



Université
de Lille
1 SCIENCES
ET TECHNOLOGIES

Université de Lille
MASTER 1 INFORMATIQUE
Deuxième semestre



Traitement d'image

TP n°07

BARCHID Sami

CARTON Floriane

Introduction

La matière vue dans ce TP traite des masques de convolution, de la problématique de l'effet de bord, des filtrages et des différents types de bruit qui peuvent s'appliquer sur une image.

Ce rapport de TP est divisé en trois parties :

- **Masque de convolution** : analyse et étude du comportement d'un masque de convolution afin de comprendre divers aspects liés à la définition de filtres linéaires simples (normalisation, taille et effets de bords).
- **Bruits et débruitage de l'image** : tests, analyses et interprétations sur la faculté de différents filtres à atténuer différents types de bruit.
- **Implémentation d'un filtrage min-max** : implémentation d'un filtrage min-max et analyse de sa faculté à atténuer certains types de bruit.

Masque de convolution

Le but de cette partie du TP est de comprendre les divers aspects liés à la notion de filtre linéaires simples (les moyenneurs) en étudiant le comportement du filtrage d'une image via un masque de convolution.

Nous sommes ainsi dans le contexte d'une transformation locale d'une image puisque, pour calculer la valeur d'un pixel $I(x, y)$ dans l'image transformée, nous tenons compte de la valeur de ce pixel mais aussi de son voisinage.

Filtre moyeneur non-normalisé

Nous allons appliquer ici le filtre de convolution **non-normalisé** de taille 3×3 suivant :

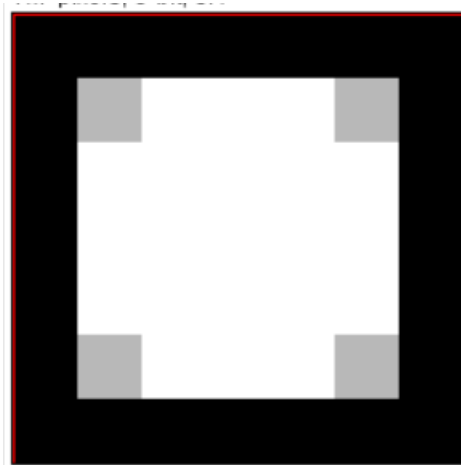
$$H_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

L'image sur laquelle sera appliqué ce filtre est l'image de synthèse « I2 » suivante :



Prefs	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	180	180	0	0
3	0	0	180	180	180	0	0
4	0	0	180	180	180	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

L'image résultante est alors la suivante :



Prefs	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	180	255	255	255	180	0
2	0	255	255	255	255	255	0
3	0	255	255	255	255	255	0
4	0	255	255	255	255	255	0
5	0	180	255	255	255	180	0
6	0	0	0	0	0	0	0

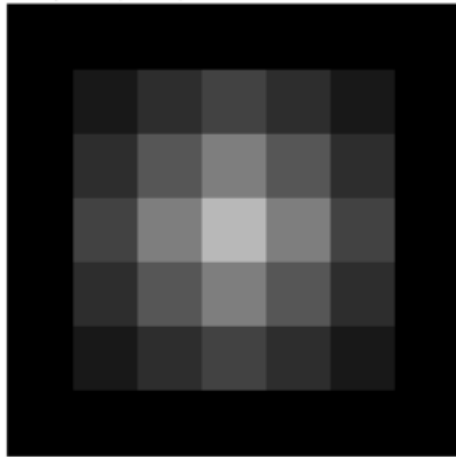
On remarque que les niveaux de gris de l'image filtrée sont largement plus grands que sur l'image d'origine. En effet, un filtrage linéaire non-normalisé dont tous les coefficients sont à 1 va simplement effectuer la somme des 9 pixels dans un voisinage 3×3 . Or, comme les niveaux de gris des pixels au centre de l'image i2 sont élevés (180), nous arrivons rapidement au niveau de gris maximal. Ce qui explique que nous obtenons un carré blanc plus gros.

Filtre moyennneur normalisé

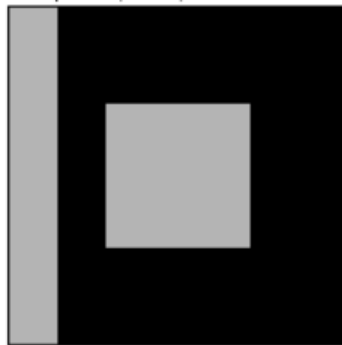
Nous appliquons le même filtre de convolution que celui vu précédemment, mais **normalisé**. C'est-à-dire que nous devons modifier le filtre de convolution de sorte que

$S = \text{la somme des coefficients du filtre} = 1$. Comme tous les coefficients sont positifs, nous devons simplement diviser chaque coefficient par S . Pour ce filtre, on obtient $S = 9$, donc le filtre de convolution normalisé sera une matrice 3×3 où tous les coefficients sont à $1/9$.

L'image ainsi filtrée obtenue est la suivante :



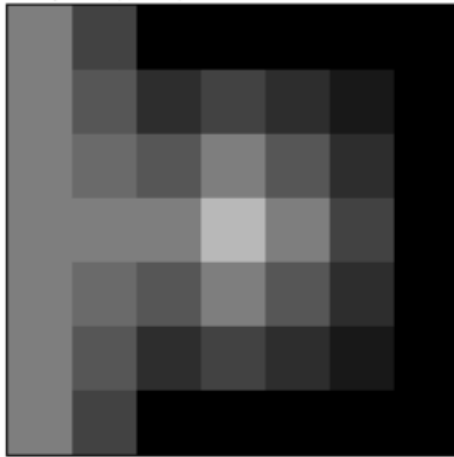
On remarque ici que la luminance de l'image d'origine est conservée dans l'image filtrée, ce qui est bien une propriété d'un masque de convolution normalisé.



Effets de bord

Nous allons ici appliquer le filtre de convolution normalisé vu précédemment sur une image « /3 », qui comporte un bord colorié.

L'image filtrée qui en résulte est la suivante :



Ce qui nous intéresse ici est la valeur des niveaux de gris obtenus après filtrage sur le bord de l'image pour pouvoir identifier la stratégie qu'utilise notre logiciel de transformation d'image (ici, **ImageJ**).

Nous pouvons comparer les stratégies :

- **Pixels laissés inchangés** : ce n'est pas possible ici car les niveaux de gris des pixels du bord transformé ne sont pas égaux aux niveaux de gris des pixels du bord d'origine.
- **Pixels extérieurs = 0** : pas possible non plus car les valeurs des niveaux de gris du bord transformée sont trop élevées.
- **Image enroulée** : la stratégie de l'image enroulée aurait donné un résultat similaire à la stratégie des pixels extérieurs dans le cas de notre image, ce n'est donc pas possible non plus.
- **Image miroir** : en simulant les calculs pour le bord de l'image, on remarque que c'est bien la stratégie qui a été utilisée.

La stratégie d'effet de bord est donc **l'image miroir**.

Bruits et débruitage de l'image

Le but de cette partie du TP est de tester et de comprendre la faculté de différents filtres à atténuer du bruit.

Pour chaque test, nous aurons en entrée :

- Une image originale non bruitée
- Une image bruitée
- Un filtre à appliquer sur l'image bruitée

Nous effectuerons deux actions sur ces données en entrée :

- **Comparer l'image originale non-bruitée avec l'image bruitée** : le but est d'identifier le type de bruit qui a été appliqué sur l'image.
- **Comparer l'image originale non-bruitée avec l'image débruitée** : le but est de conclure sur l'efficacité du filtre utilisé à réduire le bruit présent dans l'image.

Ces comparaisons se feront grâce à l'utilisation de trois indicateurs :

- L'oeil humain.
- Les informations issues de l'histogramme de l'image transformée (bruitée ou débruitée).
- Le PSNR de l'image transformée (bruitée ou débruitée) avec l'image originale non-bruitée.

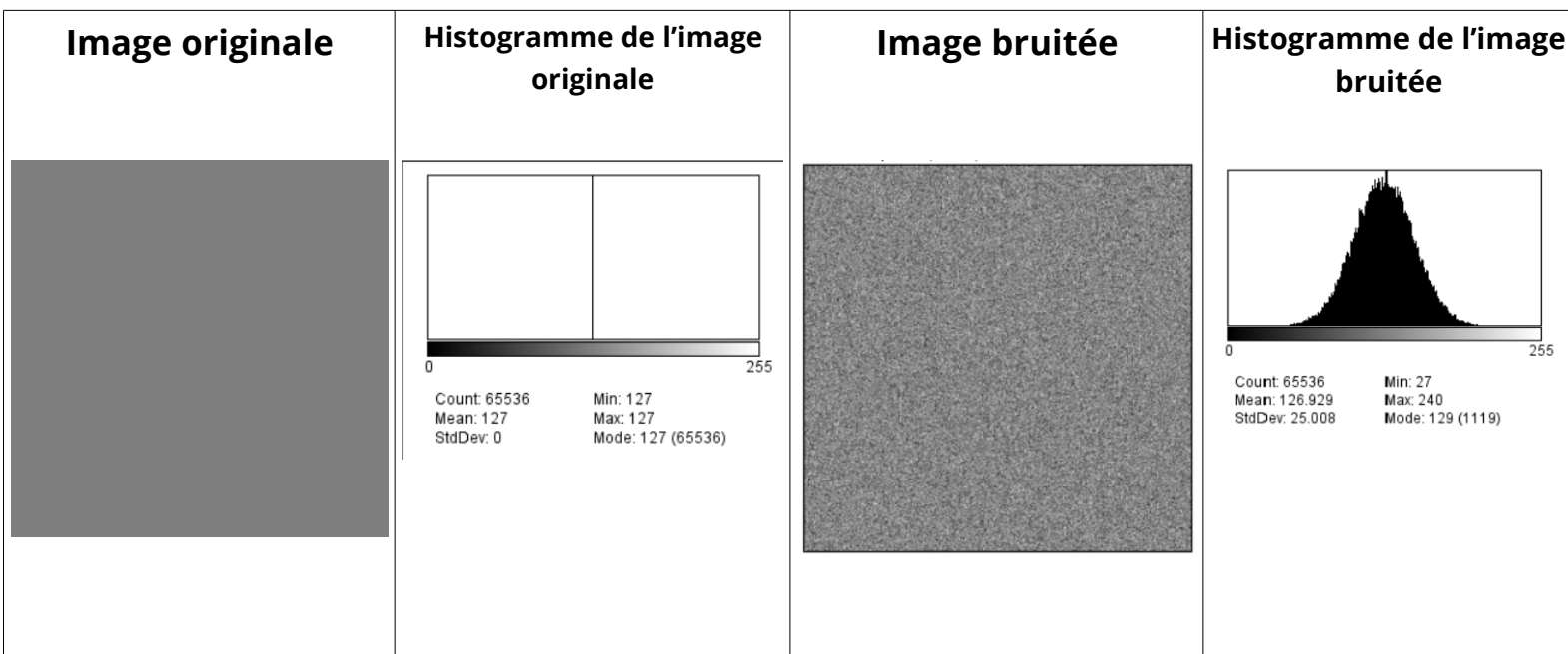
Bruit Gaussien d'une image synthétique

Nous appliquons ici un bruit gaussien sur l'image synthétique « I4 » suivante :



FIGURE – IMAGE I4

L'application du bruit gaussien sur « I4 » donne le résultat suivant :

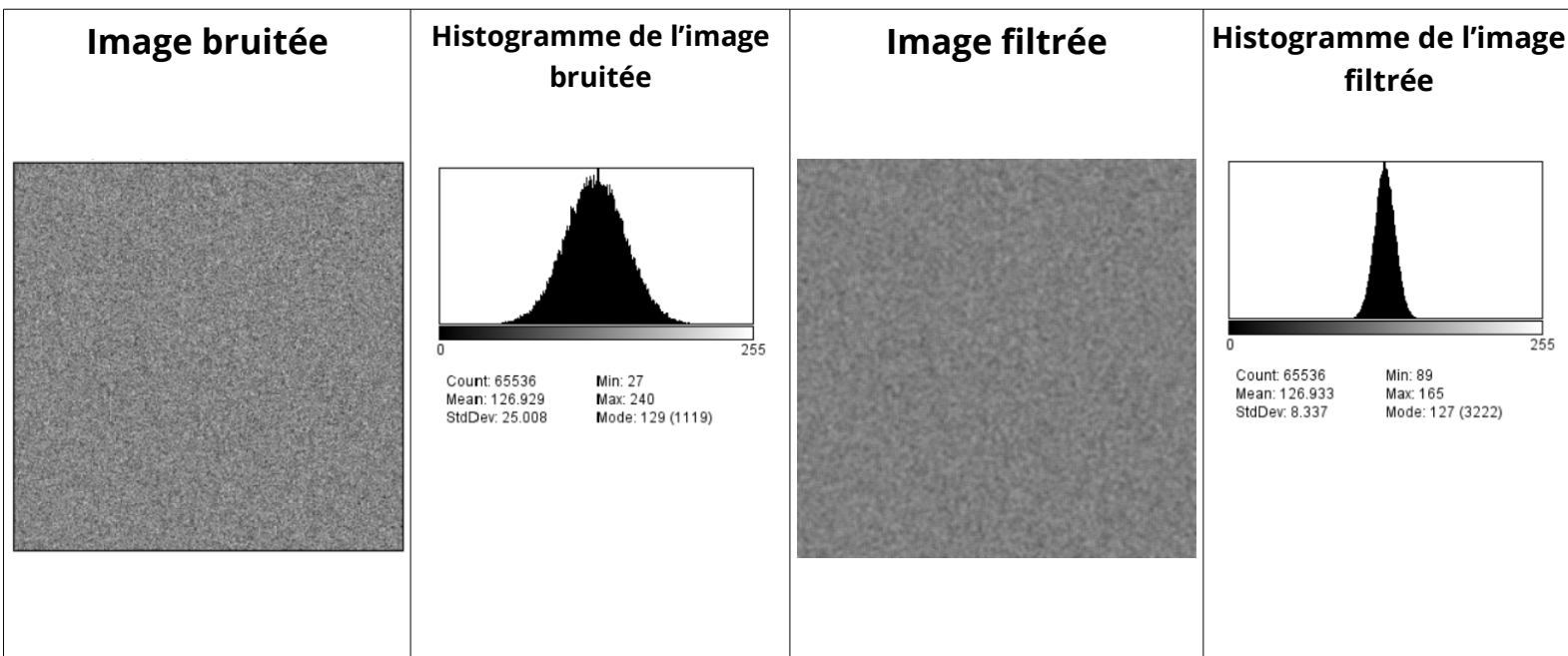


On remarque clairement ici que la distribution du bruit est gaussienne. De plus, nous remarquons que le bruit est additif. Nous pouvons donc

conclure que le type de bruit appliqué sur l'image est un **bruit d'amplification**.

Le **PSNR** entre l'originale et l'image bruitée est de **20.1996 Db**.

Nous allons appliquer alors le filtre H1 utilisé précédemment :



PSNR entre l'image filtrée et l'originale : **29.7109**.

L'application du filtre moyennneur H1 a permis de réduire l'écart-type de l'image bruitée (et donc a permis de réduire la puissance du bruit de l'image). On observe aussi que les valeurs des niveaux de gris des pixels se concentrent vers la moyenne.

En regardant à l'oeil nu, on remarque que l'image paraît plus floue qu'auparavant, ce qui a permis de lisser les pixels bruités (en d'autres termes, cela a permis de faire une atténuation du bruit).

Le PSNR nous montre également que l'image ressemble plus à l'originale (et donc que la puissance du bruit a été réduite) même si l'image ne ressemble plus réellement à l'originale.

Application de filtres moyennneurs sur une image naturelle

Nous appliquerons ici différents filtres sur une image naturelle suivante (nommée « cake ») pour comparer leur comportement :

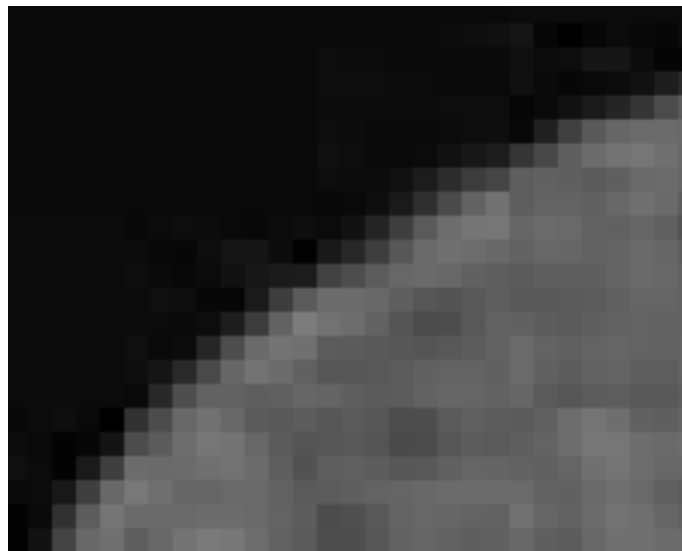
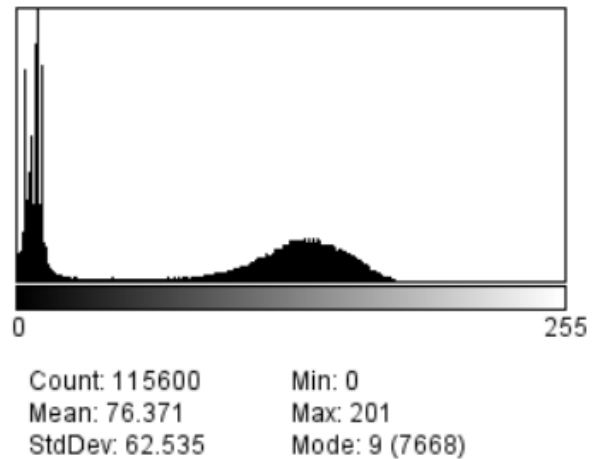
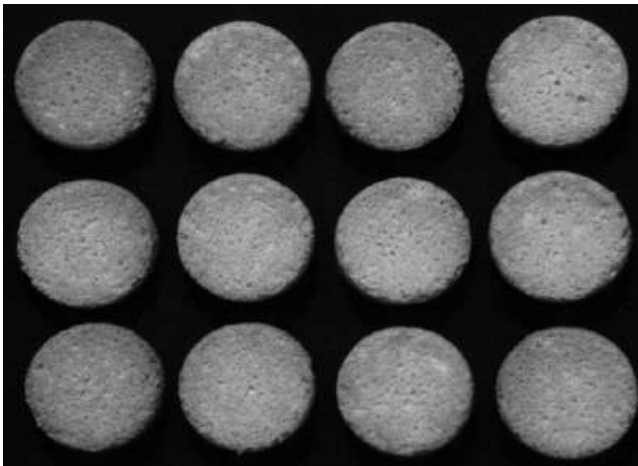
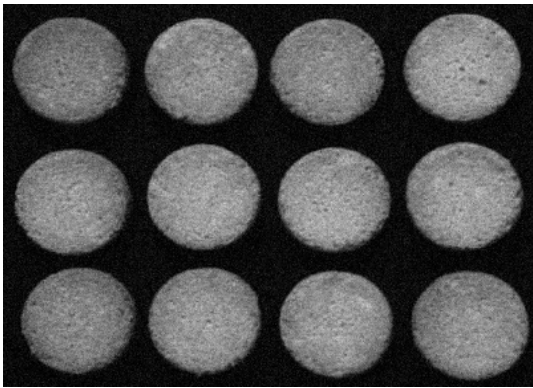
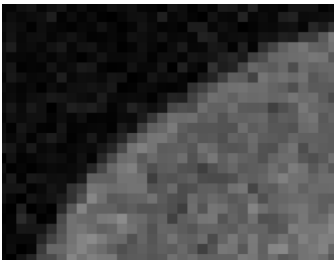
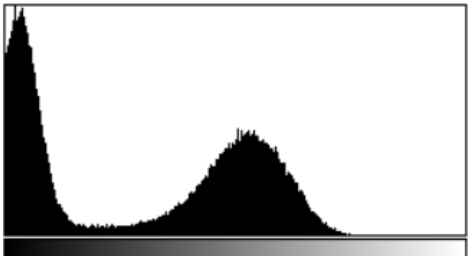


FIGURE – cake, son histogramme et un zoom

Note : le zoom sur le coin supérieur gauche de l'image des cake sert à mieux discerner les changements opérés

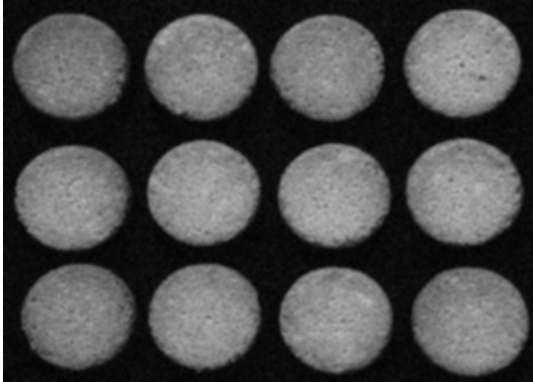
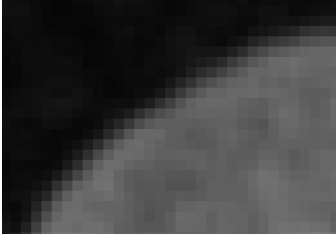
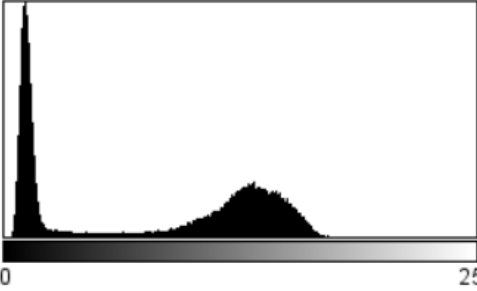
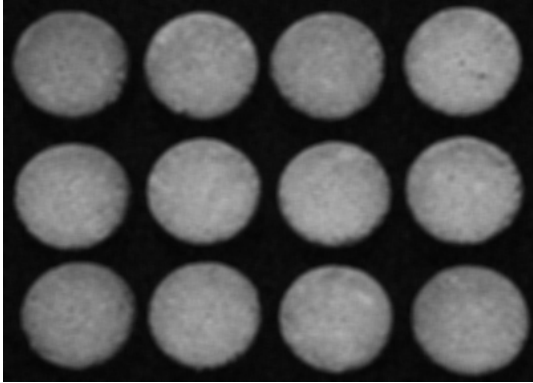
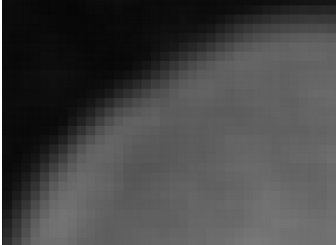

Nous appliquerons alors un bruit gaussien d'écart-type -10 pour obtenir l'image bruitée suivante :

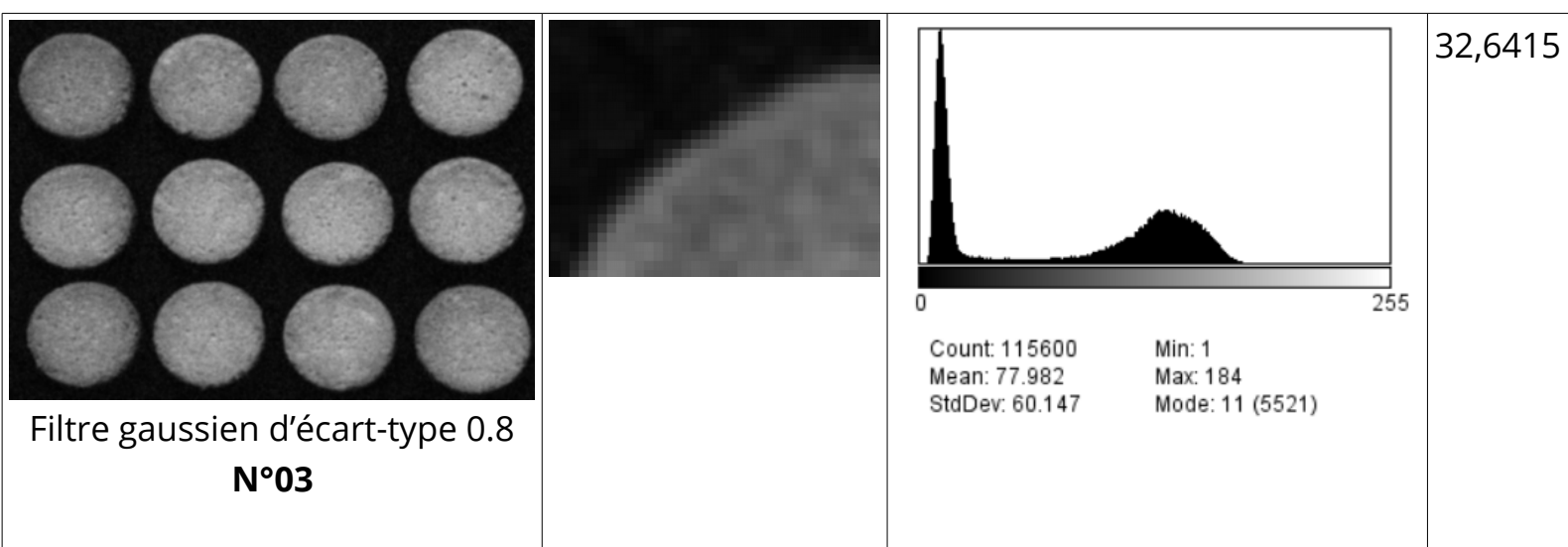
Image bruitée	Zoom	Histogramme de l'image bruitée	PSNR (en Db)
		 <p>Count: 115600 Min: 0 Mean: 77.984 Max: 209 StdDev: 61.428 Mode: 5 (2419)</p>	28,6425

Premièrement, on remarque que les contours paraissent plus nettes après application de ce bruit gaussien.

En ce qui concerne la modification au niveau de l'histogramme, nous observons que la répartition des modes présents dans l'histogramme a augmenté tandis que leur amplitude s'est réduite.

Nous allons maintenant appliquer différents filtres de lissage linéaire à l'image bruitée :

Image filtrée	Zoom	Histogramme de l'image filtrée	PSNR (en Db)
 <p>Filtre moyennneur H1 normalisé N°01</p>		 <p>Count: 115600 Min: 2 Mean: 77.984 Max: 186 StdDev: 60.095 Mode: 11 (5665)</p>	32.0044
 <p>Filtre moyennneur 5x5 normalisé avec tout coefficient à 1 N°02</p>		 <p>Count: 115600 Min: 3 Mean: 77.984 Max: 179 StdDev: 59.259 Mode: 11 (7218)</p>	29.154



À l'oeil nu, on remarque que, en général, les images filtrées sont plus floues que l'originale. En comparant les images filtrées au niveau du flou, on se rend compte que :

- L'image filtrée par le filtre moyennneur 5x5 est la plus floue : le PSNR est le moins élevé et à l'oeil nu, c'est l'image la moins nette. On observe même que certaines irrégularités des cakes disparaissent quasiment. On remarque aussi fortement les inconvénients liés au filtre moyennneur, à savoir :
 - Les contours sont fortement atténués
 - Les pixels aberrants isolés influencent fortement le résultat de l'image filtrée.
- Les problèmes vus pour le filtre moyennneur 5x5 sont aussi observables sur l'image filtrée par le filtre H1 normalisé mais sont moins marqués. On peut donc remarquer que le nombre de coefficients du filtre de convolution influe sur la transformation et lisse l'image de manière plus importante.

- Le filtre gaussien d'écart-type 0.8 a mieux réussi à conserver les contours de l'image. Le PSNR est le plus élevé des trois filtres et à l'oeil nu, on remarque aussi que le rendu de l'image filtrée est convaincant.

D'autre part, en ce qui concerne les histogrammes des images filtrées, on note que la répartition de les modes des histogrammes augmente pour les images filtrées sont plus floues.

Bruit impulsif et filtre médian

L'image que nous allons bruite pour ce sous-chapitre est l'image « I4 », que nous allons bruite au moyen d'un bruit poivre et sel. Nous allons aussi prendre en exemple un pixel P et suivre les modifications subites par ce pixel suivant les filtres appliqués à I4 bruité.

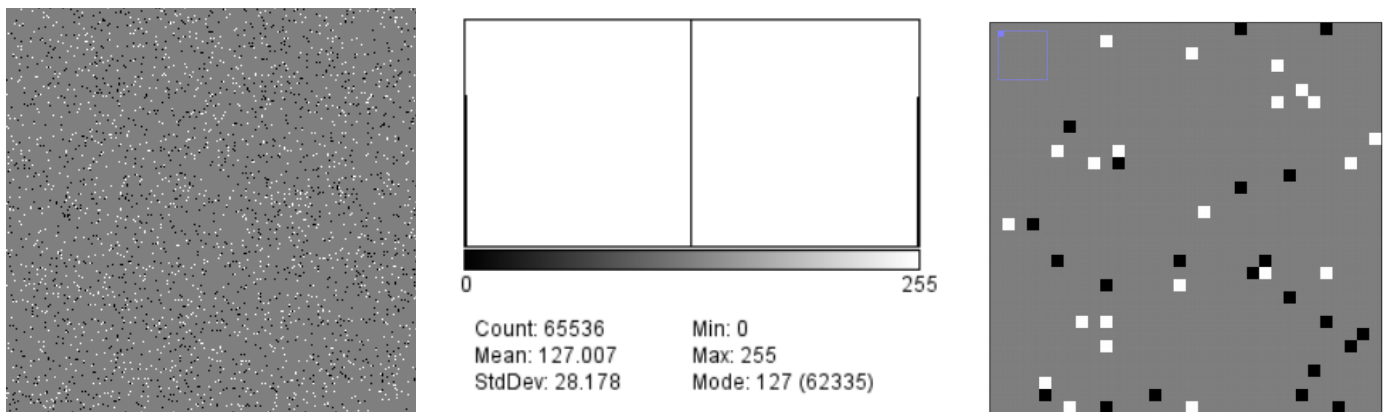


FIGURE – I4 avec bruit poivre et sel, histogramme et zoom sur le coin supérieur droit

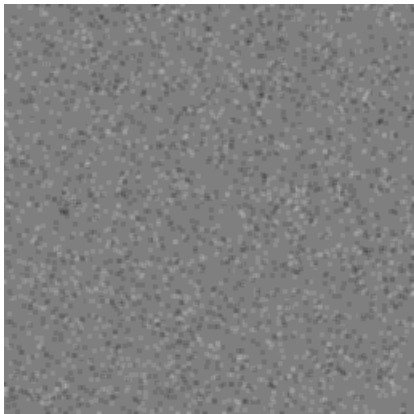
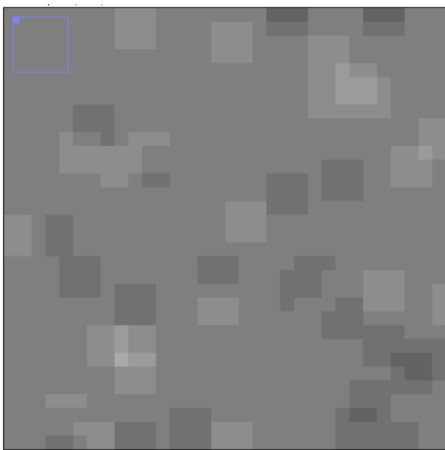
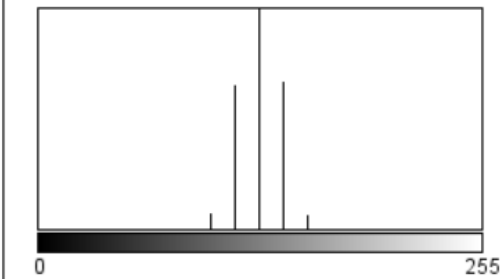
Le **PSNR** entre I4 et I4 bruitée est de **19.1326**.

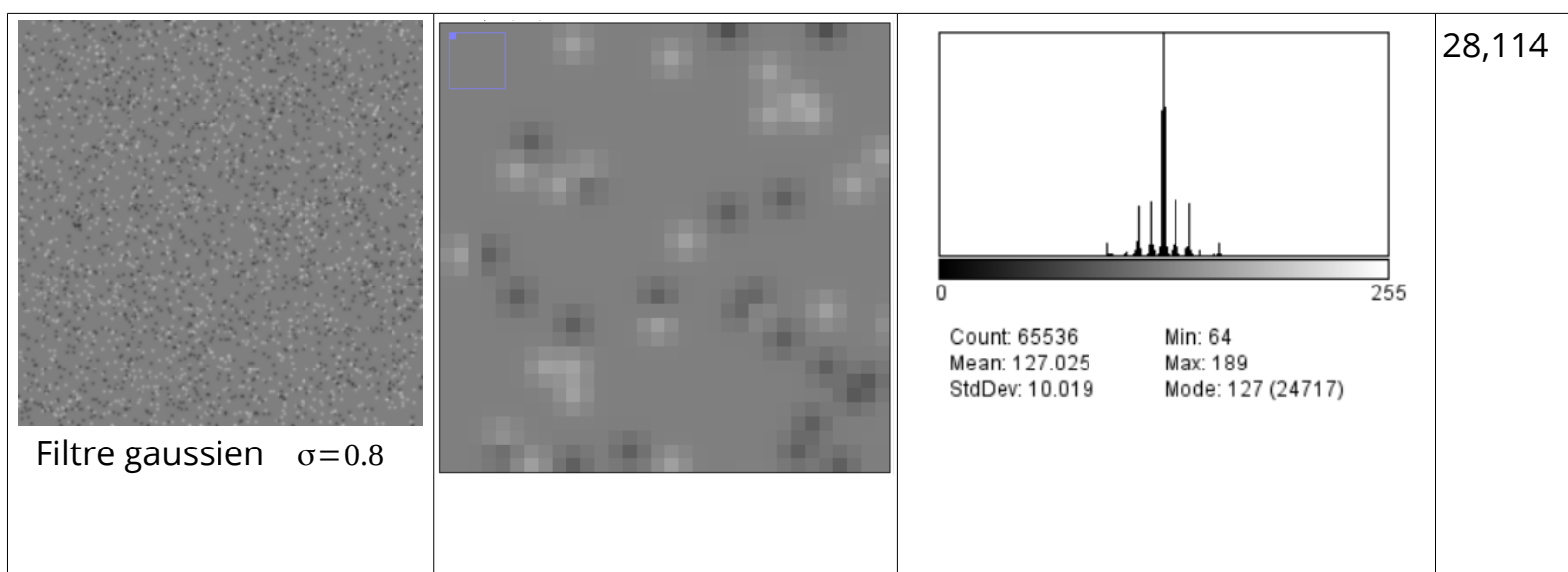
Ce bruit poivre et sel (nommé aussi « bruit impulsionnel ») a remplacé un certain pourcentage de pixels par des pixels noirs ou blancs (0 ou 255). Nous pouvons calculer le pourcentage de pixels transformés grâce à l'histogramme.

L'histogramme nous donne un nombre de pixels blancs $P_b = 1596$ et un nombre de pixels noirs $P_n = 1605$. On obtient donc un nombre de pixels transformés de 3201. Sachant que le nombre de pixels total est de 65536, on obtient alors un **pourcentage de pixels transformés par le bruit de $\pm 5\%$** .

Comportement des filtres de lissage linéaire

Nous allons utiliser ici le filtrage gaussien ($\sigma = 8$) et le filtre moyenneur H1 normalisé sur l'image bruitée I4.

Image filtrée	Zoom	Histogramme	PSNR En dB
 <p>Filtre moyenneur H1 normalisé</p>		 <p>Count: 65536 Mean: 126.984 StdDev: 9.304</p> <p>Min: 71 Max: 170 Mode: 127 (43871)</p>	28.7572



Pour les deux filtres de lissage linéaires, on remarque sur leurs histogrammes que les filtres ont essayé de répartir les niveaux de gris des pixels bruités sur les niveaux de gris avoisinant. Cela s'explique par le fait que les filtres de lissage opèrent un flou de l'image.


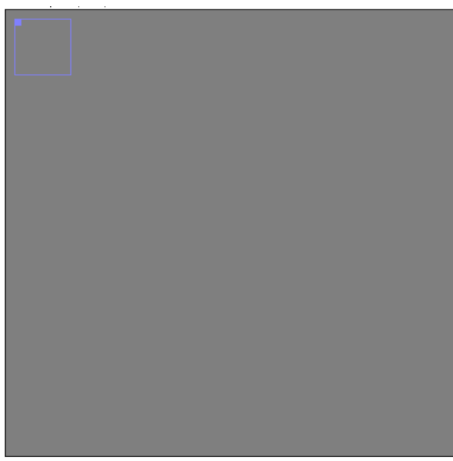
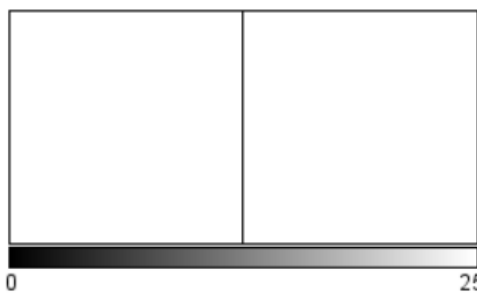
Lorsque l'on compare les filtres de lissage linéaire entre eux, on confirme bien que le filtre gaussien préserve mieux les contours que le filtre moyennneur H1 normalisé.

Toutefois, les deux filtres ne fournissent pas un résultat convaincant, étant donné qu'ils floutent uniquement sur un voisinage limité et que le pixel bruité est trop influent lors de l'application du lissage linéaire.

Comportement du filtre médian

Le filtre médian est un filtre de lissage non-linéaire (donc non réalisable par masque de convolution) qui consiste à donner au pixel calculé le niveau de gris médian des pixels de son voisinage.


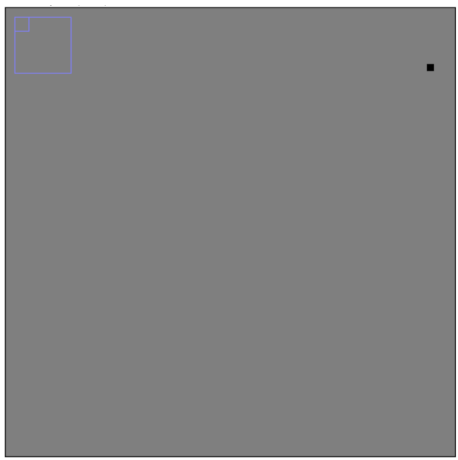

Nous avons donc appliqué un filtre médian de rayon 1. Les résultats sont les suivants :

Image filtrée	Zoom	Histogramme	PSNR En dB
 <p>Filtre médian de rayon 1</p>		 <p>Count: 65536 Min: 127 Mean: 127 Max: 127 StdDev: 0 Mode: 127 (65536)</p>	∞

On le voit facilement à l'oeil nu et même en calculant le PSNR en ∞dB :
L'image filtrée resultante est exactement la même que l'image originale.
Cela s'explique par le fait que chaque pixel bruité est placé de sorte que le niveau de gris médian d'un pixel et de son voisinage n'est pas celui d'un pixel bruité car ces pixels bruités sont trop espacés pour qu'il y en ait trop dans un même voisinage.

On peut alors se rendre compte que le filtrage médian marche particulièrement bien dans le cas d'un bruit poivre et sel.

Nous allons maintenant appliquer sur I4 un filtre médian de rayon 0.5 :

Image filtrée	Zoom	Histogramme	PSNR En dB
 <p>Filtre médian de rayon 0.5</p>		 <p>Count: 65536 Mean: 126.992 StdDev: 1.406</p> <p>Min: 0 Max: 255 Mode: 127 (65528)</p>	45,1715

Le filtre médian de rayon 0.5 n'a, pour sa part, pas réussi à entièrement supprimer le bruit impulsionnel. En effet, il reste encore quelques pixels noir et blanc ça et là (deux pixels blancs et 6 pixels noirs). Cela s'explique par le fait que le voisinage est trop petit. En effet, les pixels bruités qui sont restés étaient placés au centre d'un voisinage qui était concentré en pixels bruités. Lorsque nous prenons l'image du zoom sur l'image bruitée de l'endroit où a été prélevé l'échantillon de zoom pour l'image filtrée, nous trouvons la figure suivante :

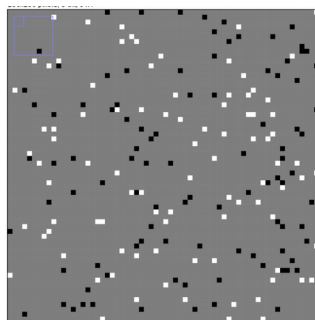


FIGURE – Zoom de l'image bruitée au même endroit que l'image filtrée

Nous remarquons que, dans l'image bruitée, le pixel noir qui a été conservé après le filtre était entouré en majorité de voisins directs de couleur noir également. Cependant, le PSNR nous montre que le résultat est tout de même acceptable car il y a peu d'erreurs.

Nous pouvons conclure sur le fait qu'il faut un voisinage assez grand pour être capable de supprimer les pixels aberrants si ceux-ci sont trop concentrés.

Test sur l'image de cake

Nous allons d'abord appliquer un bruit poivre & sel sur l'image des cakes utilisée auparavant.

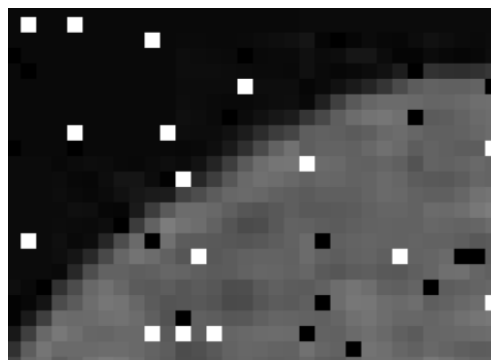
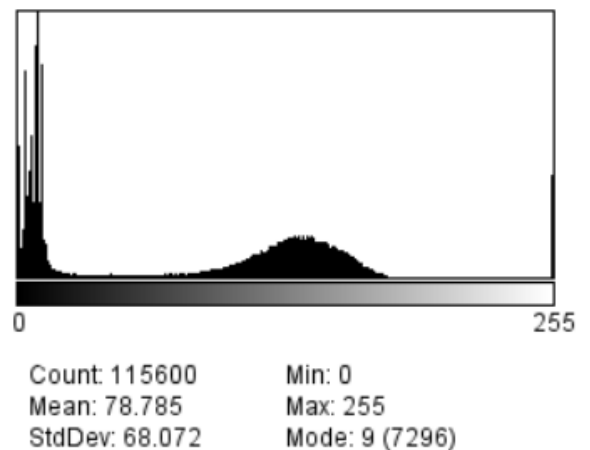
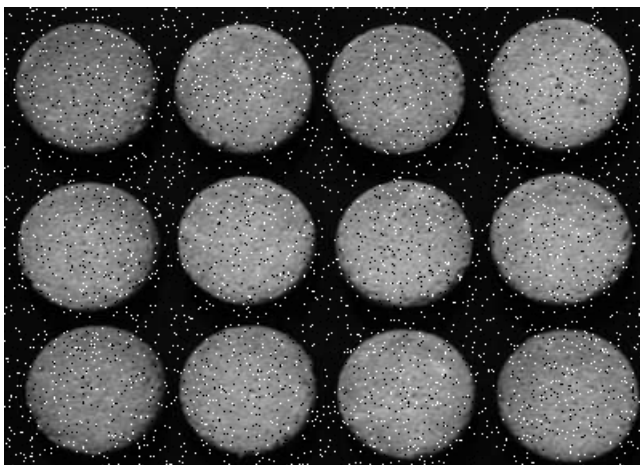
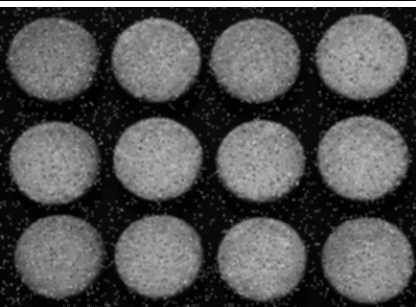
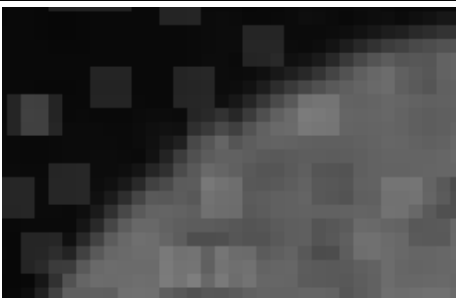
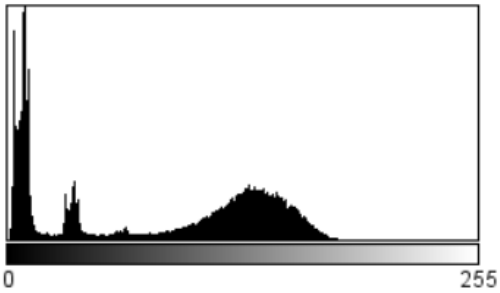
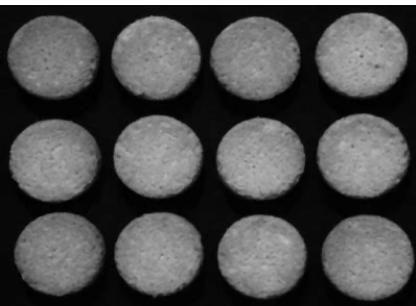
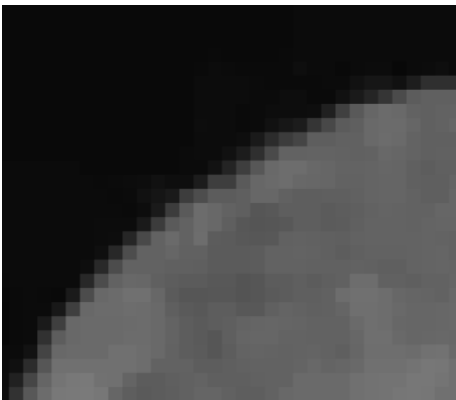



FIGURE – cakes avec bruit impulsif + histogramme + zoom

Le **PSNR** entre l'image originale et cette image bruitée est de **17,7333 dB**.

Nous allons maintenant comparer l'efficacité des filtres moyenneurs et médian pour un bruit impulsif sur une image naturelle à l'aide de cette image bruitée :

Image filtrée	Zoom	Histogramme	PSNR En dB
 Filtre moyenneur H1 normalisé		 Count: 115600 Min: 0 Mean: 78.783 Max: 198 StdDev: 59.917 Mode: 9 (5352)	26,0319
 Filtre médian de rayon 1		 Count: 115600 Min: 0 Mean: 76.364 Max: 188 StdDev: 62.284 Mode: 9 (8015)	35,0164

On remarque sans effort que le résultat obtenu par un filtrage médian de rayon 1 est largement plus convaincant que le filtrage de lissage linéaire par masque de convolution H1 normalisé.

En effet, le filtre moyenneur H1 opère un flou pour lisser les pixels aberrants tandis que le filtre médian va complètement supprimer la valeur aberrante et la remplacer par un niveau de gris médian.

Le problème que l'on peut voir sur l'image résultante du filtre médian est le fait que certaines irrégularités très légères du cake (qui n'est pas du bruit) sont aussi supprimées par le filtre.

Lorsque l'on compare les histogrammes, on voit aussi que l'histogramme de l'image filtrée par filtre médian est fortement similaire à l'histogramme original. En ce qui concerne l'histogramme de l'image filtrée par H1 normalisé, on remarque l'apparition d'un mode entre les deux modes principaux de l'histogramme. On peut émettre l'hypothèse que ce mode est composé des pixels aberrants qui sont apparus sur le fond de l'image et qui ont été floutés.

Implémentation d'un filtrage min-max

Le but de cette partie du TP est d'implémenter un filtrage min-max à l'aide du langage macro d'ImageJ.

Le filtrage min-max consiste à modifier la valeur de niveau de gris $I(x, y)$ de chaque pixel de la manière suivante :

- $I'(x, y) = i_{min} \Leftrightarrow I(x, y) < i_{min}$
- $I'(x, y) = i_{max} \Leftrightarrow I(x, y) > i_{max}$
- $I'(x, y) = I(x, y) \Leftrightarrow i_{min} \leq I(x, y) \leq i_{max}$
- où
 - $I'(x, y)$ est le nouveau niveau de gris de l'image filtrée
 - i_{min} et i_{max} sont les niveaux de gris minimal et maximal du voisinage du pixel courant $I(x, y)$

La macro permettant de calculer ce filtre pour ImageJ est la suivante :

```

// Charger l'image et ses dimensions
image = getImageID();
H = getHeight();
W = getWidth();

// Cloner l'image pour calculer la a
run("Duplicate...", "title=result");
result = getImageID();

// filtrage traitement
setBatchMode(true);

for (i = 0; i < W; i++) {
    for (j = 0; j < H; j++) {

        // Trouver les pixels min et max du voisinage
        min = 255;
        max = 0;

        selectImage(image);
        courant = getPixel(i, j); // récupérer le pixel courant

        // Pour chaque pixel du voisinage 3x3 autour du courant...
        for(x = (i-2); x <= (i+2); x++) {
            for(y = (j-2); y <= (j+2); y++) {
                if(x == i && y == j) {
                    // on ne fait rien quand on est sur le pixel courant...
                }
                else {
                    p = getPixel(x, y);
                    if(p < min) {
                        min = p;
                    }

                    if(p > max) {
                        max = p;
                    }
                }
            }
        }
        // NOTE : getPixel() sur des index qui ne sont pas dans les dimensions de l'image renvoie 0.
        // Cependant on suppose que la stratégie d'effet de bord adoptée est l'effet miroir
        // et que l'image est celle des cakes. De ce fait, les bords qui seront réfléchis valent d'office 0 pour un voisinage
        // aussi petit que celui que l'on cherche. Donc nous n'avons pas à nous inquiéter des effets de bords ici.
    }
}

// SI le pixel courant est plus grand que le max du voisinage...
if(courant > max){
    // Le pixel courant prend la valeur du max
    selectImage(result);
    setPixel(i, j, max);
}

// SI le pixel courant est plus petit que le min du voisinage...
if(courant < min){
    // Le pixel courant prend la valeur du min
    selectImage(result);
    setPixel(i, j, min);
}
}

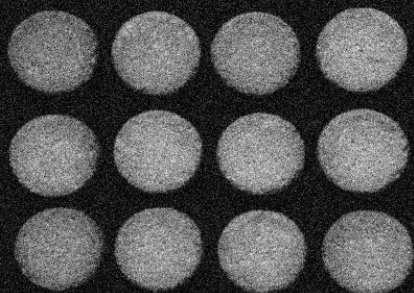
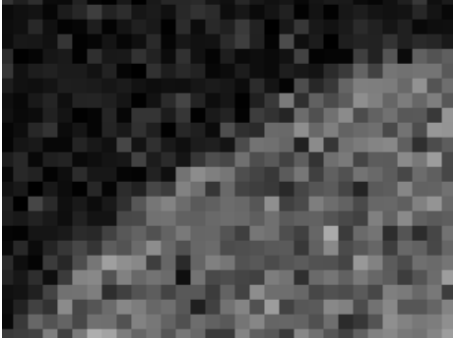
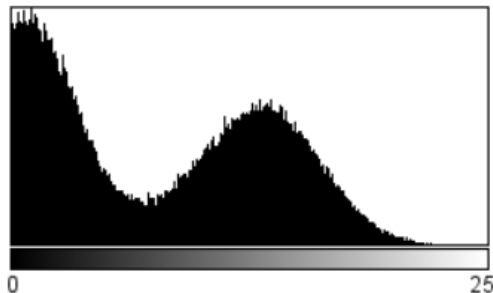
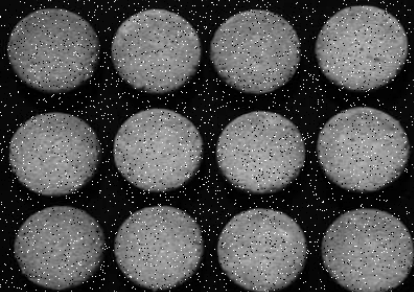
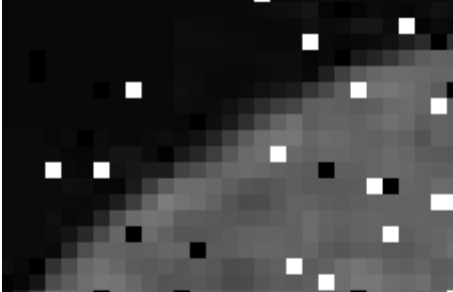

// fin filtrage
setBatchMode(false);
selectImage(result);

```

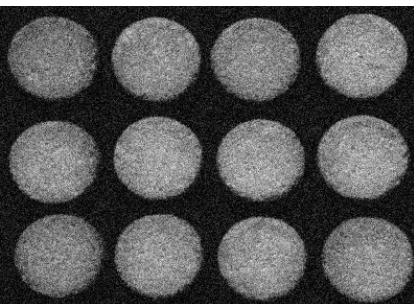
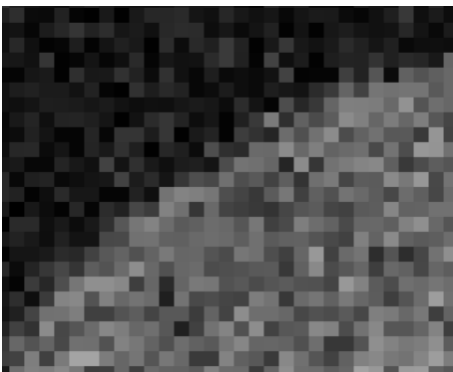
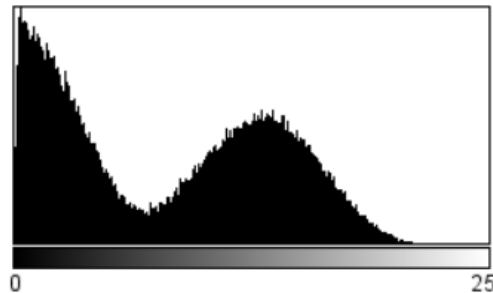
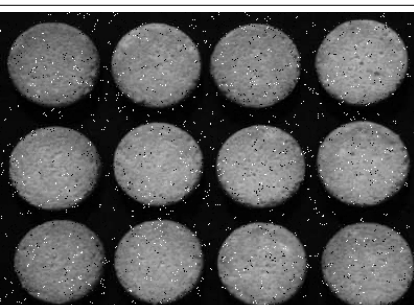
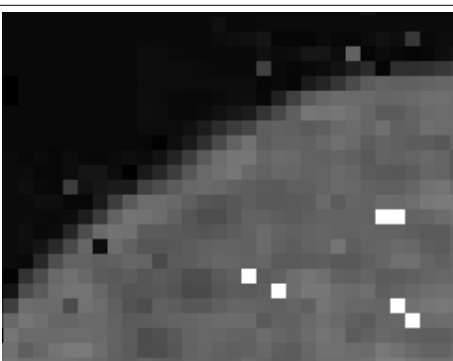
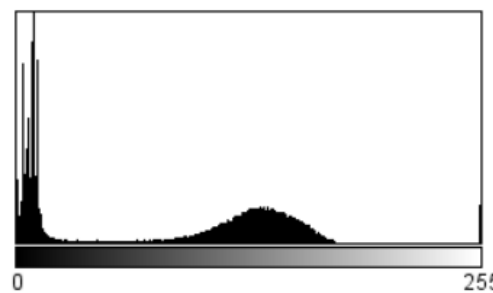

Nous allons tester ce filtrage min-max sur des images bruitées de l'image de cakes.

Les images bruitées sont :

- Image affectée d'un bruit Gaussien d'écart-type $\sigma = 25$
- Image affectée d'un bruit impulsif

Image filtrée	Zoom	Histogramme	PSNR En dB
 <p>Bruit gaussien $\sigma = 25$</p>		 <p>Count: 115600 Mean: 82.792 StdDev: 59.685</p> <p>Min: 0 Max: 252 Mode: 10 (1305)</p>	20.5518
 <p>Bruit poivre et sel</p>		 <p>Count: 115600 Mean: 78.873 StdDev: 68.097</p> <p>Min: 0 Max: 255 Mode: 9 (7299)</p>	17,6353

Les images filtrées par filtrage min-max sont les suivantes :

Image filtrée	Zoom	Histogramme	PSNR En dB
 <p>Bruit gaussien $\sigma=25$ filtré</p>		 <p>Count: 115600 Min: 0 Mean: 82.624 Max: 236 StdDev: 59.297 Mode: 3 (1433)</p>	20.9997
 <p>Bruit poivre et sel filtré</p>		 <p>Count: 115600 Min: 0 Mean: 77.476 Max: 255 StdDev: 65.041 Mode: 9 (7462)</p>	21,0958

Après ces tests, nous remarquons plusieurs choses au sujet du filtre min-max :

Pour atténuer un bruit gaussien, le filtre de lissage non-linéaire est moins efficace que d'autres méthodes (ex : filtre gaussien). Cela s'explique par le fait que le bruit gaussien modifie trop de pixels dans le voisinage d'un pixel. Par conséquent, la recherche du minimum et du maximum est

erronée et le filtre ne peut pas correctement corriger le bruit. On observe facilement ce problème puisqu'on remarque que très peu de choses ont changé après le filtre.

Le filtre d'une image bruitée à l'aide d'un bruit impulsionnel donne un résultat à priori moins efficace que le filtre médian. En outre, quand on observe la répartition de l'histogramme, on remarque que le filtre a réussi à répartir une grande partie des pixels erronés (de valeur 0 ou 255) sur les deux modes de l'histogramme.

Une autre chose à souligner lorsque l'on compare avec le filtre médian est le fait que les irrégularités du cake (qui ne font pas partie du bruit mais de l'image originale) sont mieux préservées par le filtrage min-max que par le filtrage median. Cela s'explique par le fait que le filtrage min-max va rechercher à modifier le niveau de gris du pixel calculé sur le minimum ou le maximum en cas de valeur aberrante là où le filtre médian va radicalement modifier le niveau de gris du pixel suivant la valeur médiane du voisinage.