PERCEPTION, MANIPULATION ET PROTECTION D'IMAGE

TATOUAGE NUMERIQUE

Marie Guénon / Arnaud Tanguy

Table des matières

Signal pseudo-aléatoire

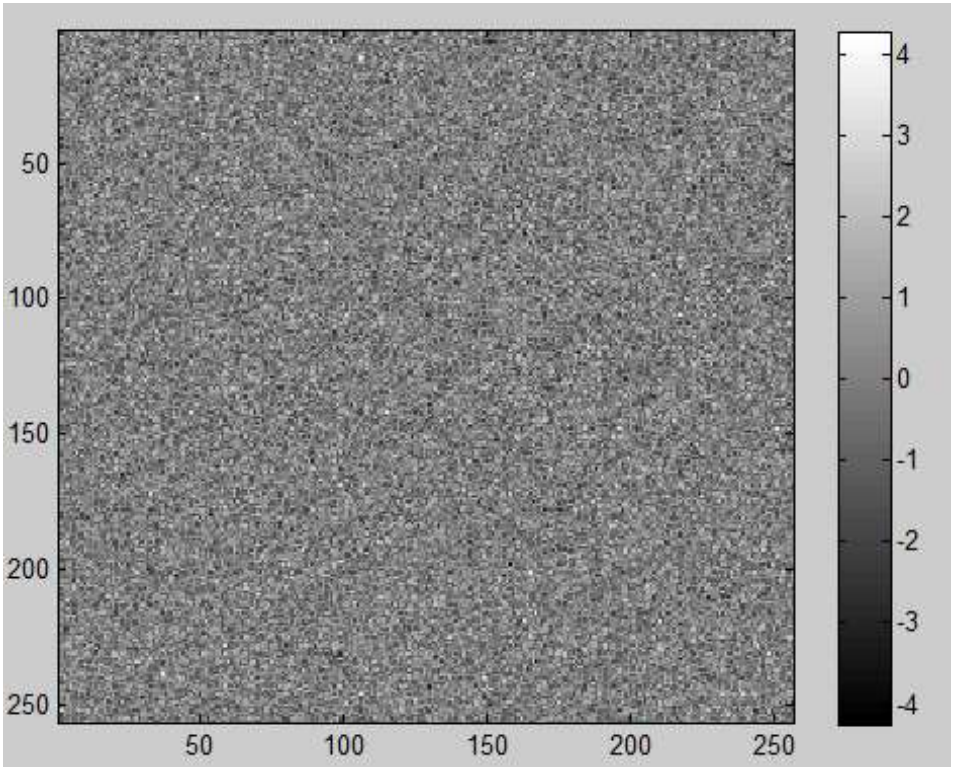
Le but est ici de créer un signal pseudo-aléatoire, c'est-à-dire un bruit blanc, de la taille de l'image que l'on va traiter (ici 256x256) suivant une loi gaussienne centrée réduite $\mathcal{N}(0,1)$. De plus ce bruit blanc doit pouvoir être recréé à partir d'une clé secrète K.
Pour cela, nous avons utilisé les fonctions de matlab *rng* et *randn* :

```
img = imread('le_cervin.png');
[w h] = size(img);
```

```
%% Question 1
% save random seed as key
key = rng;
% generate random noise
u = randn(w, h);

imagesc(u); colormap('gray'); colorbar;
```

Grâce à cela, matlab enregistre l'initialiseur de l'algorithme de génération (*seed*), et c'est ce chiffre qu'il réutilise ensuite pour générer le bruit blanc. Et on obtient ce genre de résultats :



Génération d'une marque

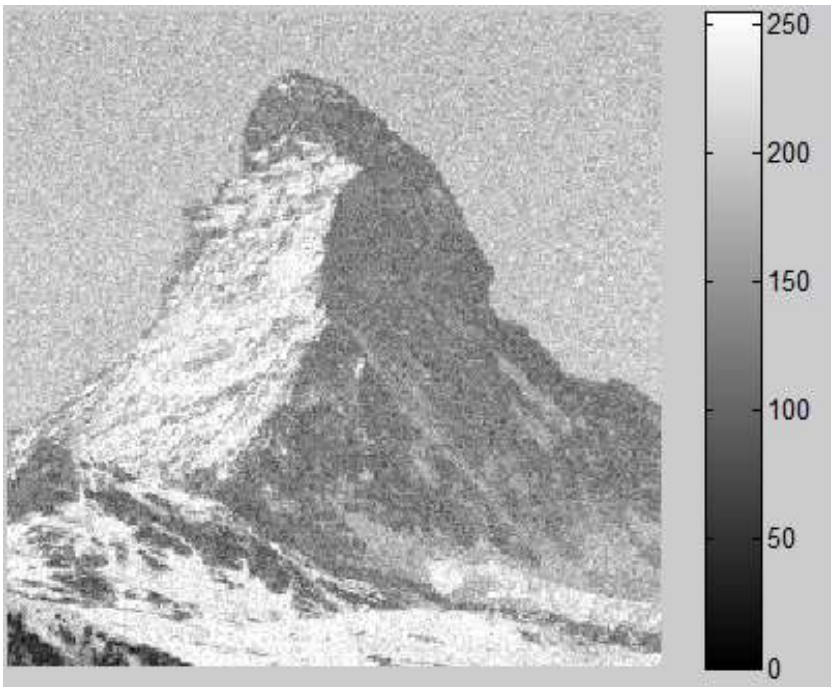
Nous cherchons maintenant à générer une marque ω permettant de dissimuler le bit caché et à l'insérer dans l'image. Pour cela, nous utilisons une modulation du type :

$$s(b) = \gamma * (-1)^b$$

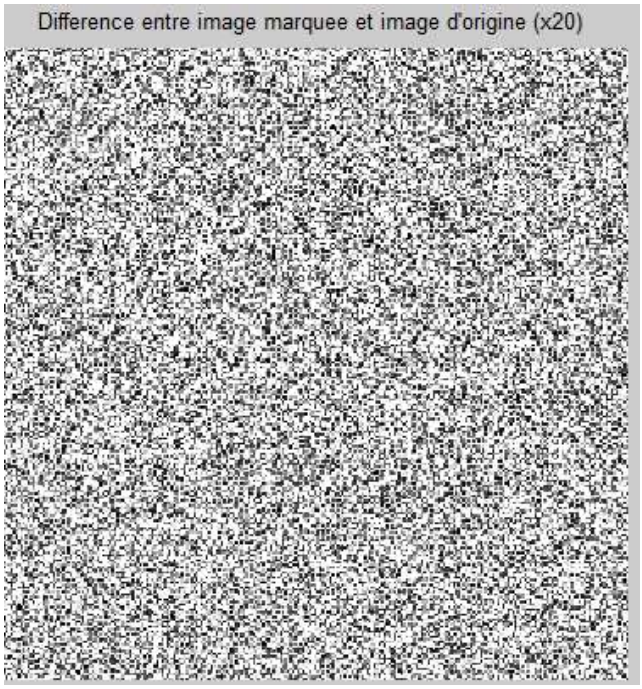
Où *b* est le bit à dissimuler et γ une constante positive.
Une fois appliqué à une image *img*, cela nous donne la formule :

$$image_{marquee} = image_{origine} + \gamma * u * (-1)^b$$

Dans notre cas, nous avons pris $\gamma = 1$, *b* le bit que nous voulons cacher dans l'image (ici, nous avons pris *b=0*) et *u* le bruit blanc que nous avons généré à l'étape précédente. Nous obtenons alors une image du Cervin légèrement bruitée :



De plus, même si ici la différence entre l'image d'origine et l'image marquée n'est pas très visible, celle-ci existe tout de même : (image d'origine à gauche, image marquée à droite et distance entre les deux images en dessous)


Difference entre image marquee et image d'origine (x20)

Module de décodage

Le but est ici de détecter s'il y a une bit de caché dans l'image et si c'est le cas de donner sa valeur. Pour cela, nous avons utilisé la formule suivante :

$$coeff\_correlation = \frac{1}{nombre\ pixels} * \sum_{i \in pixels} \langle image\_origine(i) \mid bruit(i) \rangle$$

Cette formule nous donne un coefficient à comparer à un seuil, afin d'obtenir des informations sur le bit caché. Dans notre cas, si la comparaison nous renvoie -1, il n'y a pas de bit caché, sinon la comparaison nous renvoie 0 ou 1, la valeur du bit :

$$sortie = \begin{cases} 1 & si\ coeff\_correlation > seuil \\ -1 & si\ -seuil \leq coeff\_correlation \leq seuil \\ 0\ sinon \end{cases}$$

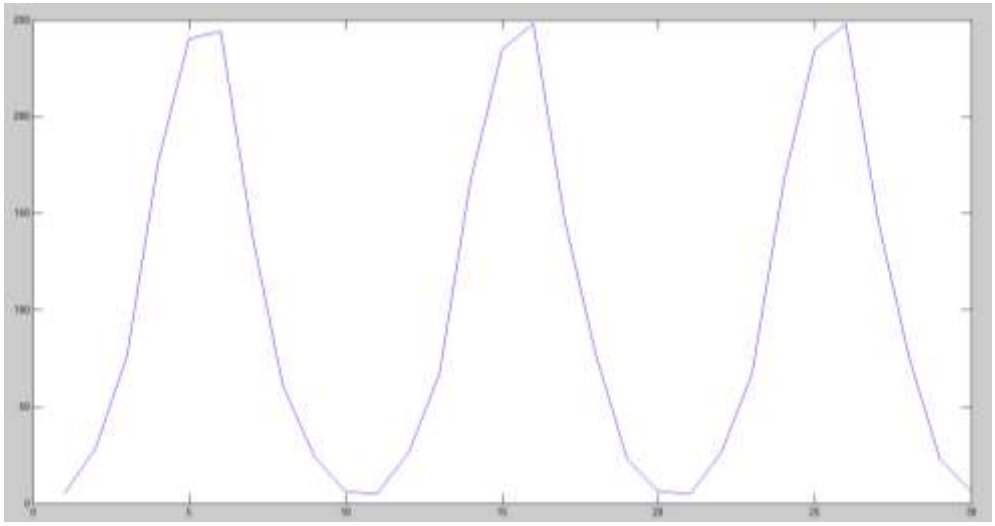
Ce qui nous donne l'algorithme :

```
function det = detecteur(key, gamma, threshold, img)
    rng(key);
    % regenerate the same noise
    u = gamma .* double(randn(size(img)));
    % Coefficient de correlation
    corr = sum(dot(img, u))/(size(img,1)*size(img,2))
    if corr > threshold
        det = 1;
    elseif ((corr > threshold) & (corr <= threshold))
        det = -1;
    else
        det = 0;
    end
end
```

Pour effectuer nos tests, nous avons fixé de manière arbitraire $\gamma = 10$ et $seuil = 10$. En effet, nous avons remarqué que sur quelques tests, ces paramètres nous donnaient une sortie cohérente.

Décodage

Le but est ici d'obtenir graphique avec trois pics, représentatifs du nombre de tirages obtenus avec une valeur de corrélation donnée. De plus, ces pics sont révélateurs ou non d'une watermark :



A gauche on a l'histogramme avec un watermark à 0, au milieu sans watermark et à droite avec un watermark à 1. DE plus, nous avions pris $\gamma = 10$ ce qui est constaté sur le seuil entre les deux premiers pics. (la courbe a été décalée par matlab, c'est pourquoi elle n'est pas centrée en 0).

```
for i=1:1000
    key = rng;
    w0 = watermark(img, 0, key, gamma);
    w1 = watermark(img, 1, key, gamma);
    [corr0, det] = detecteur(key, gamma, 10, w0);
    [corr1, det] = detecteur(key, gamma, 10, w1);
    [corr_n, det] = detecteur(key, gamma, 10, img);
    corr0_m(i) = corr0;
    corr1_m(i) = corr1;
    corrn_m(i) = corr_n;
end
figure; plot([hist(corr0_m), hist(corrn_m), hist(corrn_m)]); legend('w0', 'img', 'w1'
```