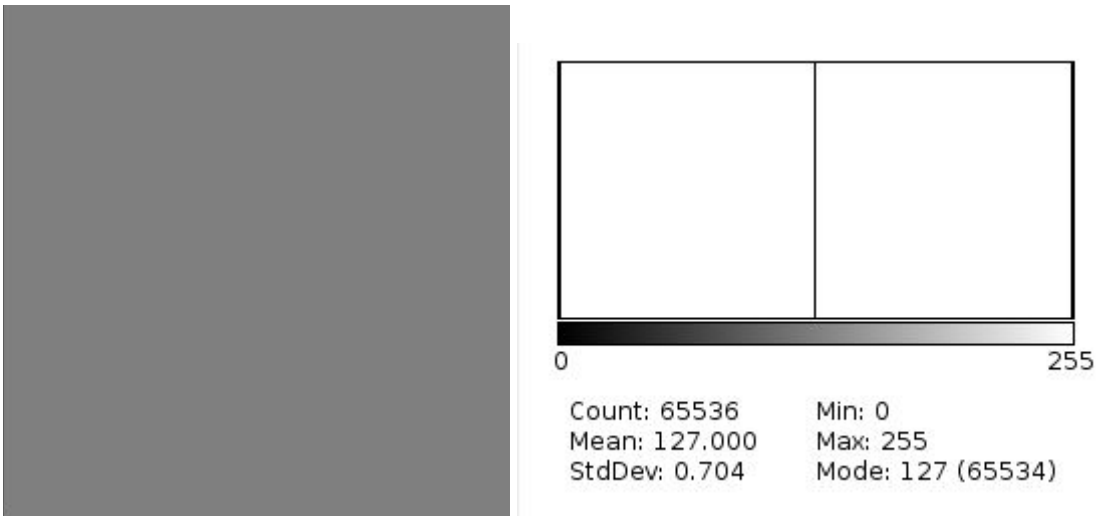

Figure 18 - Image I_4 avec un filtre médian de rayon 1 et son histogramme

L'application du filtre médian sur l'image I_4 avec un bruit de type "sel et poivre" produit un résultat satisfaisant puisque l'image obtenue ressemble beaucoup à l'image de départ.
PSNR(I_4 , I_4 filtre médian) = 19.1271 dB
Lorsqu'on calcule le nombre de pixels affectés, on trouve 0 ce qui signifie que tous les pixels issus du bruit ont été corrigés par le filtre.

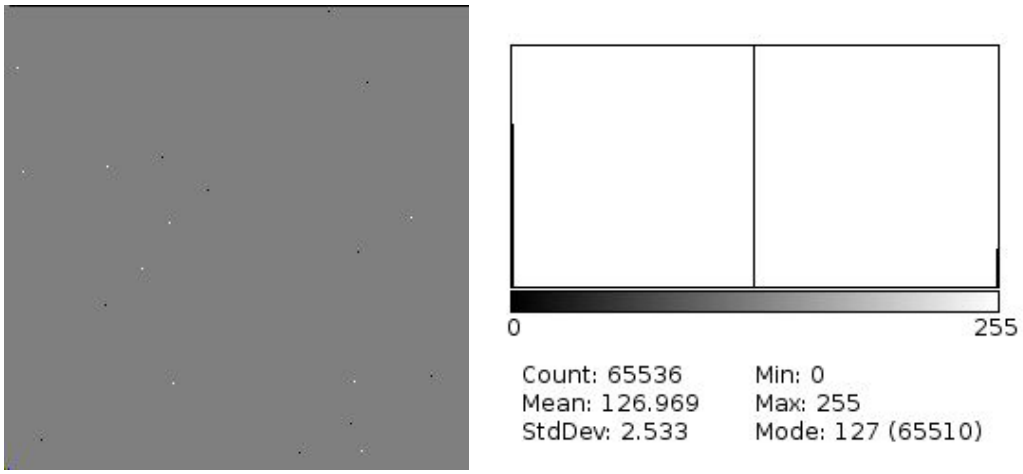

Figure 19 - Image Filtre I_4 avec un filtre médian de rayon 0.5 et son histogramme

Sur cette image on trouve 22 pixels affectés, ce qui signifie que tout le bruit n'a pas été corrigé. Le PSNR(I_4 , I_4 filtre médian) est égale 40.7518 dB.

Nous souhaitons ensuite comparer l'efficacité des filtres moyennant et médian sur l'image gateaux1.

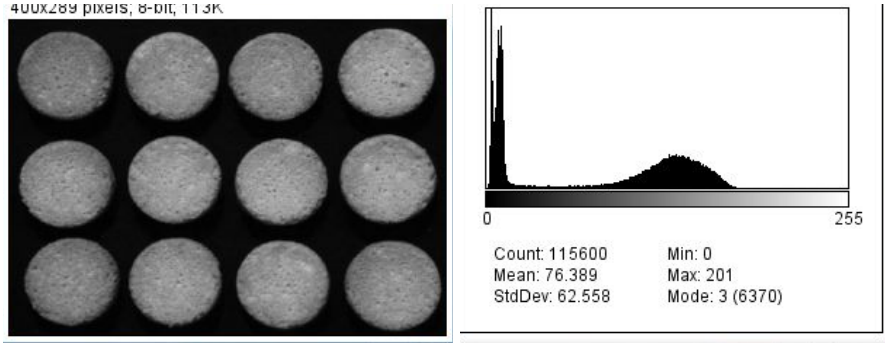

Figure 20 - Image gateaux1 et son histogramme

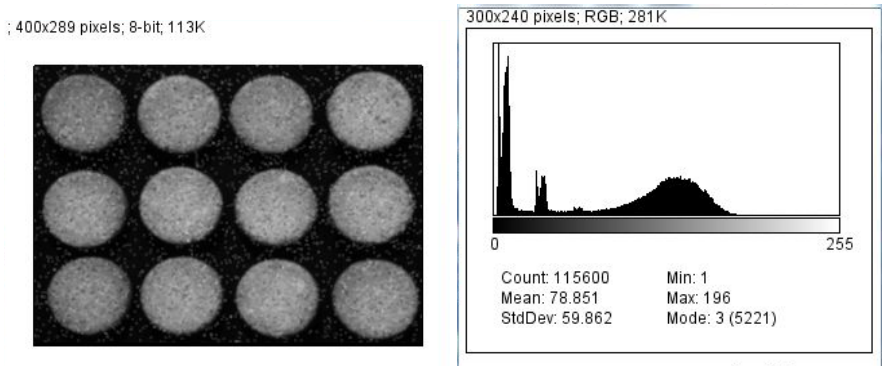

*Figure 21 - Image gateaux1 avec un filtre moyenneur 3*3 et son histogramme*

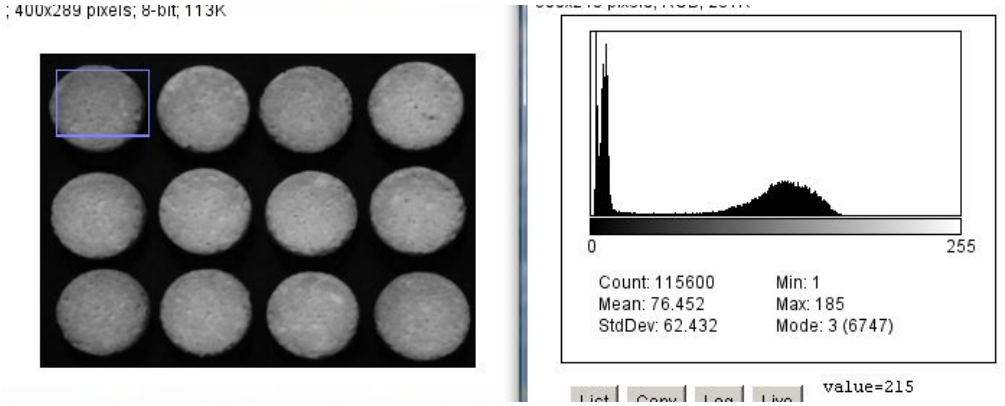

Figure 22 - Image gateaux1 avec un filtre médian et son histogramme

Comme on peut le constater le filtre médian est bien plus efficace que le filtre moyenneur. En effet, le filtre moyenneur créer de nouvelle valeur qui viennent bruiter l'image. On peut voir apparaître ces valeurs dans l'histogramme de la figure 21.

Codage d'un filtre

Nous avons implémenté une macro réalisant un filtre min/max afin de corriger différents bruits. La macro est disponible en annexe du document. Cette macro créer une nouvelle image vierge. Elle parcourt l'image à corriger. Pour chaque pixel on cherche les valeurs minimum et le maximum des pixels voisins de taille 3*3. Le pixel courant n'est pas pris en compte dans la recherche du minimum et du maximum. Si la valeur du pixel courant est comprise entre le minimum et le maximum il conserve sa valeur. Si la valeur courante est inférieur au minimum le pixel courant prends la valeur minimum. De la même manière, si la valeur courante est supérieur à la valeur maximum du voisinage elle prends sa valeur. Enfin le pixel courant est ajouté à l'image vierge créé pour ne pas influencer sur le parcours des pixels suivants. Afin de juger de l'efficacité de ce filtre nous l'avons appliqué à une même image bruité de différente manière.

Dans un premier temps nous avons réalisé le test sur l'image masqué par un bruit gaussien d'écart type 25.

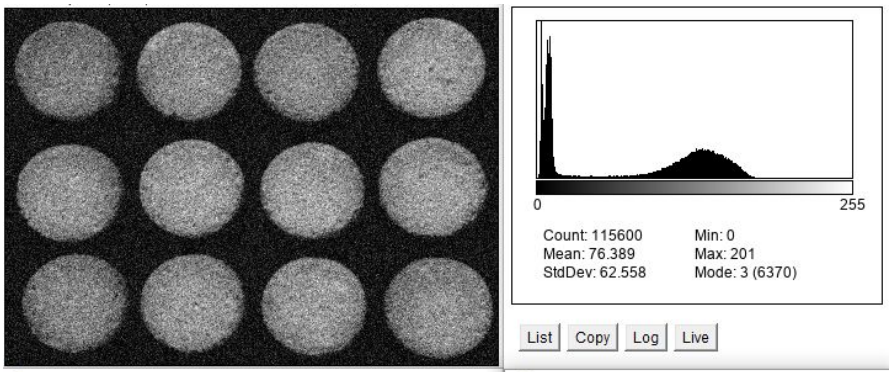

Figure 23 - Image gateaux1.png avec un bruit gaussien d'écart-type 25

On peut remarquer que les détails de l'image sont floutés, et que le fond n'est plus totalement noir. A l'oeil nue les contours restent satisfaisant mais il est possible que le bruit ai créé des imperfections sur le bord des gâteaux. Nous avons donc appliqué notre filtre min/max à l'image bruité.

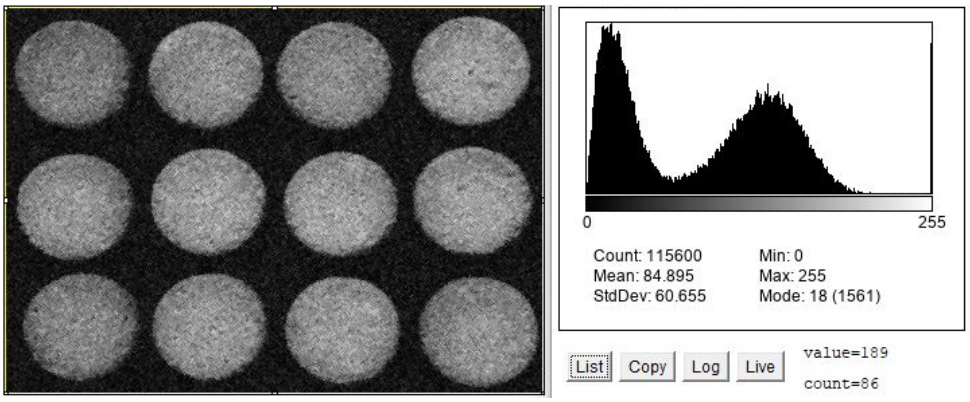

Figure 24 - Image gateaux1.png avec un bruit gaussien d'écart-type 25 puis filtré avec un filtre min/max

Nous pouvons voir une légère amélioration de l'image cependant le bruit gaussien est toujours présent. De plus le PSNR des deux images est égale à 19.1732 dB qui traduit une altération perceptible. Si le bruit ne peut pas être corriger par le filtre min/max c'est parce que les pixels formants le bruit sont trop proches l'un de l'autre s'atténuer entre eux, il y a de forte probabilité que la valeur du pixel courant soit comprise dans les valeurs des pixels voisins.

Nous avons ensuite appliqué un bruit poivre et sel à l'image gateaux1.

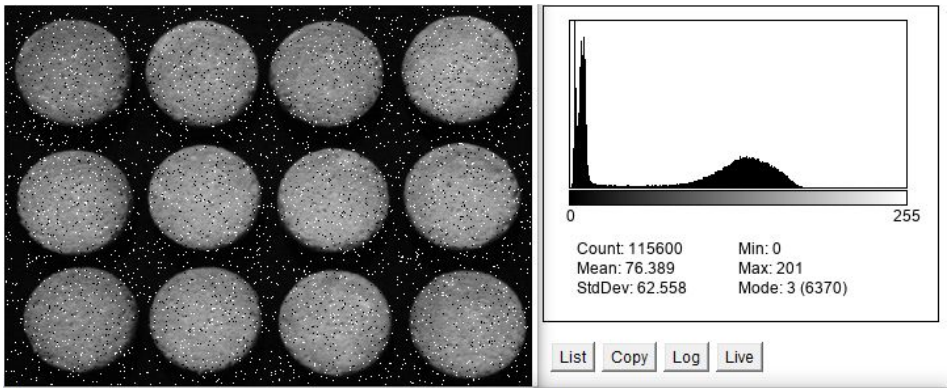

Figure 25 - Image gateaux1.png avec un bruit poivre et sel

Nous avons donc appliqué un filtre min/max sur l'image bruité en poivre et sel. L'image obtenu est bien plus exploitable que la précédente. En effet le PSNR vaut 15.2512 dB qui traduit une modification perceptible de l'image. Certaines zones sont toujours affectés par le bruit, cependant l'image semble nette et les contours exploitables.

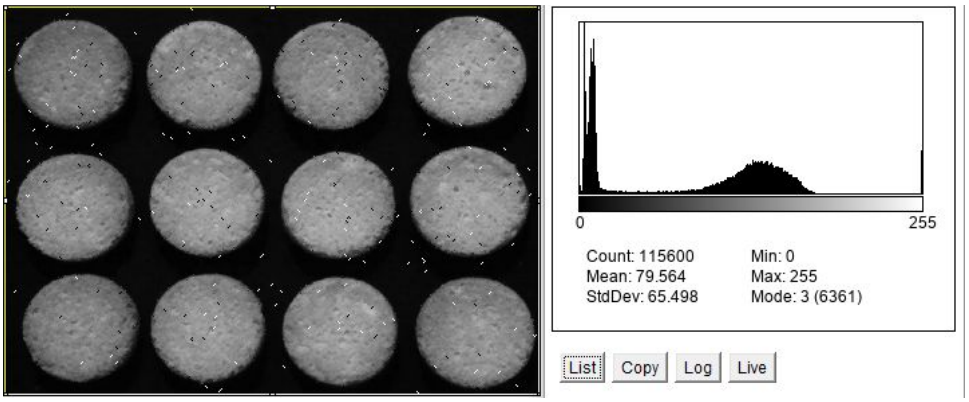

Figure 26 - Image gateaux1.png avec un bruit poivre et sel filtré avec un filtre min/max

Les pixels restés blancs ou noirs sont du au pixels voisins fixant les minimums ou maximum à 0 ou 255. C'est pourquoi on peut remarquer qu'ils sont toujours par deux dans une zone 3*3 autour d'un pixel.

Conclusion :

Tout au long de ce TP nous avons expérimenté différents altération d'image. Nous avons découvert et manipulés des filtres, comme le filtre moyenneur, le filtre médian ou encore le filtre min/max. Nous avons remarqué que chaque filtre n'agit pas de la même manière en fonction du bruit appliqués à l'image. C'est pourquoi il est intéressant de comprendre et de savoir identifier un bruit ou une altération de l'image pour savoir quel type de filtre appliqué pour obtenir l'image la plus exploitable possible.

Annexe :

```
setBatchMode(true); //images are hidden during macro execution

idImg = getImageID();

W = getWidth();
H = getHeight();

//Traçage d'une image toute noir qui sera l'image d'entrée filtré
newImage("filtre", "8-bit", W,H, 1);
idFiltre = getImageID();
setColor(255);
makeRectangle(0,0,W,H);
fill();

tailleFiltre = 3;
selectImage(idImg);

for(i = 1; i < W-1; i++){
      for(j = 1; j < H - 1; j++){
            pix = getPixel(i,j);
            min = 255;
            max = 0;

            for(x = 0; x < tailleFiltre; x++){
                  for(y = 0; y < tailleFiltre; y ++){
                        if((x != 1) && (y != 1)){
                              courant = getPixel(x-1+i, y-1+j);
                              if(courant > max) max = courant;
                              if(courant < min) min = courant;
                        }
                  }
            }

            if(pix > max) pix = max;
            if(pix < min) pix = min;
            selectImage(idFiltre);
            setPixel(i,j, pix);
            selectImage(idImg);
      }
}


setBatchMode("exit and display"); //show images
```