

Projet Traitement des signaux et données : WaterMarking

Martin Michotte

Olivier Niyonkuru

Morgan Valentin

Patrick Tchoupe

Trésor Tekadam

08/12/2020

Introduction	3
Répartition des tâches	3
Nos solutions	4
Watermarking sur des images	4
Ajout du watermark	4
Extraction et vérification du watermark	4
Utilisation	5
Librairie utilisées	5
Pistes d'améliorations	5
Watermarking sur des sons	6
Ajout du watermark	6
Extraction et vérification du watermark	7
Utilisation	7
Librairie utilisée	8
Pistes d'améliorations	8
Conclusions	9
Général	9
Martin Michotte	9
Patrick Tchoupe	9
Olivier Niyonkuru	10
Morgan Valentin	10
Trésor Tekadam	10

1. Introduction

Un watermark ou tatouage numérique est un signal ou une information ajoutée à un signal source afin de pouvoir authentifier celui-ci sans pour autant le modifier de manière trop importante.

Le but de ce projet est donc de pouvoir :

1. "watermarquer" un signal.
2. Donner le signal tatoué à une tierce personne afin qu'il le modifie ou non.
3. Analyser le signal tatoué récupéré afin de pouvoir dire s'il est authentique (non-modifié) ou non.

Un signal pouvant être une multitude de choses, nous nous sommes concentrés uniquement sur les images et les sons.

2. Répartition des tâches

Étant donnée que nous avons deux types de signaux à traiter, nous avons créé deux groupes :

- [images](#) : P.Tchoupe, M.Valentin, T.Tekadam
- [sons](#) : M.Michotte, O.Niyonkuru

Notons que même si nous avons travaillé en deux groupes distincts, nous avons régulièrement organisé des réunions afin de présenter aux autres le travail effectué ainsi que de discuter d'éventuelles modifications/améliorations à apporter.

3. Nos solutions

3.1. Watermarking sur des images

3.1.1. Ajout du watermark

- 1) Récupération de l'image d'entrée.
- 2) Récupération de l'image à tatouer.
- 3) Vérification des tailles des deux images, si elles ne sont pas égales le reste des étapes ne sera pas exécuté.
- 4) Conversion de l'image à tatouer en N/B et ensuite en Binaire.
- 5) Suppression des LSB de l'image d'entrée (cela n'a pas d'impact sur le rendu de l'image)
- 6) Remplacement des bits supprimés par ceux de l'image à tatouer.
- 7) Ecriture de l'image résultante dans l'arborescence (fichier WatermarkedImage).

3.1.2. Extraction et vérification du watermark

- 1) Récupération de l'image.
- 2) Extraction des LSB de l'image, ceux-ci devraient constituer l'image qui a été tatouée.
- 3) Comparaison des bits extraits avec l'image tatouée à la base (comparaison bit par bit). Tout logiquement,
 - a) ✓ Si les images sont égales alors on considère l'image comme **authentique**.
 - b) ✗ Si les images ne sont pas égales, alors on considère l'image comme étant **non-authentique**.

3.1.3. Utilisation

- 1) Ouvrez le dossier [wm-images](#) dans [MatLab](#).
- 2) Ajouter le dossier [src](#) ainsi que tous ses sous-dossier au [Path](#):
 - a) Clic-droit sur [src](#) > [Add to Path](#) > [Selected Folders and Subfolder](#)
 - b) **OU** : exécutez la cmd suivante : `addpath(genpath('src'));`
- 3) Afin de **watermarker** un signal:

```
addWatermark('chemin/vers/votre/imageDeBase','chemin/vers/votre/image_A_Tatouer')
```

- 4) Libre à vous ensuite de modifier (ou non) l'image résultante.
- 5) Afin de **vérifier l'authenticité** d'une image tatouée:

```
checkIntegrity('chemin/vers/votre/Image','chemin/vers/votre/WatermarkDeBase')
```

NOTE : Des exemples d'utilisation sont disponibles dans le fichier [test.m](#).

3.1.4. Librairie utilisées

Nous n'avons pas utilisé de librairies pour réaliser cette partie.

3.1.5. Pistes d'améliorations

- Effectuer le tatouage sans tenir compte le taille du watermark : Ici l'idée serait de redimensionner les deux images fournies par l'utilisateur et effectuer le watermarking avec ces nouvelles dimensions
- Watermarker une image avec du texte, texte qui pourrait être le nom de l'auteur de l'image pour des soucis de droits d'auteurs.

3.2. Watermarking sur des sons

△ Nécessite la [communications toolbox](#) en plus des toolbars prévues pour ce cours.




3.2.1. Ajout du watermark

- 1) Récupération du [signal d'entrée](#).
- 2) Création d'un watermark ([wm](#)) couvrant la totalité de la durée du signal et avec une fréquence de 0.5Hz ($< 20\text{Hz}$) (soit inaudible).
- 3) Ajout d'un préfix contenant la longueur initiale du signal en binaire sur le [wm](#).
- 4) Application d'un filtre passe-haut sur le [signal d'entrée](#) (afin de ne garder que les fréquences audibles).
- 5) Ajout du [wm](#) dans le [signal d'entrée](#).
- 6) Écriture du signal dans un nouveau fichier.

△ **Conditions:** le signal ne peut pas durer plus de **40h**!

La durée du signal converti en micro-secondes étant stockée sur 40 bits, il a fallu limiter la durée maximale du son afin d'éviter un buffer-overflow.

3.2.2. Extraction et vérification du watermark

- 1) Récupération du [signal d'entrée](#).
- 2) Extraction et décodage du préfix afin de connaître la longueur du son initial.
 - a)  Si cette étape échoue , le signal d'origine a été modifié. Il est alors considéré comme étant **non-authentique**.
- 3) Re-crédation du watermark avec les configurations standard et les données décodées.
- 4) Application d'un filtre passe-bas sur le [signal d'entrée](#) afin d'en extraire le [watermark](#).
- 5) Corrélation entre le [watermark original](#) et le [watermark extrait](#) du signal.
 - a)  Si le taux de corrélation est ≥ 0.98 alors on considère le signal comme étant **authentique**.
 - b)  Si le taux de corrélation est < 0.98 alors on considère le signal comme étant **non-authentique**.

3.2.3. Utilisation

- 1) Ouvrez le dossier [wm-sounds](#) dans [MatLab](#).
- 2) Ajouter le dossier [src](#) ainsi que tous ses sous-dossier au [Path](#):
 - a) Click-droit sur [src](#) > [Add to Path](#) > [Selected Folders and Subfolder](#)
 - b) **OU** : exécutez la cmd suivante : [addpath\(genpath\('src'\)\);](#)
- 3) Afin de **watermarker** un signal:

```
addWatermark('chemin/vers/votre/signal.wav')
```

- 4) Libre à vous ensuite de modifier (ou non) ce signal avec des outils comme [Audacity](#) ou autre.
- 5) Afin de **vérifier l'authenticité** d'un signal: [bash](#)
[checkIntegrity\('chemin/vers/votre/signal2.wav'\)](#)

NOTE : Des exemples d'utilisation sont disponibles dans le fichier [tests.m](#).

3.2.4. Librairie utilisée

Aucune librairie externe n'a été utilisée.

3.2.5. Pistes d'améliorations

- Pouvoir ajouter un watermark contenant de l'information tel que le nom de l'auteur de son ou autre.
- Ne pas avoir de contraintes de durée sur le son.

4. Conclusions

4.1. Général

Bien qu'initialement nous n'avions aucune idée sur la manière d'implémenter un watermark sur du son ou sur des images, après quelques recherches il s'est avéré qu'il existait plusieurs techniques utilisant différents algorithmes mathématiques plus ou moins complexes. Afin d'éviter d'utiliser ces algorithmes dont on ne comprenait pas l'entièreté ou pour lesquelles on n'avait pas spécialement le temps de les comprendre, nous avons décidé de tenter d'implémenter le watermarking à notre propre manière.

Nous avons dès lors complètement séparé le watermarking sur image et le watermarking sur le son et utilisé deux techniques différentes. Nous pensons être arrivés à un résultat concluant même si nous savons pertinemment que nos techniques ne sont pas optimales et peuvent très certainement être améliorées.

4.2. Martin Michotte

Ce projet m'a permis d'apprendre le fonctionnement réel du watermarking, chose qu'auparavant je connaissais mais ne comprenais pas. Le fait de devoir l'implémenter par nous mêmes et avec nos propres moyens était très intéressant et m'a permis de parfaire mes connaissances en traitement de signaux.

Avoir des consignes très vagues et une "carte blanche" quant à la réalisation du projet m'a initialement un peu perturbé. En effet, il m'était assez difficile de voir à quel point il fallait creuser la matière afin de satisfaire la demande du professeur. Encore à l'écriture de ce rapport, je ne sais toujours pas dire si notre projet est trop léger, satisfaisant ou trop complexe.

4.3. Patrick Tchoupe

Au début, je me disais que ce projet allait être compliqué à réaliser car je ne savais pas réellement ce qu'il fallait réaliser au niveau du watermarking. Mais après notre premier entretien avec le professeur, les tâches à réaliser étaient plus claires.

Les recherches et documentations sur le sujet du watermarking sur une image m'ont permises de découvrir de nouvelles notions très intéressantes telle que la stéganographie, qui est l'art de la dissimulation d'un message dans un autre et le choix a été fait d'utiliser la méthode des bits de poids faible.

4.4. Olivier Niyonkuru

Personnellement, je ne voyais pas comment commencer ce projet car j'étais pas à l'aise avec les technologies qu'on pensait utiliser (python et matlab) pour au final réaliser le watermarking. J'espère que les compétences développées durant ce projet me serviront dans ma vie professionnelle et/ou dans des projets personnels.

4.5. Morgan Valentin

J'ai passé pas mal de temps à comprendre ce qui était demandé de réaliser au niveau du watermarking car je n'avais jamais entendu parlé d'injection d'image dans une autre image. Mais finalement, après quelques réunions entre nous, j'ai vite compris le principe.

Le seul point négatif de ce projet, c'est qu'on à l'habitude à l'ephec, d'avoir des projets très détaillés et encadrés par les professeurs. Le fait que le projet soit libre m'a causé pas mal de recherches supplémentaires non prévues qui, avec le charge de travail provenant des autres cours (labo de sécurité à rendre chaque 2 semaines, projet d'intégration, projet de base de données,...) était compliqué à gérer. Heureusement qu'on était 5 sur ce projet.

4.6. Trésor Tekadam

Ce projet m'a permis d'apprendre beaucoup de choses sur les images et le son en général. Jusqu'à présent j'ai eu à faire du traitement d'images à l'aide des outils tels que photoshop sans vraiment comprendre ce qui se passe vraiment. Mais à travers ce projet j'ai maintenant une idée de comment les choses se passent derrière la scène

le fait de n'avoir aucune restriction sur la manière de procéder m'a permis de beaucoup me documenter sur le sujet et d'en apprendre davantage.

5. Sources

- <https://medium.com/intrasonics/hiding-data-in-sound-c8db3de5d6e0>
- https://nl.mathworks.com/matlabcentral/fileexchange/64535-lsb-watermarking?s_tid=answers_rc2-1_p4_BOTH
- https://fr.wikipedia.org/wiki/St%C3%A9ganographie#Usage_des_bits_de_poids_faible_d'une_image