Watermarking

Ch. Dossal

Décembre 2008

Ce TP a pour objectif de présenter différentes méthodes d'insertion d'information cachée dans une image numérique. Il existe de nombreuses situations où l'on peut utiliser un watermark. Nous nous placerons dans une situation où nous voulons insérer une information cachée et la retrouver sans connaissance préalable du contenu original et sans connaisance de la marque.On connait quand même le mécanisme d'insertion, la clé de codage et sa taille. La problématique est celle du fingerprinting.

Le problème majeur qui se pose dans l'ajout d'une marque cachée est la robustesse à différentes attaques. En effet, insérer une marque invisible qui peut être détruite par une modification mineure du contenu est aisé. En revanche insérer une marque robuste aux attaques, c'est à dire qui résiste aux attaques qui laisse le contenu dans un état acceptable est plus difficile. Nous nous restreindrons ici à une attaque simple par ajout de bruit (blanc et gaussien).

Nous utiliserons deux watermarks de référence, un watermark court une image 12×9 pixels avec deux niveaux de gris et un watermark long, une image 20×50 avec aussi deux niveaux de gris.

Pour charger des images nous utiliserons la commande *imread* et nous convertirons tout au format *double* de matlab pour les calculs.
Pour chacun des programmes, il est conseillé d'afficher les images originale et watermarkée ainsi que la différence entre les deux.
Ce TP est largement inspiré du travail de Chris Shoemaker : http://www.vu.union.edu/ shoemakc/watermarking/

1 Générateurs pseudo aléatoires

Le watermarking utilise des marques qui sont sensées être aléatoires. Pour le watermarking, il est donc important de savoir utiliser les générateurs pseudo aléatoires.

1. (a) A l'aide de la commande randn générer deux fois la chaine pseudo aléatoire (CPSA) de dix réels.
   (b) Ecrire un programme qui génère deux fois la même suite de 1000 CPSA différentes de taille 64 et ne prenant pour valeur que des 1 et des -1.
   (c) A l'aide de la transformée de Walsh, comment peut-on générer n vecteurs *aléatoires* de $\{-1,1\}^{2^j}$ avec $n < 2^j$, orthogonaux deux à deux ?

2 Watermark par correlation

Nous proposons ici d'insérer le watermark, bit par bit. Chaque bit étant codé par une suite aléatoire.

2. Une première méthode consiste a associer un block de l'image à chaque bit. Plus le watermark est court plus le block peut être grand.
   Nous considèrerons pour simplifier des blocks carrés.

   (a) Réaliser un programme qui insère un watermark par ajout d'une unique chaine pseudo aléatoire $CPSA$ sur chacun des blocks. Le watermark est supposé être une matrice de taille $N1 \times N2$ composées de 1 et de -1.

   ```
   >>IW=InsersionWatermark(I,W,CPSA,k)
   ```

   Où IW est l'image watermarquées, W le watermark, $CPSA$, la chaine pseudo aléatoire et k un paramètre qui définit l'intensité du watermark. Plus k est grand plus le watermark est visible, plus il est petit, moins il est détectable.

   (b) Ecrire un programme qui détecte et affiche le watermark par correlation à partir de IW, $CPSA$, $N1$ et $N2$. Arrive t-on a détecter parfaitement le watermark pour de petites valeurs de k ? Pourquoi ?

   (c) Cette approche a un défaut majeur : Le watermark est tellement souvent présent dans l'image qu'on peut l'estimer simplement à partir de l'image bruitée en utilisant l'autocorrelation de l'image.
   Ecrire un programme similaire qui code chaque bit par des CPSA différentes sans les stocker toutes.

   (d) Ecrire un programme qui effectue la détection.

   (e) Tester ces différents programmes avec des données non watermarkées, des données watermarkées et des données watermarkées et bruitées. Faire varier le niveau de bruit, la valeur de k.

3. Le watermark global.

   On peu aussi coder chacun des bits sur l'ensemble de l'image. Chaque bit est alors associé à une CPSA.

   (a) Ecrire un programme qui insère un watermark global sur toute l'image en utilisant pour chaque bit une CPSA différente, sans les stocker toutes.

   (b) Ecrire un programme qui détecte et affiche ce watermark par correlation.

3 Codage par transformées

3.1 Utilisation de la DCT

La DCT, transformée en cosinus discrète est un outil classique de traitement d'image. Elle est à la base du codage Jpeg. En Jpeg, chaque image est découpée en blocks 8×8 et une DCT est effectuée sur chaque block. On ne conserve alors que les plus grands coefficients de la transformée. Si l'on insère le watermark directement dans la DCT on peut espérer que le watermark résiste à une compression jpeg. De plus en insérant le watermark sur certaines fréquences on peut limiter l'impact visuel du watermark sur les zones homogènes.

Le plus souvent le watermark est inséré sur des fréquences moyenne.

4. Watermark par correlation

(a) Ecrire un programme qui construit une CPSA dont la dct (commande dct2 de matlab) est à support sur les fréquences moyennes.

(b) Ecrire un programme qui insère un watermark dans une image, où chaque bit du watermark est codé par une CPSA différente à support sur les fréquences moyenne d'un bloque 8×8.
Comment peut on améliorer le procédé si moins de la moitié des carrés 8×8 sont utilisés.

(c) Ecrire un programme qui détecte un tel watermark et le tester en présence de bruit.

5. Watermark par comparaison de coefficients. On peut également coder un bit en modifiant deux coefficients de la DCT. On choisit deux coefficients c_1 et c_2 pour chacun des carrés 8×8 et on modifie éventuellement les coefficients de manière à ce que 1 soit codé par $c_1 > c_2$ et 0 $c_2 > c_1$. En pratique on choisit des coefficients qui correspondent à des fréquences proches et qui sont quantifiées de la même manière dans jpeg pour resister à la compression jpeg. Pour assurer une meilleure robustesse on peut aussi remplacer la condition $c_1 > c_2$ par $c_1 > c_2 + k$.

(a) Ecrire un programme qui insère un watermark suivant cette procédure, le programme doit prendre k en paramètre.

(b) Ecrire le programme correspondant qui détecte le watermark. Le tester en présence de bruit et avec différentes valeur de k.

4 Transformée en ondelettes

La transformée en ondelettes est également un outil permettant le watermarking.

6. On peut procéder d'une manière proche du watermaking en DCT en codant chaque bit par une CPSA ajoutés aux coefficients d'ondelettes. On va alors d'abord transformer l'image en ondelettes puis effectuer un watermark global sur une ou plusieurs sous-bandes de coefficients. En pratique on choisit prioritairement les 3 sous-bandes associées aux hautes fréquences.

(a) Ecrire un programme qui insère un watermark sur les trois sous-bandes associées aux échelles d'ondelettes les plus fines par ajout de CPSA.

(b) Ecrire le programme associé de détection et le tester en présence de bruit. Etudier l'influence de l'ondelette choisie sur les artefacts créés par le watermark, en particulier comparer des ondelettes régulières et Haar.

Cette approche a l'inconvénient d'insérer des artefacts dans des parties de l'image homogènes ce qui les rend plus visibles. On peut aussi utiliser la transformée en ondelettes pour *cacher le watermark* dans des zones où il est moins visible.

(a) On peut utiliser les ondelettes pour définir un masque qui permettra de localiser le watermark en dehors des zones homogènes : On dira qu'un pixel p_1 est dans une zone homogène si tous les support des ondelettes de plus petite échelles au dessus d'un seuil T n'intersecte pas p_1.
Ecrire un programme qui à partir d'un seuil T et d'une image I renvoie le masque (une image valant 0 ou 1) associé à la zone homogène.

(b) Ecrire un programme de watermark global qui utilise le masque.

(c) Ecrire le programme d'extraction de watermark associé et le tester en présence de bruit en faisant varier l'intensité du marquage et la valeur de T.

(d) On peut aussi essayer d'insérer une information dans les coefficients d'ondelettes par comparaison comme on l'a fait avec la DCT ou en utilisant uniquement les coefficients d'ondelettes de forte amplitude.
Ecrire un programme qui sur chaque colonne de coefficients d'une sous bande échange éventuellement l'ordre des deux coefficients de plus grand amplitude de manière à coder 1 si le plus grand est plus haut dans la colonne et -1 s'il est plus petit.

(e) Ecrire le programme effectuant le décodage.
Comment rendre ce codage un peu plus robuste au bruit ?
On pourrait utiliser des ensembles autres que des colonnes en utilisant des ensemble pseudo aléatoires de manière à assurer que sur chaque ensemble il y ait de fort coefficients.