ENSIMAG MMIS 2A : Multimédia

TP de tatouage d'image

François Cayre – francois.cayre@grenoble-inp.fr

Buts

- développer un système complet de tatouage numérique pour l'image
- comprendre le codage BPSK
- comprendre les enjeux du masquage psycho-visuel
- appréhender l'apport de l'utilisation de l'information adjacente à l'encodage
- comprendre les prolèmes de désynchronisation
- élaborer une attaque d'effacement du tatouage

Logiciels

- Ce TP utilise Libit comme librairie de manipulation des vecteurs et matrices en C ;
- Assurez-vous que vous disposez des paquets Debian suivants : `gcc`, `make`, `cvs`, `libjpeg-progs`, `gimp`
- Vous êtes libres d'utiliser votre langage de script préféré pour automatiser ce qui doit l'être.

Préparation

Installation de Libit

1- Préparez le répertoire :

```
$ mkdir ~/packages && cd packages
```

2- Récupérez Libit sur le serveur CVS (pressez Entrée si on vous demande un mot de passe) :

```
$ cvs -d:pserver:anonymous@libit.cvs.sf.net:/cvsroot/libit login
$ cvs -z3 -d:pserver:anonymous@libit.cvs.sf.net:/cvsroot/libit co -P libit
```

3- Installez Libit (utilisez `--prefix=<rep>` pour spécifier où vous voulez l'installer) :

```
$ cd libit/ && ./configure --prefix=<rep> && make && make install && cd
```

Installation du code du TP

1- Téléchargez l'archive :

```
$ wget www.balistic-lab.org/pub/ens/wm/wmlab.tgz
```

2- Extrayez les fichiers :

```
$ tar -xzvf wmlab.tgz && cd wmlab
```

Contenu des répertoires du code du TP

- `cfg/` : fichiers de configuration pour le codeur et le décodeur
- `images/` : quelques images PGM pour les tests
- `include/` : fichiers d'en-tête
- `src/` : code source (là où vous devrez travailler !)

Compilation, nettoyage, exécution

Compilez avec :

```
$ make
```

Nettoyez avec :

```
$ make clean
```

Exécutez le codeur avec :

```
$ ./embedder cfg/embed.cfg
```

Exécutez le décodeur avec :

```
$ ./decoder cfg/decoder.cfg
```

Description du schéma de tatouage proposé

- Codage : BPSK;
- Enfouissement : ISS (*Improved Spread Spectrum*) [1] et Maximum Linear Correlation [2].

Une image `I` de taille `MxN` pixels est divisée en tuiles carrées de taille `SZxSZ` [3]. Chaque tuile est tatouée avec le même message. On obtient un vecteur 1D à partir d'une tuile en utilisant la fonction `mat_to_vec`. La réciproque est effectuée avec la fonction `vec_to_mat`. Un paramètre de sur-échantillonnage (qui doit être une puissance de 2) permet de sur-échantilloner une matrice en répétant chaque coefficient dans un carré. On utilise ce sur-échantillonnage uniquement sur les matrices représentant les porteuses. Au décodage, l'estimation du signal de tatouage est basée sur un filtre à la Wiener (ce débruiteur simple est implanté dans la fonction `mat_estimate`).

Questions

1- Codage et décodage BPSK

1.1- Rappelez en quoi consiste le codage BPSK.

1.2- Donnez l'expression du nombre maximum de bits `Nc` que l'on peut cacher en fonction de la taille `SZ` du côté d'une tuile carrée, et du paramètre de sur-échantillonnage `S`. Indice : les contraintes de variance unitaire et de moyenne nulle pour toutes les porteuses imposent une restriction sur le nombre de vecteurs linéairement indépendants.

1.3- Dans le fichier `ss.c`, complétez la fonction `get_correlations` pour effectuer le décodage.

1.4- Dans le fichier `modulation.c`, complétez la fonction `encode_ss` destinée à effectuer le codage BPSK. Ne faites pas attention au paramètre `wcr`, la distorsion d'enfouissement est calculée ailleurs en utilisant le PSNR cible.

2- Masquage psycho-visuel

Le masquage psycho-visuel consiste en la variation de la puissance locale d'enfouissement du signal de tatouage, en fonction de la sensibilité de l'œil humain. L'activité locale autour d'un pixel est modélisée par la variance locale autour de ce pixel. Un filtre de Sobel est généralement utilisé pour détecter les contours. Vérifiez le fichier `masking.c` pour les détails. Le masquage dit `custom` est simplement un lissage de la sortie du filtre de Sobel.

2.1- Dans le fichier `cfg/embed.cfg`, faites varier le paramètre de masquage psycho-visuel. Discutez les résultats en fonction des caractéristiques locales de l'image (contours, textures, aplats).

2.2- Dans le fichier `cfg/embed.cfg`, faites varier le paramètre de sur-échantillonnage. Discutez son influence.

2.3- Trouvez la valeur du PSNR cible (fixé dans `cfg/embed.cfg`) qui assure l'invisibilité du tatouage (pour vous !) pour chaque masquage psycho-visuel. Discutez l'influence du masquage psycho-visuel sur la puissance effective du signal de tatouage.

3- Paramétrage et utilisation de l'information adjacente à l'enfouissement

3.1- Faites varier le paramètre de sur-échantillonnage dans l'ensemble {1,2,4}. Discutez en fonction de la compression JPEG. Utilisez le script d'attaque pour obtenir une image compressée / décompressée avec le facteur de qualité JPEG voulu. Fixez le paramètre de sur-échantillonnage à sa valeur optimale en fonction du compromis robustesse / imperceptibilité.

3.2- Tracez les performances du schéma de tatouage en fonction de la compression JPEG. Utilisez plusieurs images, plusieurs clefs et plusieurs messages pour lisser les résultats.

Indices :
- Trouvez d'abord les valeurs extrémales du facteur de qualité JPEG ;
- Les paramètres des fichiers de configuration peuvent être fixés en ligne de commande (utile pour les scripts).

3.3- Construisez un encodeur utilisant l'ISS (Improved Spread Spectrum) : remplacez `modulation.o` par `imodulation.o` dans `src/Makefile`. Tracez les performances en robustesse de ce nouvel encodeur en fonction de la compression JPEG. Discutez la pertinence de l'utilisation de l'information adjacente à l'encodeur.

4- Problèmes de désynchronisation

4.1- Effectuez un fenêtrage de l'image tatouée. Essayez de décoder le message. Expliquez ce qui se passe.

4.2- Implantez une stratégie simple pour résister au fenêtrage. Mettez à jour la fonction `get_correlation` dans le fichier `ss.c` de votre implantation.

4.3- Utilisez la fonction `mat_acf2d`, calculez la fonction d'autocorrélation de l'estimation du signal de tatouage (dans le fichier `decoder.c`). Recommencez avec une image tatouée qui a subi une rotation. Utilisez `mat_pgm_write` après `mat_scale` pour écrire l'image sur le disque. Imaginez (sans l'implanter) une stratégie permettant de résister à des transformations géométriques plus générales.

5- Attaques plus subtiles

5.1- Implantez une attaque spécifiquement destinée à effacer le tatouage, fondée sur le débruitage. Trouvez le PSNR moyen pour une attaque réussie.

5.2- Utilisez la structure périodique du signal de tatouage, implantez une attaque d'effacement du signal de tatouage. Tracez, en terme de PSNR, les performances de cette attaque en fonction du nombre de tuiles à disposition de l'adversaire pour l'estimation du signal de tatouage.

Bibliographie

[1] H. Malvar and D.A.F. Florêncio. *"Improved Spread Spectrum: A New Modulation Technique for Robust Watermarking"*. *IEEE Transactions on Signal Processing*, 51(4):898-905. Special Issue on Signal Processing for Data Hiding in Digital Media and Secure Content Delivery, 2003.

[2] M.L. Miller, I.J.Cox and J.A. Bloom, *"Informed Embedding: Exploiting Image and Detector Information During Watermark Insertion"*, Proc. 7th *IEEE Int. Conf. on Image Processing*: Special Session on Second Generation Watermarking Methods, Vancouver, Canada, Vol. III, pp. 1-4, 2000.

[3] F. Deguillaume, S. Voloshynovskiy and T. Pun, *"Method for the Estimation and Recovering from General Affine Transforms in Digital Watermarking Applications",* SPIE Photonics West, Electronic Imagin, Security and Watermarking of Multimedia Contents IV, San Jose, CA, USA, 2002.