UNIVERSITÉ DE RENNES 1

ENSSAT LANNION

Tatouage d'une image

Compte-rendu

Réalisé par **M. Bruhat et M. Jové**

Encadré par **M. Cariou** Année **2017-2018**

Introduction

Le module d'analyse et traitement d'image s'inscrit dans la seconde année de la formation IMR (Informatique Multimédia et Réseaux) à l'ENSSAT. Ce module permet de nous faire découvrir, via une mise en pratique, le tatouage d'une image à l'aide du logiciel Scilab.

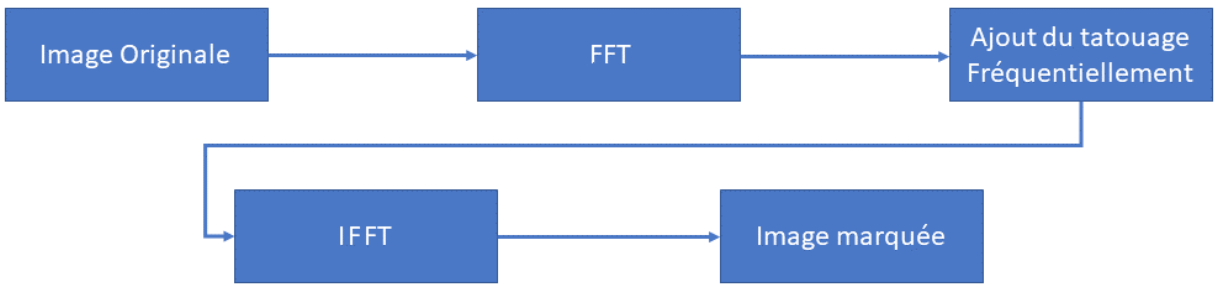
L'objectif est d'appliquer une signature spécifique à une image dans le but de pouvoir identifier à l'avenir de façon certaine que l'image appartient à tel ou tel entité. Cette signature doit également être robuste afin de résister à d'éventuels baisses de qualités (compressions, rotations, extraction d'une sous-image, etc.). Enfin la dernière partie de ce TP consiste à développer un algorithme permettant de reconnaître un tatouage appliqué à une image. Cette dernière partie n'a pas pu être traitée totalement, faute de temps, cependant nous présenterons un schéma explicatif concernant cette partie.

Nous commencerons, par vous présenter dans un premier temps l'étude préliminaire réalisée avant de présenter étape par étape le développement de notre projet. Puis nous terminerons avec la présentation de nos différents résultats obtenus ainsi qu'une brève comparaison de la robustesse entre nos différentes images.

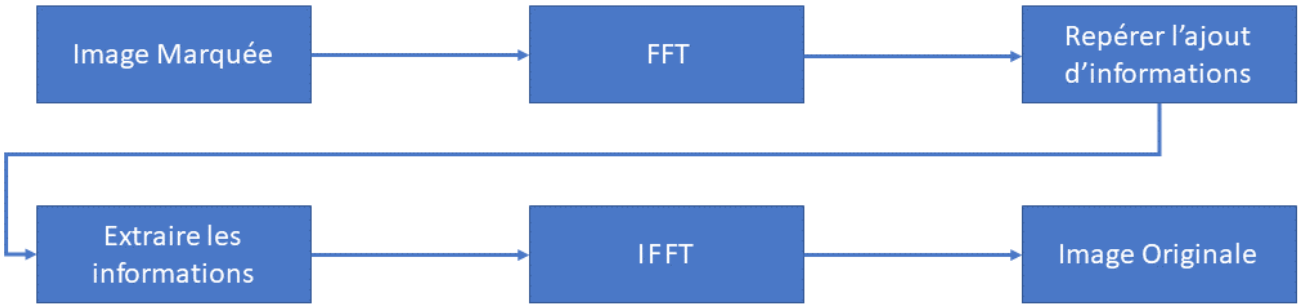
I- Etude préliminaire

Il existe différentes méthodes d'insertion d'une signature (appelée aussi marque). On peut distinguer généralement deux types de marques. Le premier type de marque est celle dans le domaine spatial, l'autre type de marque est celle travaillant dans le domaine spectral. Dans le cadre de notre TP, nous souhaitons un marquage robuste. Or les techniques purement spatiales résistent mal à certaines attaques comme le zoom et le recadrage. A contrario, les techniques opérant dans le domaine fréquentiel résistent bien à ce type d'attaques.

Afin de respecter les contraintes initiales, nous avons fait le choix d'inscrire notre tatouage dans le domaine fréquentiel. Ce domaine est obtenu après utilisation d'une TFD (Transformée de Fourier Discrète) ou d'une TCD (Transformée en Cosinus Discrète). La TFD permet d'évaluer une représentation spectrale discrète d'un signal discret sur une fenêtre de temps finie. Nous utiliserons dans notre TP la fonction FFT (Fast Fourier Transform) qui est un algorithme spécifique de calcul de la TFD. La DCT, quant à elle, possède une excellente propriété de « regroupement » de l'énergie : l'information est essentiellement portée par les coefficients basses fréquences. Voici maintenant le schéma explicitant le procédé de marquage que nous avons choisi de mettre en place :



Aussi, nous avons prévu un second schéma permettant de repérer et d'extraire un tatouage dans une image. Voici les différentes étapes prévues ci-dessous :



Enfin nous avons également prévu d'ajouter un module permettant de calculer la robustesse de nos images résultantes. Pour cela nous réaliserons une fonction nommée PSNR (Peak Signal to Noise Ratio) qui nous retournera un résultat en dB, si le résultat obtenu est supérieur ou égal à 35dB on estime que le tatouage est robuste. L'intégration se fera avec la formule suivante :

$$PSNR = 10 * \log 10\left(d^2 / EQM\right)$$

avec EQM correspondant à l'erreur quadratique moyenne définie par la formule suivante :

$$EQM = (1/mn) * \sum_{0}^{m-1} \sum_{0}^{n-1} \left(Io(i,j) - Ir(i,j)\right)^2$$

Passons à présent à l'explication du code que nous avons déployé pour réaliser ce projet.

II- Etapes de développement

Nous avons cherché dans la phase de développement à suivre les étapes qu'on a décrit dans les schémas bloc de l'étude préliminaire. Pour cela, on a procédé étape par étape, pour être sûr de notre code et comprendre l'ensemble des fonctions, point à point.

Après avoir chargé une image dans Scilab grâce à notre fonction, on a pu travailler sur la matrice. Cette fonction permet de convertir l'image en matrice. Il y a également une option qui permet de convertir une image couleur en image en nuances de gris. C'est ce que nous avons utilisé : Nous n'avons travaillé qu'avec des images en nuances de gris.

```
function matriceImage=chargerImage(path, isRGB)
  if isRGB == 0 then
    matriceImage = double(imread(path));
  else
    matriceImage = double(rgb2gray(imread(path)));
  end
endfunction
```

Une fois notre image sous forme matricielle on a pu chercher a appliquer notre première étape, le passage d'une image en image fréquentielle. Pour cela, on réalise dans un premier temps une transformée de Fourier.

A) Application FFT

On utilise la fonction de Scilab FFT. Cette fonction permet de récupérer une matrice de nombres complexes. Ces nombres contiennent donc les informations de modules et phases de notre image. Dans notre cas, on va chercher à appliquer notre tatouage à notre module, et laisser notre phase inchangée, pour pouvoir reconstituer l'image à la fin. Pour cela, nous avons créé une fonction qui permet de calculer la phase et le module :

```
ImageFreq = fft(image);

[module,phase] = complexe(ImageFreq);

function [mod, phase]=complexe(matrice)

    [N,M] = size(matrice)

    for x = 1:N

        for y = 1:M

            mod(x,y) = abs(matrice(x,y))

            phase(x,y) = atan(imag(matrice(x,y)),real(matrice(x,y)))

        end

   end

endfunction
```

Maintenant que nous avons le module de notre image, nous pouvons créer un tatouage pour chercher à l'appliquer. Nous avons choisi un filtre très basique, qui nous permet de tester nos différentes fonctions. Ce filtre comporte une symétrie centrale :

```
Tatouage = [1,1,1,1,1,1,1;

            1,0,1,1,1,0,1;

            1,1,1,1,1,1,1;

            0,1,1,1,1,1,0;

            1,1,1,1,1,1,1;

            1,0,1,1,1,0,1;

            1,1,1,1,1,1,1 ]
```

B) Application du tatouage

La première chose à faire est de dimensionner notre tatouage à la taille de notre image. Le rendu n'est pas forcement très esthétique, il s'agit de 6 gros rectangles noirs. L'objectif qu'on s'est fixé est de mettre en place nos fonctions, sans regarder le côté « tatouage utile ». On utilise pour cela la fonction de scilab imresize. On lui donne en paramètres notre tatouage, un coefficient alpha et comment on veut l'agrandir. Dans notre cas, alpha correspond au ratio entre le nombre de lignes du tatouage et le nombre de lignes de notre image. Vu qu'on travaille avec des images carrées, le ratio est le même pour les colonnes. On choisit un agrandissement « area » pour garder les proportions.

```
[N,M] = size(image);
[R,C] = size(tatouage);
alpha = N/R;
img_tatouage = imresize(tatouage,alpha,'area');
```

Maintenant que notre image et notre tatouage font la même taille, nous pouvons appliquer un fftshift a notre module et le multiplier point à point par notre tatouage. Il faut ensuite appliquer un second fftshift, pour recentrer correctement les valeurs.

```
image_marquee = fftshift(module);
image_marquee = image_marquee .* img_tatouage;
image_marquee = fftshift(image_marquee);
```

C) Reconstitution de l'image tatouée

La dernière étape consiste à reformer notre image à partir du « module tatoué » et de la phase de l'image initiale. On applique pour cela la formule de trigonométrie qui lie le module et la phase grâce a une exponentielle. Il ne reste ensuite qu'a faire une transformée inverse de Fourier pour reconstituer notre image pour pouvoir la visualiser.

```
cplx = image_marquee .* exp(%i .* phase);
image_marquee = fft(cplx,1);
```

Tout au long du code, il est possible de visualiser « l'image fréquentielle » pour vérifier que les différentes étapes sont correctes. On peut aussi grâce à ça vérifier que le tatouage est bien présent dans notre image. On applique la fonction calcule module pour recalculer le module. On utilise également la fonction « afficheImage » pour afficher les images de façon classique lorsqu'elles ne sont pas en fréquence

```
function mod=calculmodule(matrice)
  [N,M] = size(matrice)
  for i=1:N
    for j=1:M
      mod(i,j) = log(matrice(i,j) +1)
    end
  end
endfunction

function afficherImage(matriceImage)
  imshow(uint8(matriceImage));
endfunction
```

La dernière partie que l'on a mis en place consiste à calculer le PSNR, comme expliqué dans la partie précédente, pour connaitre la robustesse de notre tatouage :

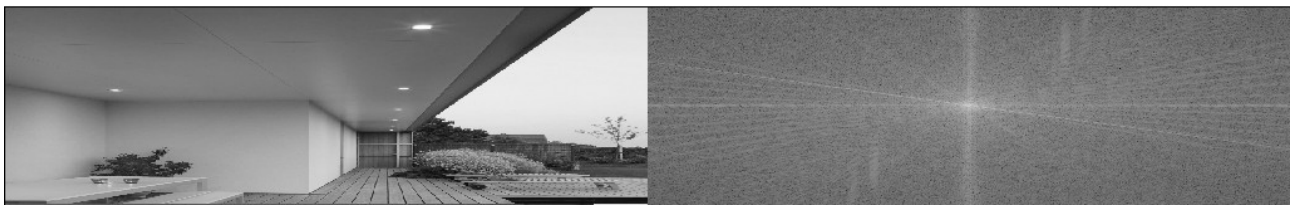
```
function res=PSNR(image, image_res)
  res = 10*log10( (255^2)/(mean((image_res-image).^2)) );
endfunction
```

Vous trouverez dans la partie suivante l'ensemble de nos résultats obtenus et tests effectués.

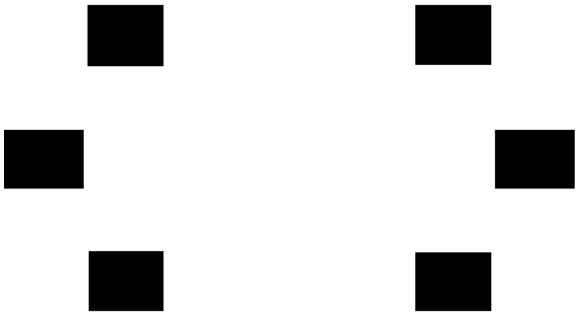
III- Résultats obtenus et robustesse du tatouage

Afin de tester nos différents algorithmes, nous avons choisi une image aléatoirement sur internet, puis appliquer un marquage facilement repérable et également robuste. Voici l'image de départ :

Cette image n'a subit à priori aucune modification externe avant l'application de notre code. Si l'on représente notre image sous forme fréquentielle, nous obtenons le résultat suivant :



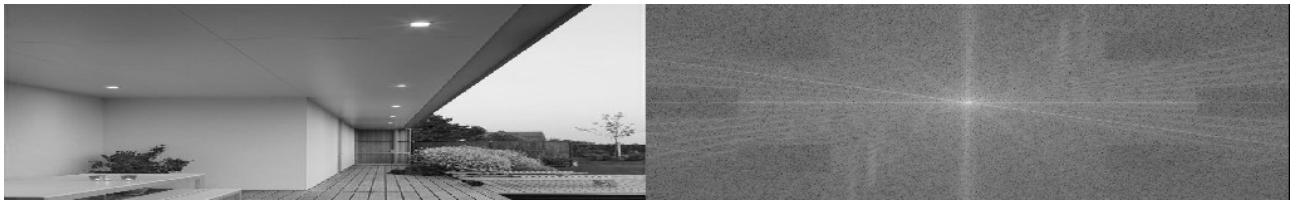
On peut remarquer un point central blanc, plus intense qu'aux alentours ce qui signifie que le contenu de notre image est représenté par ces fréquences (au centre) donc si l'on venait à modifier les fréquences en ce point, notre image serait probablement déformée. Nous allons donc intégrer le tatouage suivant à notre image fréquentielle précédente. Voici une illustration d'une tatouage que nous souhaitons intégrer :



Après intégration sur notre image, nous obtenons le résultat suivant et constatons qu'effectivement notre tatouage ne modifie pas la photo d'origine en apparence en tout cas…

La capture est un peu étirée mais le tatouage appliqué réellement est bien celui présenté ci-avant. Avec une image tatouée de la sorte, nous obtenons un PSNR équivalent à 38,74 dB ce qui est convenable en terme de robustesse, cependant le tatouage est largement visible, cela laisse penser assez aisément que le tatouage pourrait être retirer assez rapidement.

Enfin nous avons mis à l'épreuve notre image tatouée en l'envoyant sur la célèbre plateforme des réseaux sociaux Facebook. Comme nous pouvons le constater ci-après, la compression de l'image à détérioré notre tatouage, cependant le PSNR reste correct avec une valeur de 35,80 dB.



Conclusion

Ce projet, qui consiste à tatouer une image, nous a permis d'approfondir nos connaissances au niveau du langage Scilab et nous a également permis de mieux comprendre le fonctionnement du traitement d'une image de façon générale.

On a apprécié la mise en application d'un tatouage sur une image. Cette application nous montre bien que même avec un tatouage qui semble très visible a l'œil, il peut passer inaperçu après des traitements. Cela nous a permis de bien comprendre les enjeux des watermark, qui permettent d'identifier le propriétaire d'une image. Cela pose aussi des questions de transmissions d'informations de façon discrète, comme par des procédés de stéganographie.

On a également pu se rendre compte du temps de traitement que ce procédé demande. Aussi, pour marquer des images a grande échelle, ou pour marquer un grand nombre d'images, il faudrait des ordinateurs bien plus performants.

Ce projet était très intéressant dans le cadre de nos études, mais aussi de nos futurs emplois.