Les objectifs du TP sont bien compris et les étapes globalement bien décrites.

Alice Sparrow
Gautier Kasperek

Traitement d'Image : TP8 - Watermarking
Tatouage d'image par étalement de spectre

## I. Introduction

L'objectif de ce compte-rendu est de montrer comment nous avons obtenu, à partir du squelette d'une macro, un système complet de tatouage d'image. Celui-ci permettant l'insertion ("embedding") d'un message d'un nombre quelconque de bits dans une image hôte codée en niveau de gris. Il permet également le décodage ("decoding") d'un message caché dans une image tatoué. Ce TP est réalisé avec imageJ permettant de réaliser des opérations sur les images ainsi que de leur appliquer des macros.

## II. Filtres de contours

Dans un premier temps, nous allons calculer la norme du vecteur gradient d'une image. Le but est de comprendre son fonctionnement et son résultat afin de possiblement l'exploiter dans le tatouage d'une image.

Il est important de réaliser ces opérations avec des images au format 32 bits puisque nous réalisons de nombreux calculs sur les valeurs des pixels. Ainsi, cela évite de perdre certaines valeurs lorsque l'on calcule par exemple le carré de pixel approchant 255.Pour cette partie, nous commencerons avec l'image *"peppers.pgm"*.

Il y a une autre raison, quel type de donnée code-t-on habituellement avec 32 bits ?

*Figure 1 : Image peppers.pgm*

Pour calculer la norme du vecteur gradient nous avons utilisés les filtres de Sobel sur deux images dupliqués de “peppers.pgm”.

$$H_x = \begin{bmatrix} -0.125 & 0 & 0.125 \\ -0.25 & 0 & 0.25 \\ -0.125 & 0 & 0.125 \end{bmatrix}, \; H_y = \begin{bmatrix} -0.125 & -0.25 & -0.125 \\ 0 & 0 & 0 \\ 0.125 & 0.25 & 0.125 \end{bmatrix}$$

Le filtre de Sobel calcule le gradient de l’intensité de chaque pixel. Le gradient nous permet de désigner la direction de la plus forte variation du plus clair au plus sombre. Les changements de luminosités nous permettent de définir de probable contour à l’image.
L’application du filtre Hx permet d’effectuer une détection de manière verticale, alors que la matrice Hy une détection dans le sens horizontal.

***A préciser***

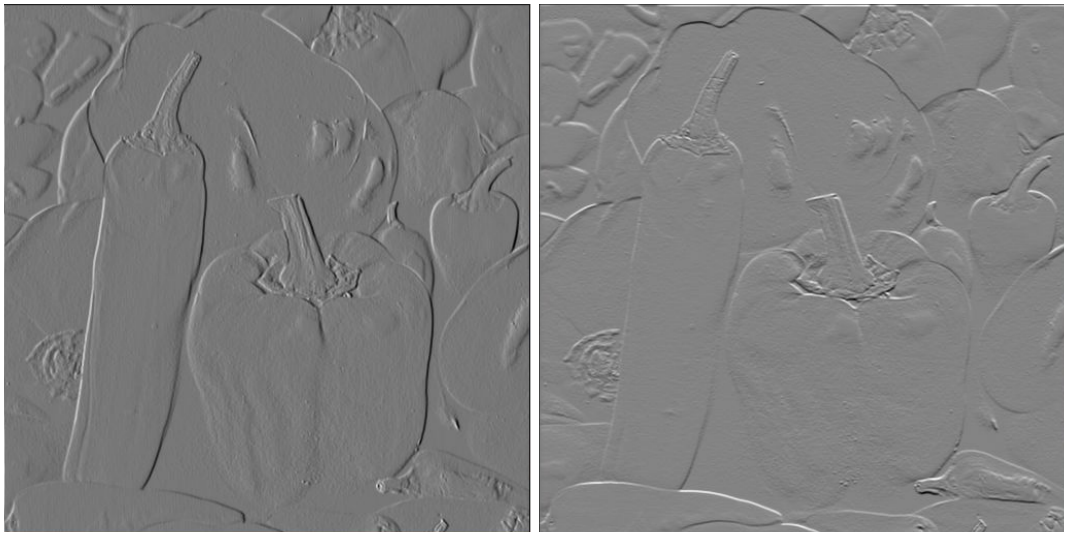


*Figure 2 : A gauche, l’image dx, Hx appliqué “peppers.pgm”. A droite, l’image dy, Hy appliqué à “peppers.pgm”*

Nous pouvons déjà voir apparaître certains changements de luminosité et de contour sur ces deux images. Nous pouvons observer que certains contours de *dx* et *dy* sont complémentaires. Par exemple au niveau de la tige poivron centrale. Nous allons maintenant élevé au carré les valeurs des pixels de ces deux images.

*Figure 3 : A gauche, dx élevé au carré. A droite dy élevé au carré*

Ensuite, nous additionnons les valeurs de pixels de dx et dy afin d'obtenir une seule image.

*Figure 4 : Image résultat de dx + dy*

Enfin, nous appliquons la racine carré de l'image résultat ainsi qu'un seuillage entre 0 et 16 permettant d'obtenir un résultat satisfaisant des contours de l'image de départ *"peppers.pgm"*.

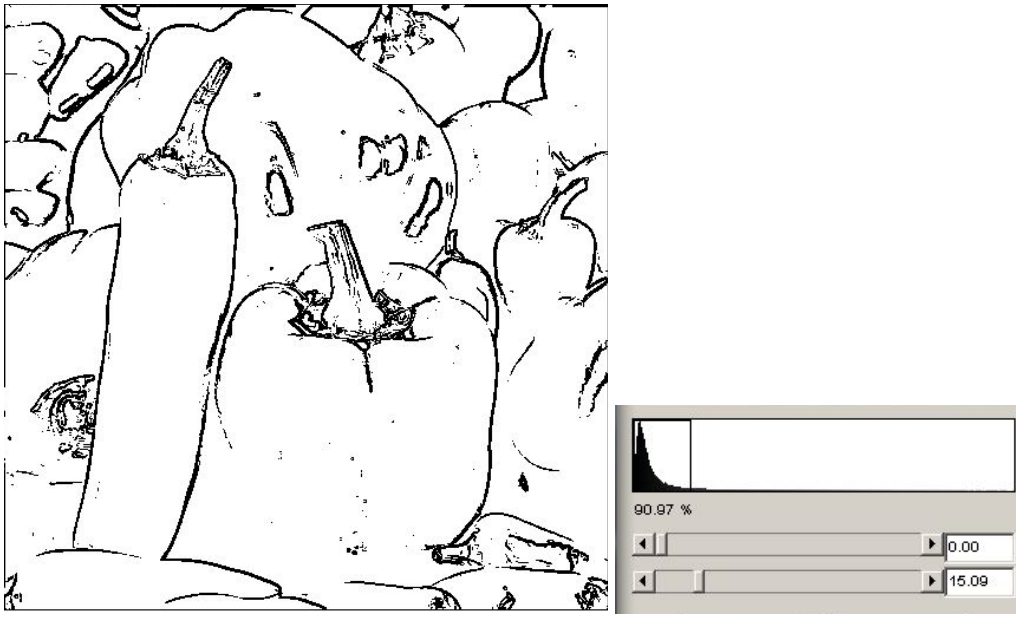



*Figure 5 : Filtre de Sobel appliqué à l'image "peppers.pgm" plus seuillage*

Ayant obtenus un résultat satisfaisant de l'application du filtre de Sobel nous avons décidé de l'appliquer à d'autres images afin de mettre à l'épreuve l'efficacité du filtre en utilisant des images possédant des contours plus ou moins marqué. Les opérations ne sont pas détaillés mais chaque image représente l'une des opérations intermédiaire précédentes.

Le résultat est-il vraiment satisfaisant ? Récupère-t-on tous les contours de l'image ? Récupère-t-on des détails qui ne sont pas des contours ?

*Figure 6 : Image “godafos.pgm”*

La figure suivante illustre les étapes intermédiaires de l’application du filtre de Sobel.

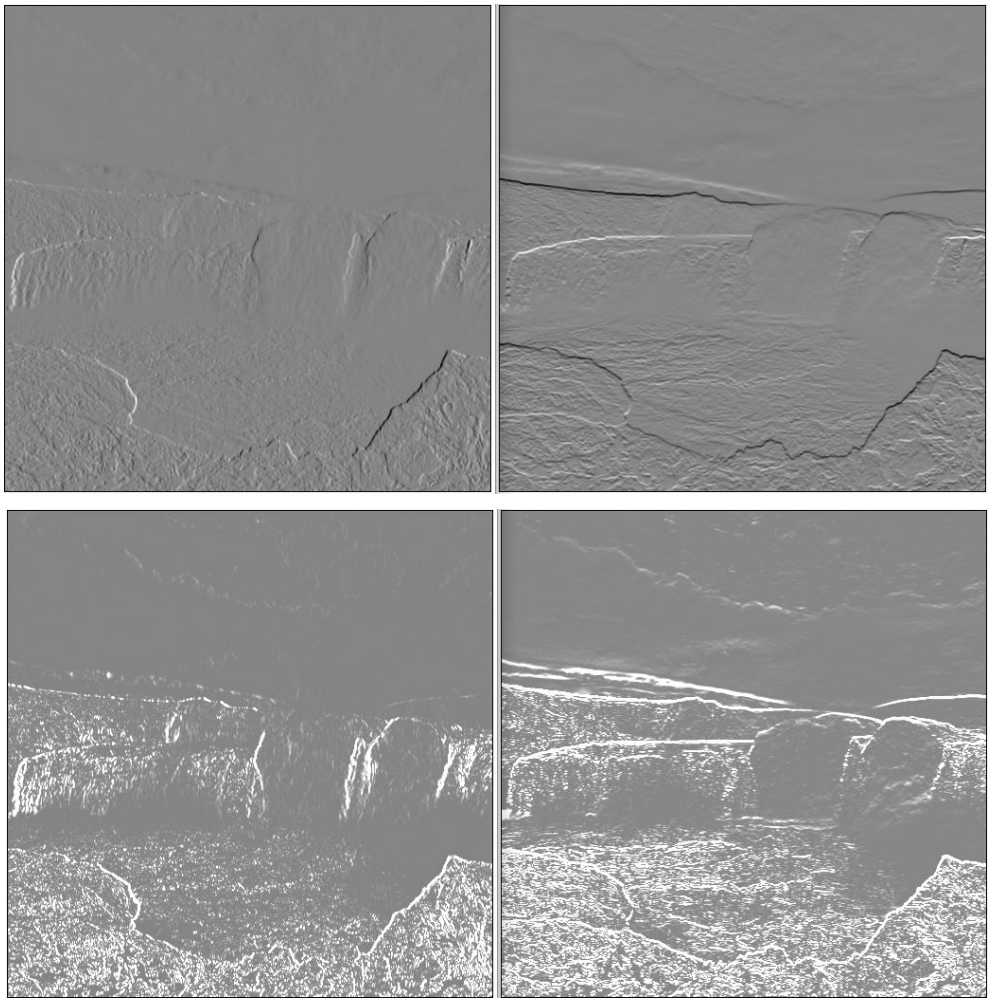


*Figure 7 : Filtre de Sobel appliqué à l'image "godafoss.pgm"*

Sur les résultats suivant, nous avons réalisés deux seuillages différents. Le premier entre 0 et 4, le second entre 0 et 14. Nous pouvons remarquer que la texture des rochers et de l'eau complique la détection de contours. Dans le premier cas, nous avons une délimitation clair du ciel et des nuages. Dans le second cas, le seuillage permet de séparer l'eau des rochers mais ne distingue pas du tout la délimitation de la terre et du ciel

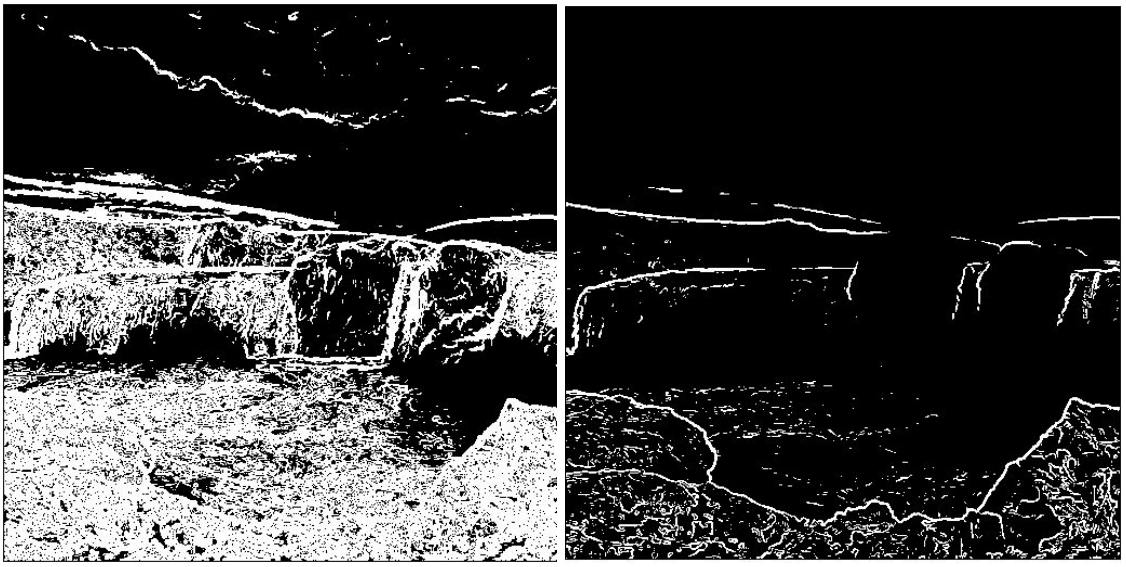


*Figure 8 : Seuillage de l'image godafoss.pgm après application du filtre de Sobel*

L'un des principaux avantage du filtre de Sobel est sa simplicité. Les opérations simples qui le propose permette de l'implémenter et de l'exécuter facilement en obtenant un résultat souvent satisfaisant. Cependant, certains cas peuvent être problématiques pour ce filtre notamment lorsque les différentes composantes de l'image ne sont pas assez contrasté ou lorsque les contours sont trop proches les un des autres.

Oui, d'où l'intérêt de techniques + évoluées pour la détection de contours

Nous allons maintenant nous intéresser au tatouage d'une image par étalement de spectre, nous verrons notamment comment utiliser le filtre de Sobel pour améliorer la qualité du tatouage.

III. Tatouage par étalement de spectre

Dans cette partie nous allons réaliser un tatouage par élément de spectre. Pour cela, on utilise une image hôte que l'on divise en tuiles. Chacune des tuiles sera tatouée par étalement de spectre avec le même message de **Nc** bits. La clé secrète est constitué de **Nc** porteuses *"carriers"* de la taille des tuiles et est généré depuis une graine *"seed"*. L'insertion se réalise par l'addition de l'image hôte et du signal de tatouage multiplié par un facteur gamma dépendant d'un PSNR fixé permettant de maîtriser la distorsion.
Nous allons d'abord réaliser le tatouage d'une image. Nous choisissons comme image hôte l'image *"mandrill.pgm"* présente ci-dessous.

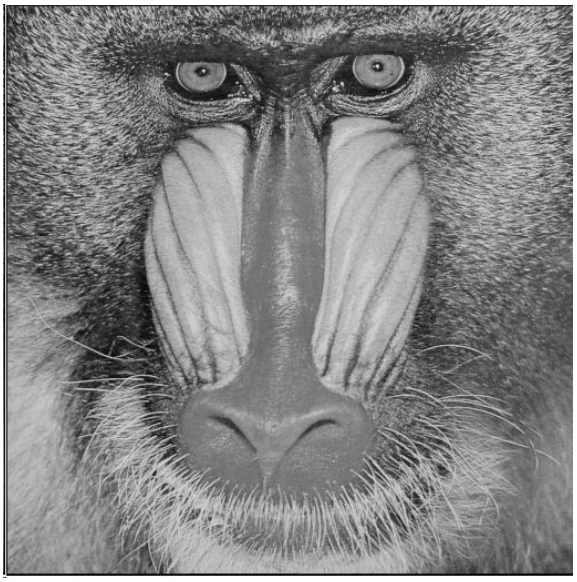


*Figure 9 : "mandrill.pgm" Image hôte choisis*

En lançant l'insertion avec les paramètres de base de la macro nous obtenons l'image ci-dessous :

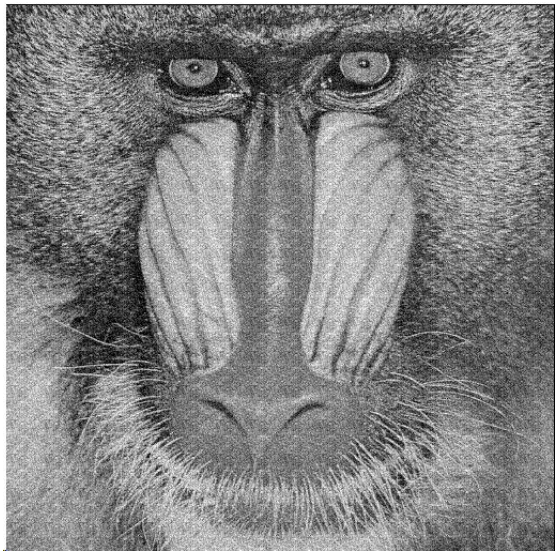


*Figure 10 : "mandrill.pgm" plus insertion d'un message*

Quel message ? Quel PSNR cible ?

Sur cette image, on peut facilement remarquer que le tatouage a apporté des modifications perceptible à l'oeil. En effet, la partie nasale du mandrill qui est une zone uniforme et qui possède une basse fréquence est grandement modifié.

On choisit donc empiriquement 35 dB pour valeur de PSNR de manière à ce que le tatouage ne soit plus perceptible à l'oeil. ✔

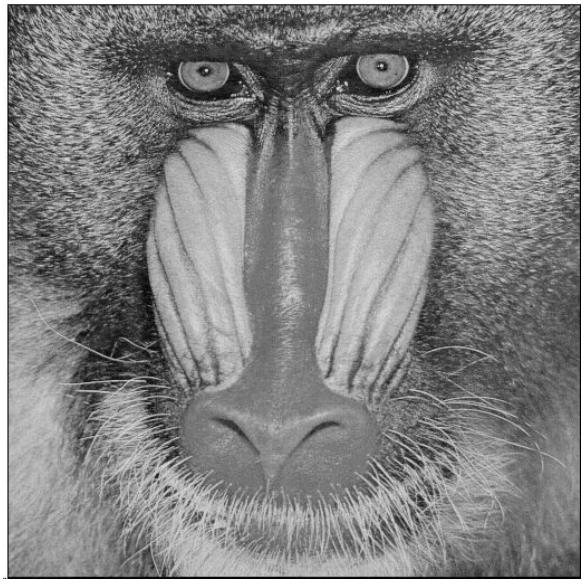


*Figure 11 : "mandrill.pgm" avec insertion de message (PSNR ciblé 35dB)*

Ci-dessous, la macro makeWmkSignal avec les instructions détaillés en commentaires.

```
function makeWmkSignal(IDcarriers,widthImg,heightImg,msg)
{

    selectImage(IDcarriers); /* On selectionne l'image courante */
    getDimensions(widthTile, heightTile, channels, Nc, frames);
    /* Créé une image 32 bit de largeur widthImg et de hauteur heightImg
*/
    newImage("WmkSignal", "32-bit black", widthImg, heightImg, 1);

    IDWmk = getImageID();

    /* Créé une image 32 bit de largeur widthTile et de hauteur
heightTile*/
    newImage("WmkTile", "32-bit black", widthTile, heightTile, 1);
    IDWmkTile = getImageID();



    /* Spread-Spectrum */
    /*On ajoute le tatouage à chacunes des stacks de l'image.*/
```

```
    for(i=0; i< Nc; i++)
    {
       selectImage(IDcarriers);
       //indexes of stacks start at 1!!
       setSlice(i+1);
       if (msg[i] == 1)
             run("Multiply...", "
       // On additionne la porteuse et tuile
       imageCalculator("Add 32-bit", IDWmkTile, IDcarriers);
    }



    selectImage(IDWmkTile);

    /* update display LUT */
    resetMinAndMax();

    // On créer un rectangle de la taille d'une tuile
    makeRectangle(0, 0, widthTile, heightTile);
    run("Copy");
    run("Select None");



    // Sur l'image on colle tuile par tuile
    selectImage(IDWmk);
    for(j = 0 ; j < widthImg; j = j + widthTile)
    {
       for(i = 0 ; i < heightImg; i = i + heightTile)
       {
             makeRectangle(j, i, widthTile, heightTile);
             run("Paste");
       }
    }
    run("Select None");



    return IDWmk;
}
```

la porteuse est multipliée par -1 si on veut cacher un 1

une sélection sur l'image

Afin de réaliser le décodage, nous avons réalisé une macro “bitErrorRate”. Cette fonction calcule le Bit Error Rate d’une image. C’est-à-dire la distance de Hamming entre deux

messages divisé par le nombre de porteuses, autrement dit :

$$BER = \frac{DistanceHamming(msg, msgd)}{Nombre\ de\ porteuses}$$

```
function bitErrorRate (msg,msgd){
	cpt =0;
	for(i=0;i<Nc;i++){
		if(msg[i] != msgd[i]){
			cpt++;
		}
	}
	return (cpt/Nc);
}
```

Test ? Le BER vaut-il 0 pour une image tatouée à 35dB ?

Pour améliorer le tatouage d'une image naturelle, il est possible de prendre en compte le système visuel humain (SVH). Le SVH est moins sensible aux zones de hautes fréquences comme les textures ou les contours, il est donc plus sensible aux zones de basses fréquences comme les surfaces planes. C'est le phénomène que nous avons constaté plus haut sur le nez du mandrill. Afin d'exploiter ce résultat nous allons prendre en compte la norme du gradient, pour chaque pixel le signal du tatouage sera multiplié par la norme du gradient de Sobel de l'image hôte. Ainsi, le signal sera atténué sur les zones où le gradient est faible, c'est-à-dire les surfaces planes.

et à contrario, sera augmenté sur les hautes fréquences

La macro *makeGradientNorm* ci-dessous réalise le filtre de Sobel que nous avons détaillé dans la première partie de ce TP.

```
function makeGradientNorm(id){
	selectImage(id);
	run("Duplicate...", "title=dx.pgm");
	selectImage("id");
	run("Duplicate...", "title=dy.pgm");
	run("32-bit");
	run("Convolve...", "text1=[-0.125 -0.25 -0.125\n0 0 0\n0.125 0.25
0.125\n] normalize");
	selectWindow("dx.pgm");
	run("32-bit");
	run("Convolve...", "text1=[-0.125 0 0.125\n -0.25 0 0.25\n-0.125 0
0.125\n] normalize");
	run("Square");
	selectWindow("dy.pgm");
	run("Square");
	imageCalculator("Add create 32-bit", "dx.pgm","dy.pgm");
	selectWindow("Result of dx.pgm");
```

```
	run("Square Root");
	run("Max...", "value=255");
	return getImageID();
}
```

La macro précédente nous permet de récupérer la norme du gradient de l'image hôte que l'on souhaite tatouer. Ci-dessous, le résultat du filtre de Sobel de l'image :

*Figure 12 : Norme du gradient de l'image mandrill.pgm*

Nous allons maintenant exploiter la norme du gradient dans le tatouage de l'image. Pour chaque pixel de coordonné (x,y) on multiplie la valeur du tatouage par celle de la norme du gradient aux coordonnées correspondantes.

```
	/* IDWmk le tatouage de la taille de l'image */
	IDWmk = makeWmkSignal(IDcarriers,widthImg,heightImg,msg);

	/* psychovisual masking using the norm of the gradient */
	/* IDgradient est la norme du gradient de l'image hôte */
	IDgradient = makeGradientNorm(IDhost);
	for(x = 0; x < widthImg; x++){
		for(y = 0; y < heightImg; y++){
			selectImage(IDgradient);
			pixGrad = getPixel(x,y);
			selectImage(IDWmk);
			wmk = getPixel(x,y);
			/* On multiplie la valeur du tatouage par la norme du
gradient en (x,y) */
```

```
                setPixel(x,y,wmk*pixGrad);
            }
}
```

L'application de la norme du gradient au tatouage nous permet d'obtenir le résultat ci-dessous.

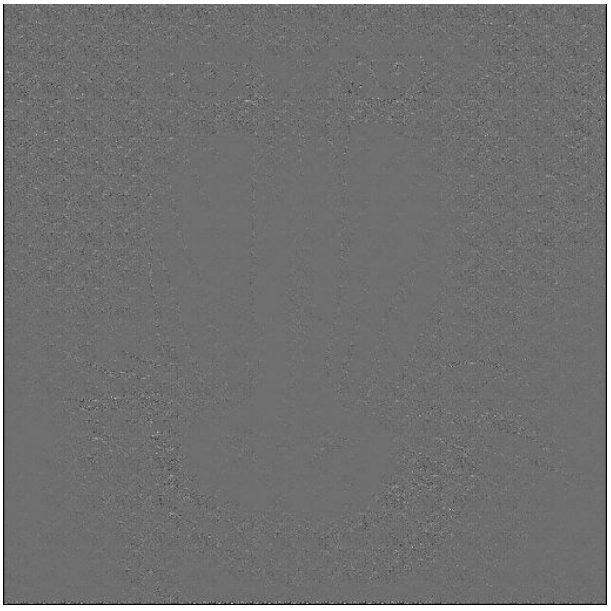


*Figure 13 : Tatouage multiplié par la norme du gradient*

On peut remarquer que certaines zones du tatouage sont lisses. En effet, elles correspondent aux zones où la norme du gradient est nul ou très faible. Ainsi, en multipliant les valeurs du tatouages par des valeurs tendant vers 0, cela permet d'atténuer la perceptibilité du tatouage sur les zones de basses fréquences comme le nez ou les joues du mandrill.

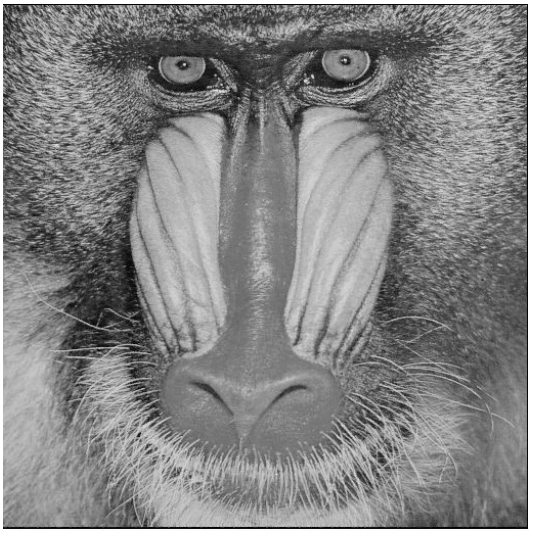


*Figure 14 : Image hôte tatoué en prenant en compte la norme du gradient*

IV. Robustesse du schéma de tatouage

Après avoir réaliser une macro permettant de tatouer une image de manière à ce qu'elle soit peu perceptible pour le système visuel humain, nous allons nous intéresser à la robustesse de la méthode de tatouage. C'est-à-dire, sa capacité à conserver l'information tatouée après que des modifications soient apportées à l'image.
Pour cette partie nous utiliserons l'image *"mandrillWmk.pgm"* image résultant du tatouage de la partie précédente.

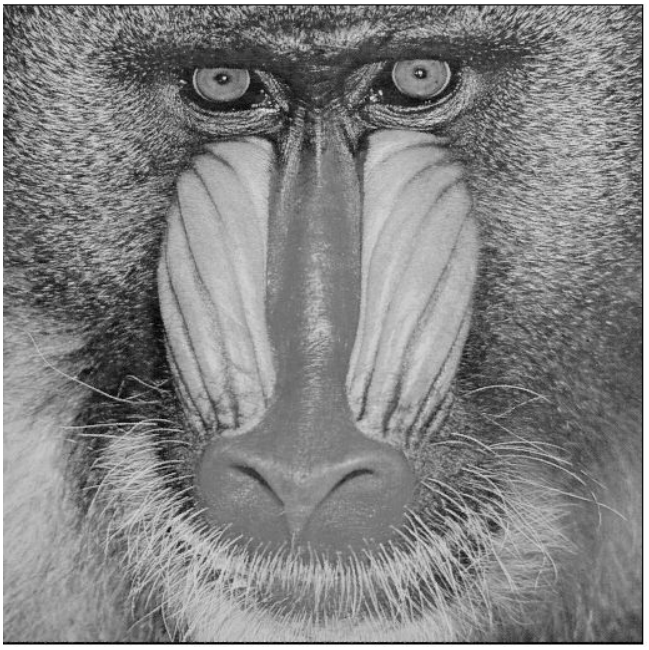


*Figure 15 : mandrillWmk*

Nous allons chercher l'écart type du bruit à partir duquel on commence à obtenir des erreurs de décodage. Pour cela nous allons employer une méthode empirique. Les figures allant de 16 à 22 montrent les tests effectués afin de trouver cette valeur.


*Figure 16 : Bruit Gaussien d'écart-type $\sigma = 300$*


*Figure 17 : Bruit Gaussien d'écart-type $\sigma = 200$*


*Figure 18 : Bruit gaussien d'écart-type $\sigma = 150$*


*Figure 19 : Bruit gaussien d'écart-type $\sigma = 125$*


*Figure 20 : Bruit gaussien d'écart-type σ = 140*


*Figure 21 : Bruit gaussien d'écart-type σ = 135*

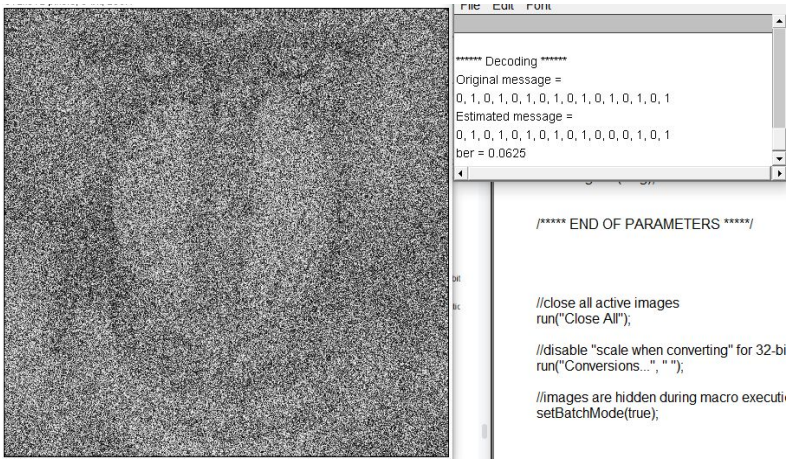



*Figure 22 : Bruit gaussien d'écart-type σ = 130*

Avec un limite à un bruit gaussien d'écart-type 130, le tatouage semble plutôt robuste à l'ajout de bruit. Si une attaque par bruit gaussien est réalisés, l'image ne serait presque plus visible ce qui détectera l'attaque.

Pour présenter la robustesse, on aurait pu tracer le BER en fonction de sigma.

Nous allons maintenant tatouer une image et analyser sa FFT.

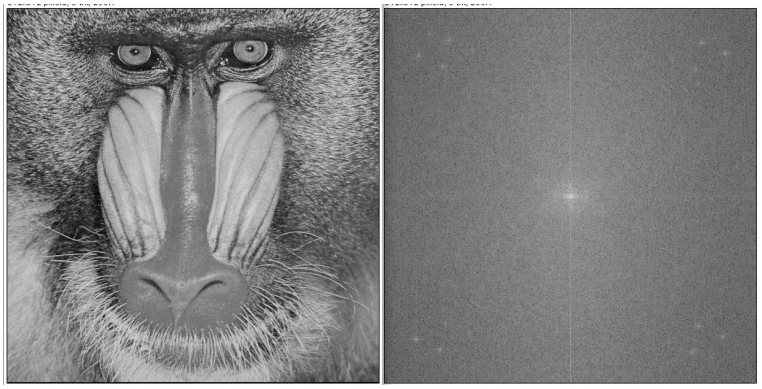


*Figure 23 : Image hôte mandrill.pgm et sa FFT*

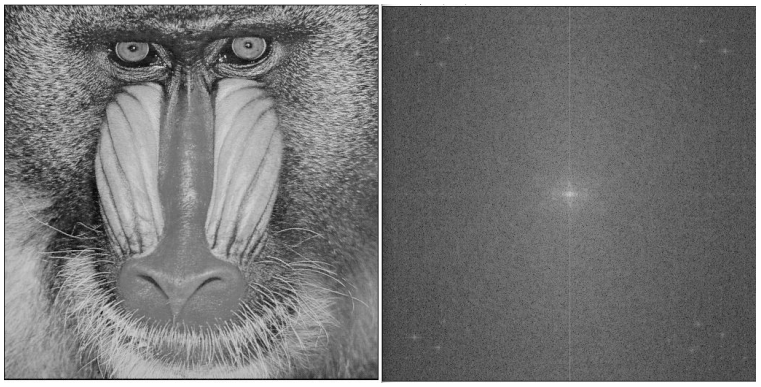
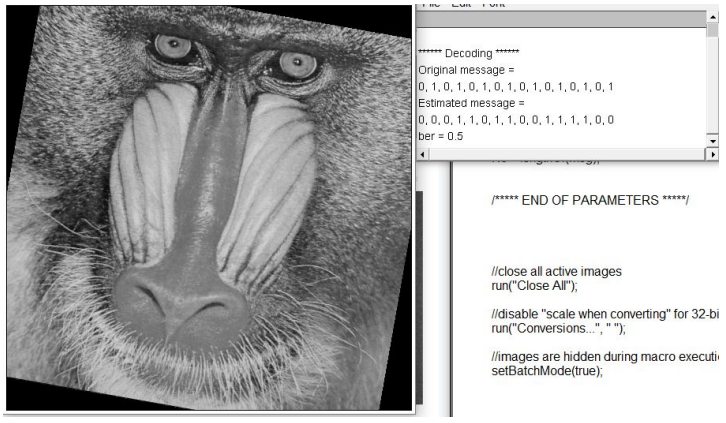


*Figure 24 : Image tatoué mandrillWmk.pgm et sa FFT*




Il faut + détailler les résultats obtenus, ici un BER = 0.5 signifie bien que le décodage n'est pas correct (50% du message non décodé)

*Figure 25 : Image MandrillWmk.pgm avec rotation et son décodage*

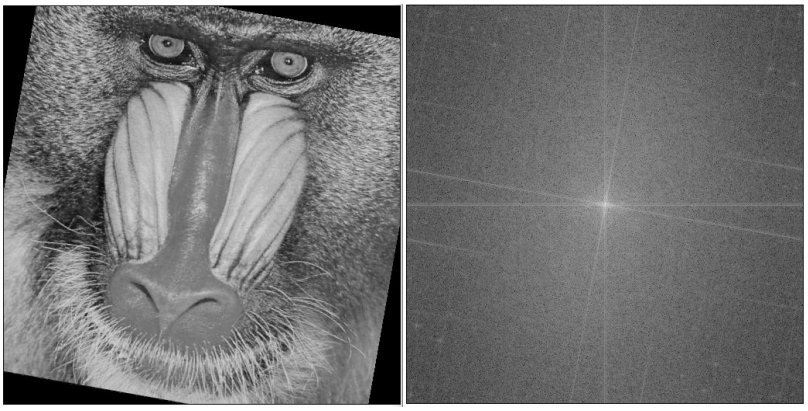


*Figure 26 : Image MandrillWmk.pgm avec rotation et sa FFT*

On peut utiliser la transformée de Fourier pour rendre le tatouage robuste à la rotation. Grâce aux raies secondaires créées en diagonale dans celle-ci. On calcule l'angle créé en (0,0) d'un couple de raies. Cet angle correspond à la rotation appliquée sur l'image tatouée, on applique la rotation d'un angle inverse à celui trouvé grâce aux raies de la FFT puis on décode.

**Bonne analyse**

On récupère le message décodé même avec les partie noire une fois que l'on a fait une rotation inverse égale à la première rotation effectué

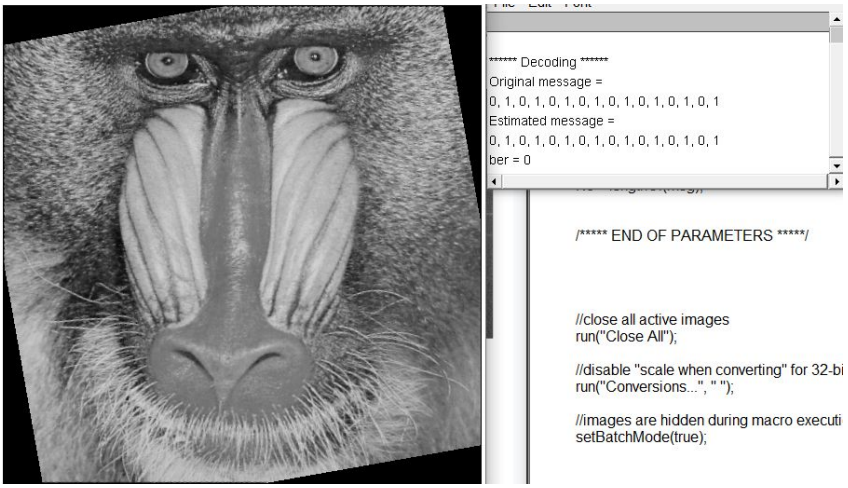



*Figure 27 : Rotation +10 puis -10 sur l'image mandrillWmk.pgm*