TD - Transformations des images

C.F. sujet

I - Introduction

C.F. sujet

II - Augmentation du contraste

Exercice 4

1. Coder f(x)

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

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

2. Coder g(x)

```
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. Contraster une image

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

GREY_CAT = 'Ressources/image_grise.jpg'
RATE_1 = 2              # contraste rate of the first image
RATE_2 = -2             # contraste rate of the second image

def Exercice_4():
    # 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()
```

III - Fitrage d'une image

Exercice 5

1. Récupérer les rouges d'une image et les filtrer

non fait par flemme

2. Coder le filtrer d'une image

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

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


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. Filtrer une image

```
LENA = 'Ressources/lena.jpg'

def Exercice_5():
    # 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()
```

Exercice 6

```
PHOTO = 'Ressources/photographer.jpg'
THRESHOLD_1 = 100       # contraste rate of the first image
THRESHOLD_2 = 200       # contraste rate of the second image

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

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

def Exercice_6():
    # 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()
```