Licence 1 CMI 2020-2021

Projet d'initiation à l'ingénierie

Analyse d'images et stéganographie

Frédéric Lavancier

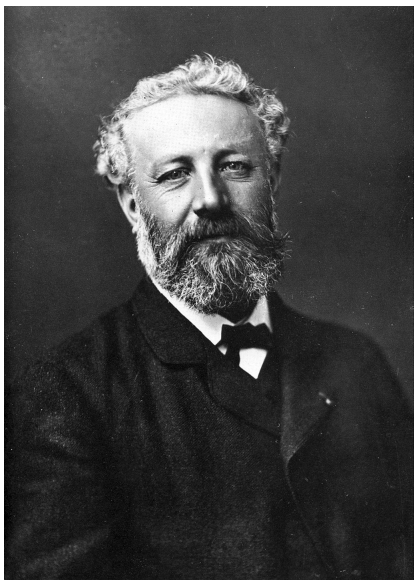
1 Description générale

L'objectif de ce projet est de développer (sous Python) quelques fonctions d'analyse d'images. Il s'agira dans un premier temps de clarifier ce qu'est une image pour un ordinateur, que ce soit une image en noir et blanc ou une image en couleur. Dans un second temps, quelques transformations basiques seront mises en oeuvre, par exemple : modification du niveau de gris (ou de la couleur) d'une image, filtrage pour débruiter l'image, transformation d'une grande image en une série de petites images de type "photomaton". Enfin on s'intéressera à la stéganographie, qui est (dixit wikipedia) "l'art de la dissimulation : son objet est de faire passer inaperçu un message dans un autre message [en l'occurence une image dans une autre image]. Elle se distingue de la cryptographie, « art du secret », qui cherche à rendre un message inintelligible à autre que qui-de-droit." Cette technique sera appliquée pour créer une fonction sous Python capable de cacher une image dans une autre, et une autre fonction capable de retrouver l'image dissimulée dans une autre image.

Les images utiles à ce projet sont téléchargeables depuis :
`http://www.math.sciences.univ-nantes.fr/~lavancie/enseignement.html`

2 Manipulations d'images sous Python

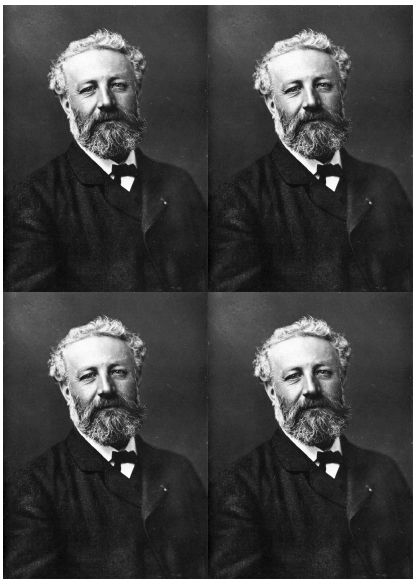
On commence par traiter la photo en noir et blanc d'un illustre Nantais.



1. Lire l'image sous Python et l'afficher.
2. En quel type d'objet Python l'image a-t-elle été importée ? Quelles sont ses dimensions ? Comment sont encodés les niveaux de gris ?

3. Créer une fonction qui permet d'inverser le niveau de gris de l'image. Le résultat doit être l'image ci-dessous

4. Effet photomaton : écrire une fonction qui permet de créer 4 mini-images de l'image originale, comme ci-dessous à gauche. La taille de l'image totale doit être identique à la taille de l'image originale (à une ligne ou colonne près). En réitérant cette procédure, on obtient l'image de droite.

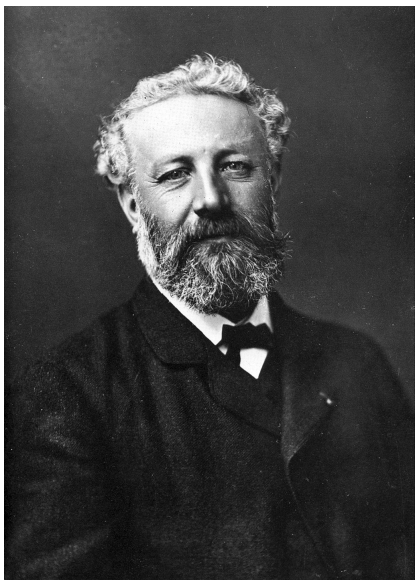
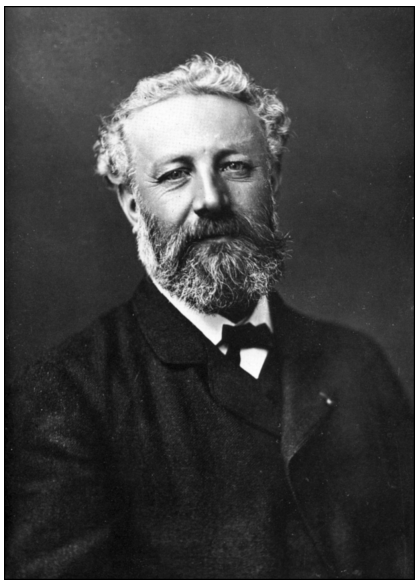


5. On peut débruiter l'image (ou de façon équivalente la lisser) en appliquant un filtre. Cela consiste à remplacer chaque pixel par une moyenne pondérée de ses pixels voisins. Appliquer ce principe à l'image originale en considérant un voisinage de ± 2 pixels (ce qui représente pour chaque pixel une sous-image de taille 5×5 centrée sur le pixel à remplacer) et en moyennant à l'aide des poids suivants :

$$\frac{1}{100}\begin{pmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{pmatrix}$$

Le résultat de ce débruitage est l'image de droite ci-dessous, qui est moins bruitée que l'image originale (à gauche), ce qui se vérifie en zoomant sur les images. Que se passe-t-il si on réitère ce lissage un grand nombre de fois (par exemple 10 fois ou 50 fois) ?

Essayer de lisser l'image en utilisant une autre matrice de poids.

6. Reprendre les questions précédentes pour une image en couleur.

3 Stéganographie

1. Les images de Nantes ci-dessous cachent d'autres illustres personnalités Nantaises. Qui sont-elles ? Indication : ils sont cachés dans les 4 derniers bits de chaque pixel. Ecrire une fonction capable de décoder les images.


2. Le premier chapitre d'un roman mythique est caché dans l'image ci-dessous. De quel roman s'agit-il ? (Il faut ici regarder de près le dernier bit de chaque pixel, chaque caractère étant codé en ASCII en base 2). Ecrire une fonction capable de décoder l'image.

3. Ecrire une fonction capable de cacher une image dans une autre, comme cela a été fait ci-dessus.
4. Ecrire une fonction capable de cacher un texte dans une image, comme cela a été fait ci-dessus.

4 Installer Python

On peut soit utiliser Python en ligne, par exemple avec Google Colab, ou l'installer sur sa machine.

4.1 En ligne avec Google Colab

Il suffit d'aller à l'adresse `https://colab.research.google.com/`.

Sélectionner "Nouveau Notebook" la première fois que vous vous y connectés.

Vous accédez alors à un notebook dans lequel vous pouvez taper votre code dans des "cellules". Une fois votre code saisi, il suffit de cliquer sur la flèche "lecture" à gauche de la cellule pour exécuter la cellule. Le résultat du code s'affiche. Vous pouvez modifier la cellule et recommencer ou créer une nouvelle cellule vide dessous en cliquant sur "+Code" en haut de la fenêtre.

Exemple : taper dans la cellule 2+2 puis exécuter la.

Vous pouvez notamment charger des fichiers depuis une page web.

Exemple : pour charger l'image "JulesVerne.jpg" sur le site "site.fr" :

```
import imageio
image= imageio.imread("https://www.site.fr/JulesVerne.jpg")
```

La première ligne sert à pouvoir utiliser les fonctions de la libraire `imageio`, ici `imageio.imread`. On peut ensuite afficher l'image :

```
import matplotlib.pyplot as plt
plt.imshow(image,cmap='gray")
```

La première ligne sert à utiliser les fonctions de la sous librairire `pyplot` de `matplotlib`, que l'on nomme en plus court `plt`. L'option `cmap='gray'` permet d'afficher une image en noir et blanc

Pour accéder à vos fichiers personnels (par exemple vos images) :

- Vous avez besoin d'avoir un compte Google drive
- Placer vos fichiers dans un dossier de votre Google drive
- Il faut lier votre Google drive à Colab : pour cela, dans Colab, cliquer sur l'icône "dossier" à gauche, puis cliquer sur l'icône dossier foncée avec un triangle à l'intérieur ("Installer Drive"). On vous demande de sélectionner votre compte Google et tout se fait ensuite tout seul. Un nouveau dossier "drive" doit alors apparaitre à gauche.
- Pour accéder à votre dossier Google drive dans les codes, le chemin d'accès à écrire est /content/drive/My Drive/

Exemple : si votre dossier dans Google drive se nomme "Projet_L1" et contient l'image "JulesVerne.jpg", vous pouvez la charger dans Python comme ceci :

```
import imageio
image=imageio.imread("/content/drive/My Drive/Projet_L1/JulesVerne.jpg")
```

Puis l'afficher comme précédemment :

```
import matplotlib.pyplot as plt
plt.imshow(image,cmap='gray")
```

4.2 Installation en local

Il y a plusieurs façons de procéder, mais la difficulté est de bien lier les librairies de fonctions disponibles à Python.

Une manière simple qui fait tout tout seul est d'installer Anaconda : `https://www.anaconda.com/products/individual#Downloads`. Anaconda est une interface pour utiliser différents logiciels et langages, dédiés en priorité à la Data Science.

Pour utiliser Python via Anaconda, lancer Anaconda : vous pourrez soit utiliser Spyder ou Jupyter Notebook.

- Sypder offre un environnement simple composé de plusieurs fenêtres. Une fenêtre sert d'éditeur de textes dans lequel on écrit les codes. On peut avoir plusieurs fichiers de codes ouverts en parallèle. On peut exécuter un fichier de codes dans son intégralité ou uniquement une partie (une ligne par exemple), et les résultats apparaissent dans la fenêtre "console". Les images et graphiques apparaissent également dans une fenêtre dédiée.
- Jupyter Notebook permet d'utiliser Python sous forme de Notebook, exactement comme dans Colab (cf la section précédente). Les fichiers Notebook s'ouvrent en format html dans un navigateur internet. On entre les codes dans des cellules que l'on exécute les unes après les autres.

5 Quelques fonctions sous Python

Pour se mettre dans le bon répertoire de travail :

```
import os
os.chdir('/blabla/my_dir')
```

Pour manipuler les images, on a besoin des librairies `imageio`, `matplotlib` et `numpy` :

```
import imageio
import matplotlib.pyplot as plt
import numpy as np
```

Pour ouvrir une image : `image=imageio.imread('JulesVerne.jpg')`
Pour sauvegarder une image : `imageio.imsave('new_image.png', image)`
Pour afficher une image : `plt.imshow(image,cmap='gray')` ou `plt.imshow(image)`

Voici quelques fonctions qui pourront être utiles au projet :
`chr(n)` : permet de transformer l'entier `n` en le caractère correspondant dans l'encodage ASCII
`int(x)` : retourne la partie entière de `x`, le résultat étant un "integer"
`int(s,2)` : convertit la chaîne de caractères binaires `s` (contenant des 0 et des 1) en l'entier correspondant.
`np.sum` : calcule la somme des éléments d'un vecteur ou d'une matrice
`np.reshape` : permet de réorganiser les dimensions d'un "array" (par exemple transformer une matrice en vecteur ou inversement)
`ord(s)` : permet de transformer le caractère `s` en l'entier correspondant dans l'encodage ASCII
`'{0:08b}'.format(n)` : permet d'obtenir l'écriture binaire en 8 bits d'un entier `n` inférieur à 255