Séance 13 :

Transformation des images

Avant toute chose :

- ❏ J'ai crée un dossier `helpers` qui contient les fichiers `poltter.py` et `helper.py` (CF Annexe)
- ❏ J'ai crée un dossier `ressources` qui contient toutes les images de la séance : `image_grise.jpg` `lena.jpg` `original.jpg` `photographer.jpg`
- ❏ A la racine de la séance j'ai créé 3 fichiers `exercice_i.py` avec `i` $\in [\![4, 6]\!]$ qui contiennent les réponses de chaque exercice.
- ❏ Je travaille dans le fichier `main.py` à la racine, dans lequel j'exécute les fonctions.
  Il contient les imports et constantes suivant(e)s :

```python
import numpy as np
import sys
import traceback
import PIL.Image as im
from colorama import Fore, Style
from helpers.plotter import Plotter as pltr
from helpers.helper import pat
from exercice_4 import contraste
from exercice_5 import filtre
from exercice_6 import outline
```

```python
# Images
GREY_CAT = pat('ressources/image_grise.jpg')
LENA = pat('ressources/lena.jpg')
PHOTO = pat('ressources/photographer.jpg')

# Exercice 4
RATE_1 = 2              # contraste rate of the first image
RATE_2 = -2             # contraste rate of the second image

# Exercice 6
THRESHOLD_1 = 100       # contraste rate of the first image
THRESHOLD_2 = 200       # contraste rate of the second image

# exercice nb to exec
EXEC_NB = 6
```

1. Exercice 4

1. ***Définir sous python cette fonction.***
   La faire tracer dans l'intervalle $[0, 1]$ ***par pas de*** 0.01 ***et faire tracer la composée*** $f(f(f(x)))$***.***
   Voici `f` :

```python
def f(x): return 0.5 + 0.5 * np.sin(np.pi * (x - 0.5))
```

Voici sa réciproque :

```python
def f_reciproque(x): return np.arcsin(2 * x - 1) / np.pi + 0.5
```

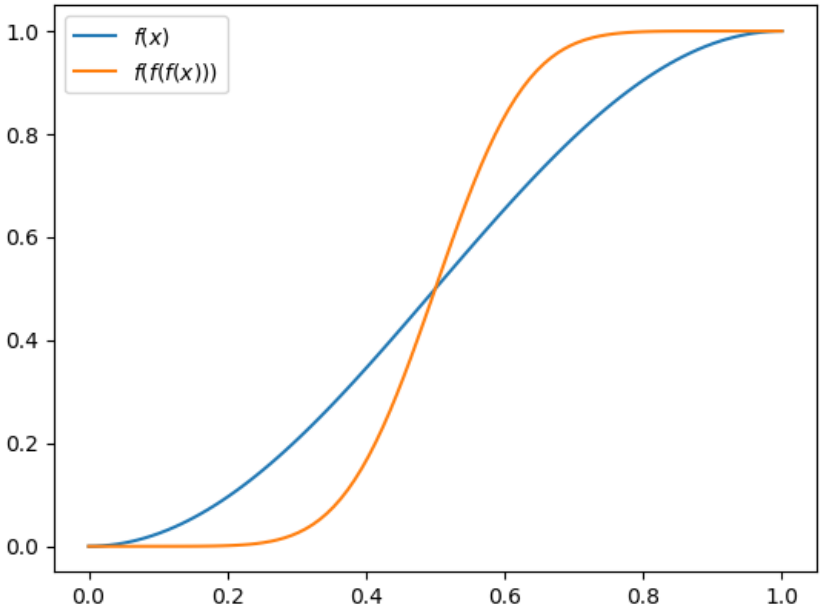
Voici composée $n^{ème}$:

```python
def f_compose(n, x):
    if (n <= 1):
        return f(x)
    return f(f_compose(n-1, x))
```

Je trace tout ça :

```python
def traceF():
    X = np.linspace(0, 1, 100)
    Y1 = [f(x) for x in X]
    Y2 = [f_compose(3, x) for x in X]

    plt.plot(X, Y1, label='$f(x)$')
    plt.plot(X, Y2, label='$f(f(f(x)))$')
    plt.legend()
    plt.show()
```

Et ça donne :



2. ***Définir sous python la fonction*** $g(x) = E(f(f(f(\frac{x}{256}))) \times 256)$ ***avec*** $E(s)$***la partie entière de*** s.
Je le fais pour n intération et non seulement 3.

```python
def _g(f, x, n):
    if (n == 1):
        return f(x/256)
    return f(_g(f, x, n-1))


def g(x, n):
    return _g(f, x, n) * 256 if n > 0 else _g(f_reciproque, x, abs(n))
```

3. ***Ecrire le programme permettant de charger l'image en niveau de girs nommée*** `image_grise.jpg`***, de lui appliquer la fonction*** g ***et de visualiser cette image.***
Je préfère traiter cette question directement avec la suivante.
4. ***Comment diminuer le contraste ? Ecrire cette fonction et la tester.***

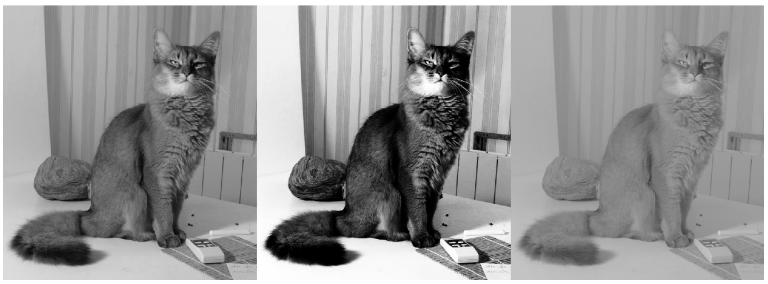
Pour diminuer le contraste, il faut composer plus que 3 fois f. Voici la fonction `contraste` :

```python
def contraste(img, rate):
    return g(img, rate)
```

On teste :

```
# image import
i = np.array(im.open(GREY_CAT))

# plot creation
P = pltr('Exercice 4')

# contrasted images
c1 = contraste(i, RATE_1)
c2 = contraste(i, RATE_2)

# plot images
P.addSubplot(i, "contrast=0")
P.addSubplot(c1, "contrast={}".format(RATE_1))
P.addSubplot(c2, "contrast={}".format(RATE_2))

# show plot
P.show()
```

Ce qui donne (de gauche à droite `c=0`, `c=+2`, `c=-2`) :



2. Exercice 5

1. ***On dispose d'une image en couleur (`koala.jpg`), on souhaite récupérer les pixels rouges dans un tableau et visualiser ce tableau en niveau de gris.***
   Question non traitée (par flemme...)
2. ***Appliquer le flitre ci-dessus à cette image et visualiser ces deux images côte à côte.***

Je définis les matrices des filtres suivantes :

```
REGULAR = np.array([
    [1, 1, 1],
    [1, 1, 1],
    [1, 1, 1]
])

GAUSSIAN = np.array([
    [1, 2, 1],
    [2, 4, 2],
    [1, 2, 1]
])
```

Puis je code la fonction `filtre`. Je n'afficherai le résultat qu'à la question suivante.

```python
def filtre(img, fType='REGULAR'):
    img1 = np.zeros_like(img)
    h, w = np.shape(img)
    f, n = REGULAR, 9
    if fType == 'REGULAR':
        f, n = GAUSSIAN, 16

    for i in range(1, h-1):
        for j in range(1, w-1):
            a = img[i-1: i+2, j-1: j+2]
            img1[i][j] = np.sum((a * f))/n
    return img1
```

3. ***Appliquer le flitre gaussien à cette même image.***

```python
# plot creation
P = pltr('Exercice 5')

# image import
i = np.array(im.open(LENA).convert('L'))

# filtered images
f1 = filtre(i)
f1 = filtre(i, 'GAUSSIAN')

# plot images
P.addSubplot(i, "without filter")
P.addSubplot(f1, "regular filter")
P.addSubplot(f1, "gaussian filter")

# show plot
P.show()
```

Ce qui donne, de gauche à droite : sans flitre, avec le flitre normal, avec le flitre gaussien :

3. Exercice 6

1. ***Appliquer l'algorithme suivant à une image.***
   Voici la fonction `seuil` :

```python
def seuil(greyScaleArray, threshold):
    return np.where(greyScaleArray < threshold, 0, 255)
```

Voici la fonction `outline` :

```python
def outline(img, threshold):
    n, m = np.shape(img)
    Gx = Gy = np.zeros((n-2, m-2))
    for i in range(n-2):
        for j in range(m-2):
            Gx[i][j] = img[i][j+2] - img[i][j-2]
            Gy[i][j] = img[i+2][j] - img[i-2][j]

    N = np.sqrt(Gx ** 2 + Gy ** 2)
    N /= np.amax(N)
    N *= 255
    N = N.astype(int)
    N = seuil(N, threshold)

    return N
```

On teste tout ça :

```python
# plot creation
P = pltr('Exercice 6')

# image import
i = np.array(im.open(PHOTO).convert('L'))

# filtered images
o1 = outline(i, THRESHOLD_1)
o2 = outline(i, THRESHOLD_2)

# plot images
P.addSubplot(i, "original")
P.addSubplot(o1, f"outlined: {THRESHOLD_1}")
P.addSubplot(o2, f"outlined: {THRESHOLD_2}")

# show plot
P.show()
```

Ce qui donne quelque chose de faux... malheuresement.

4. Annexe

Voici le contenu du fichier `ressources/poltter.py`

```python
import matplotlib.pyplot as plt
import numpy as np
import PIL


class Plotter:

    def __init__(self, name=''):
        self.imageToPlot = []
        self.name = name
        plt.rcParams['toolbar'] = 'None'

    def addSubplot(self, image, name='', axis=False):
        if type(image) == PIL.JpegImagePlugin.JpegImageFile:
            image = np.array(image)
        cmap = 'gray' if image[0][0].size == 1 else 'viridis'

        self.imageToPlot.append(ImageToPlot(image, cmap, name, axis))

    def show(self):
        plt.suptitle(self.name, fontsize=22, fontweight=4, color='#2c3e50')
        l = len(self.imageToPlot)
        nbOfColumn = int(np.sqrt(l))
        nbOfLines = l // nbOfColumn
        nbOfLines = nbOfLines if l % nbOfColumn == 0 else nbOfLines + 1

        i = 0
        for img in self.imageToPlot:
            i += 1
            plt.subplot(nbOfColumn, nbOfLines, i)
            plt.imshow(img.image, cmap=img.colorMode)
            if not img.axis:
                plt.axis('off')
                plt.title(img.name, fontweight='bold', color='#27ae60')
                plt.margins(0, 0)
                plt.subplots_adjust(top=1, bottom=0, right=1, left=0,
                                    hspace=0, wspace=0)
            pass

        figManager = plt.get_current_fig_manager()
        figManager.window.showMaximized()
        plt.show()


class ImageToPlot:
    def __init__(self, image, colorMode, name, axis):
        self.image = image
        self.colorMode = colorMode
        self.name = name
        self.axis = axis
```

Voici le contenu du fichier `ressources/helper.py`

```python
import os.path


def pat(p):
    dirname = os.path.dirname(__file__)
    return os.path.join(dirname, '..', p)
```