INFO-F101 – Programmation
Projet 4
Introduction à la stéganographie

Année académique 2014–2015

Introduction

Une image peut-être décomposée en un tableau de points élémentaires appelés *pixels* (abréviation de *picture element*). Supposons que nous ne manipulions que des images en *niveaux de gris*, c'est-à-dire, des images dont les « couleurs » sont uniquement des nuances (plus ou moins foncées) de gris. On peut représenter une telle image par une matrice d'entiers, dont la valeur des éléments représente l'intensité lumineuse des pixels de l'image. Par conséquent, un traitement d'image peut être réalisé en manipulant la matrice qui la représente.

Fichier PGM

Un fichier PGM (*Portable Gray Map*) est un fichier composé de deux parties : un en-tête et une matrice dont chaque élément représente le niveau de gris d'un pixel de l'image. L'en-tête contient 3 lignes. La première ligne est toujours la même, il s'agit de l'identifiant d'un fichier PGM : `P2`. La seconde ligne contient 2 entiers : le nombre de colonnes et le nombre de lignes de pixels de l'image. Enfin, la troisième ligne contient la valeur maximale que peut prendre un pixel (cette valeur représente le blanc). Les lignes qui suivent donnent l'intensité de chaque pixel. Ceux-ci sont énumérés ligne par ligne et colonne par colonne. Ainsi la première valeur est l'intensité du pixel du coin supérieur gauche, la seconde est l'intensité du pixel de la première ligne, deuxième colonne, *etc*.

Par exemple, un fichier `ulb.pgm` contenant les informations ci-dessous est l'encodage d'une image représentant « ULB ».

```
P2
13 7
255
10  10  10  10  10  10  10  10  10  10  10  10  10
10  88  10  88  10  88  10  10  10  88  88  10  10
10  88  10  88  10  88  10  10  10  88  10  88  10
10  88  10  88  10  88  10  10  10  88  88  10  10
10  88  10  88  10  88  10  10  10  88  10  88  10
10  88  88  88  10  88  88  88  10  88  88  10  10
10  10  10  10  10  10  10  10  10  10  10  10  10
```

Contenu du fichier `ulb.pgm`.

De nombreux outils existent pour visualiser ces fichiers, les mots clés à taper dans un moteur de recherche pour les trouver, sont : `pgm viewer` . Dans les salles informatiques, vous pouvez utiliser le *visualisateur d'image* pour ouvrir et imprimer des images `.pgm`.

Stéganographie : un exemple simple de codage et décodage

Contrairement à la cryptographie, qui chiffre des messages de manière à les rendre incompréhensibles, la stéganographie (en grec "l'écriture couverte") consiste a cacher des messages dans un support, par exemple des images. Vous allez réaliser dans ce projet un programme qui permettra de cacher ou d'accéder à une information cachée dans une image qui vous est donnée dans fichier au format PGM.

L'idée est de voir un procédé simplifié de stéganographie. Celui-ci part du principe que si une image PGM est codé avec 256 niveau de gris (la valeur maximale que peut prendre un pixel est 255) l'oeil humain ne perçoit pas de différence si les valeurs des pixels sont très légèrement modifiées. Nous pouvons donc jouer avec cela pour cacher de l'information dans cette image.

Concrètement, prenons deux images PGM de même dimension. Une (l'image source) dont la valeur maximale que peut prendre un pixel est 255, et l'autre image (l'image code) dont les valeurs des pixels sont dans $\{0, 1\}$ (donc "0" sera le noir et "1" sera le blanc). Nous allons modifier les pixels de l'image source de telle sorte que les pixels qui ont un niveau de gris paire dans l'image source correspondent (au niveau de leurs positions) aux pixels noirs de l'image code et les pixels qui ont une valeur impaire dans l'image source correspondent aux pixels blancs de l'image code.

Exemple

Voici une illustration du procédé peut-être plus parlante. Reprenons l'image `ulb.pgm` et supposons que nous voulons cacher dedans le message « Hi ». Il faut créer un fichier PGM (de même taille que `ulb.pgm`) avec des valeurs de gris pour les pixels qui sont 0 ou 1 comme le montre le fichier `hi.pgm` ci-dessous.

```
P2
13 7
1
0  0  0  0  0  0  0  0  0  0  0  0  0
0  1  0  0  1  0  0  1  0  0  0  0  0
0  1  0  0  1  0  0  0  0  0  0  0  0
0  1  1  1  1  0  0  1  0  0  0  0  0
0  1  0  0  1  0  0  1  0  0  0  0  0
0  1  0  0  1  0  0  1  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0
```

Contenu du fichier `hi.pgm`.

Ensuite, nous modifions le fichier `ulb.pgm` de telle sorte que les pixels noir dans le fichier `hi.pgm` soient associés à des valeurs paires dans le nouveau fichier `ulb.pgm` et que les pixels blancs dans le fichier `hi.pgm` soient associés à des valeurs impaires dans le nouveau fichier `ulb.pgm` comme illustré dans la fichier de résultats ci-dessous : `res.pgm`. La valeur des pixels est donc cachée dans le bit de poids faible du nombre (ici, nous nous cantonnerons à des modifications de maximum une unité sur les valeurs des pixels). Attention à ne pas avoir pour les pixels, des valeurs négatives ou supérieurs au niveau maximum définit dans l'entête.

```
P2
13 7
255
10  10  10  10  10  10  10  10  10  10  10  10  10
10  89  10  88  11  88  10  11  10  88  88  10  10
10  89  10  88  11  88  10  10  10  88  10  88  10
10  89  11  89  11  88  10  11  10  88  88  10  10
10  89  10  88  11  88  10  11  10  88  10  88  10
10  89  88  88  11  88  88  89  10  88  88  10  10
10  10  10  10  10  10  10  10  10  10  10  10  10
```

Contenu du fichier `res.pgm`.

Le codage est donc terminé, pour décoder, il suffit de créer un nouveau fichier PGM dont l'entête sera `P2 13 7 1`, ensuite de prendre la matrice des pixels du fichier `res.pgm` et d'écrire un `0` à la place des nombres paires et `1` à la place des nombres impaires. Nous retombons ainsi sur le fichier `hi.pgm`.

Fonctions à écrire

On vous demande d'écrire les fonctions suivantes :

- Une fonction `info_image(nom_image)`,
  qui reçoit un nom de fichier PGM dans `nom_image` sous forme d'une chaîne de caractères, et lit les informations contenues dans l'entête du fichier. Cette fonction doit s'assurer qu'on lui transmet bien un fichier PGM (si ce n'est pas le cas elle affichera une erreur), ensuite elle renverra un tuple composé de l'identifiant du fichier PGM, la largeur de l'image, la hauteur de l'image et la valeur maximale que peut prendre un pixel.
- Une fonction `charger_image(nom_image)`,
  qui reçoit un nom de fichier PGM dans `nom_image` sous forme d'une chaîne de caractères, et lit l'image contenue dans ce fichier. La fonction doit recopier les valeurs des pixels de l'image dans une matrice $h \times l$ avec h et l étant respectivement la hauteur et la largeur de l'image traitée. Une fois le travail effectué, la fonction retournera cette matrice.
- Une fonction `encodage(source,code,res)`,
  qui reçoit trois noms de fichiers PGM sous forme de chaînes de caractères : `source`,`code` et `res`. Respectivement le nom d'une image "source" dans laquelle vous allez dissimuler un code, le nom d'une image "code" qui contiendra l'information à cacher et le nom d'une image résultat qui contiendra (après l'exécution de la fonction) le résultat du codage. La fonction devra donc crée une image PGM "résultat" à partir de `source` et `code`, qui visuellement ressemblera à `source` mais contiendra l'image `code` par le procédé d'encodage décrit dans la section précédente.
- Une fonction `decodage(crypt,claire)`,
  qui reçoit deux noms de fichiers PGM sous forme de chaînes de caractères : `crypt`, et `claire`. Respectivement le nom d'une image sensée contenir une information cachée et le nom d'une image qui contiendra (après l'exécution de la fonction) l'information dissimulé. La fonction devra donc crée une image PGM "claire" à partir de `crypt`, qui contiendra le secret caché par le procédé de décodage décrit dans la section précédente.

Informations supplémentaires importantes

Pour tester votre projet, il vous est fourni plusieurs images :

- `math.pgm`
- `hitchhiker.pgm`
- `kubrick.pgm`

Après réalisation de votre projet vous devez être capable de dissimuler le contenu du fichier `hitchhiker.pgm` dans `math.pgm`. De plus, l'image `kubrick.pgm` contient un message caché que vous devez pouvoir extraire par les méthodes décrites ci-dessus. Il vous est demandé également au début de votre code d'indiquer en commentaire le contenu de ce message secret.

Pour rappel, il faut pouvoir gérer les erreurs qui vous semblent naturelles à l'aide des exceptions. Par exemple si le fichier source n'existe pas ou bien si le fichier ouvert n'est pas un fichier PGM.

Votre projet sera testé à l'aide des commandes suivantes :

- `python3 projet4.py codage image1.pgm image2.pgm image3.pgm`.

qui cachera `image2.pgm` dans `image1.pgm`, le résultat sera écrit dans le fichier pgm `image3.pgm`.

- `python3 projet4.py decodage image1.pgm image2.pgm`.

qui exhibera l'image cachée dans `image1.pgm`, le résultat sera écrit dans le fichier pgm `image2.pgm`.

Donc votre programme devra récupérer les paramètres dont il a besoin en ligne de commande (à l'aide de la librairie `sys`).

Exemple

Voici un exemple d'utilisation en ligne de commande du script terminé.

```
python3 projet4.py codage math.pgm hitchhiker.pgm resultat.pgm
Fichier source :  math.pgm
Fichier code :  hitchhiker.pgm
Fichier destination :  resultat.pgm
Codage: travail terminé !


python3 projet4.py decodage kubrick.pgm claire.pgm
Fichier contenant le message :  kubrick.pgm
Fichier message decodé :  claire.pgm
Décodage: travail terminé !
```

Morale de l'histoire

Une image peut en cacher une autre...

Consignes pour la remise du projet

Les consignes pour la remise du projet sont disponibles en ligne sur la page du cours sur l'Université Virtuelle. Ces consignes sont à respecter *scrupuleusement* ; relisez-les attentivement avant la remise !

Pour toutes les questions **importantes** concernant l'énoncé, nous vous invitons à vous adresser à Keno Merckx (kmerckx@ulb.ac.be)

Date limite de remise : Lundi 8 décembre 2014 à 13h.