4 Images

July 26, 2024

1 Opérations de base, fonction de test

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import PIL.Image as Image
```

[2]:
```
im = Image.open('exemple.png')
```

[3]:
```
im.size
```

[3]:
```
(588, 592)
```

[4]:
```
fig, ax = plt.subplots()
ax.imshow(im)
plt.show()
```

[5]:
```
M = np.asarray(im)
M.shape
```

[5]:
```
(592, 588, 3)
```

[6]:
```
def test(fic, f, new_fic=None):
    im = Image.open(fic)
    M = np.asarray(im)
    M2 = f(M)
    im2 = Image.fromarray(M2)
    fig, ax = plt.subplots(1, 2, figsize=(10, 5))
    ax[0].imshow(im)
    ax[1].imshow(im2)
    if new_fic is not None:
        im2.save(new_fic)
    plt.show()
```

[7]:
```
def Id(M):
    return M
```

[8]:
```
test('exemple.png', Id)
```

2 Niveaux de gris

[9]:
```
def ng(M):
    M2 = np.zeros(M.shape, dtype=np.uint8)
    n, m, k = M.shape
    for i in range(n):
        for j in range(m):
            r, v, b = M[i, j]
            M2[i, j] = r//3 + v//3 + b//3
    return M2
```

[10]:
```
test('exemple.png', ng)
```

[11]:
```
def ng2(M):
    M2 = np.zeros(M.shape, dtype=np.uint8)
    G = M[:,:,0]//3 + M[:,:,1]//3 + M[:,:,2]//3
    M2[:,:,0] = M2[:,:,1] = M2[:,:,2] = G
    return M2
```

[12]:
```
test('exemple.png', ng2)
```

3 Négatif

[13]:
```
def neg(M):
    M2 = np.zeros(M.shape, dtype=np.uint8)
    n, m, k = M.shape
    for i in range(n):
        for j in range(m):
            r, g, b = M[i, j]
            M2[i, j] = 255-r, 255-g, 255-b
    return M2
```

[14]:
```
test('exemple.png', neg)
```

[15]:
```
def neg2(M):
    return 255 - M
```

[16]:
```
test('exemple.png', neg2)
```

4 Seuillage

[17]:
```
def seuil(M, s=120):
    M2 = np.zeros(M.shape, dtype=np.uint8)
    n, m, k = M.shape
    for i in range(n):
        for j in range(m):
```

```
            if M[i, j, 0] > s:
                M2[i, j] = 255, 255, 255
            else:
                M2[i, j] = 0, 0, 0
    return M2
```

[18]:
```
test('exemple.png', seuil)
```

[19]:
```
def seuil2(M, s=10):
    M2 = np.array(M)
    M2[M[:,:,0] > s] = 255
    M2[M[:,:,0] <= s] = 0
    return M2
```

[20]:
```
test('exemple.png', seuil2)
```

5 Floutage

[21]:
```
def flou(M, N=10):
    M2 = np.array(M, dtype=np.uint8)
    n, m, k = M.shape
    for _ in range(N):
        for i in range(1, n-1):
            for j in range(1, m-1):
                r1, g1, b1 = M2[i+1, j]
                r2, g2, b2 = M2[i-1, j]
                r3, g3, b3 = M2[i, j+1]
                r4, g4, b4 = M2[i, j-1]
                M2[i, j] = r1//4 + r2//4 + r3//4 + r4//4, g1//4 + g2//4 + g3//4␣
 ↪+ g4//4, b1//4 + b2//4 + b3//4 + b4//4
    return M2
```

[22]:
```
#test('exemple.png', flou)
```

[23]:
```
def flou2(M, N=20):
    M2 = np.array(M, dtype=np.uint8)
    n, m, k = M.shape
    for _ in range(N):
        M2[1:-1, 1:-1] = M2[2:, 1:-1]//4 + M2[:-2, 1:-1]//4 + M2[1:-1, 2:]//4 +␣
 ↪M2[1:-1, :-2]//4
    return M2
```

[24]:
```
test('exemple.png', flou2)
```

6 Réduction

[25]:
```
def avg(M, i1, i2, j1, j2):
    S = 0
    nb_pix = (i2 - i1)*(j2 -j1)
    for i in range(i1, i2):
        for j in range(j1, j2):
            S += M[i, j]
    return S//nb_pix
```

[26]:
```
def red(M, n2=100, m2=100):
    M2 = np.zeros((n2, m2, 3), dtype=np.uint8)
    n, m, k = M.shape
    fn = n//n2
    fm = m//m2
    for i2 in range(n2):
        for j2 in range(m2):
            M2[i2, j2, 0] = avg(M[:, :, 0], i2*fn, (i2+1)*fn, j2*fm, (j2+1)*fm)
            M2[i2, j2, 1] = avg(M[:, :, 1], i2*fn, (i2+1)*fn, j2*fm, (j2+1)*fm)
            M2[i2, j2, 2] = avg(M[:, :, 2], i2*fn, (i2+1)*fn, j2*fm, (j2+1)*fm)
    return M2
```

[27]:
```
test('exemple.png', red)
```

[28]:
```
def red2(M, n2=33, m2=54):
    M2 = np.zeros((n2, m2, 3), dtype=np.uint8)
    n, m, k = M.shape
    fn = n//n2
    fm = m//m2
    for i2 in range(n2):
        for j2 in range(m2):
            M2[i2, j2] = np.mean(M[i2*fn:(i2+1)*fn, j2*fm:(j2+1)*fm], axis=(0,␣
 ↪1))
    return M2
```

[29]:
```
test('exemple.png', red2)
```

7 Agrandissement

[30]:
```
def agr(M, n2=800, m2=800):
    M2 = np.zeros((n2, m2, 3), dtype=np.uint8)
    n, m, k = M.shape
    for i2 in range(n2):
        for j2 in range(m2):
            M2[i2, j2, 0] = M[i2*n//n2, j2*m//m2, 0]
            M2[i2, j2, 1] = M[i2*n//n2, j2*m//m2, 1]
            M2[i2, j2, 2] = M[i2*n//n2, j2*m//m2, 2]
```

```
    return M2
```

[31]:
```
test('exemple.png', agr)
```

[32]:
```
def agr2(M, n2=800, m2=800):
    M2 = np.zeros((n2, m2, 3), dtype=np.uint8)
    n, m, k = M.shape
    for i2 in range(n2):
        for j2 in range(m2):
            M2[i2, j2] = M[i2*n//n2, j2*m//m2]
    return M2
```

[33]:
```
test('exemple.png', agr2)
```

8 Contours

[34]:
```
def gradi(M):
    Mc = np.array(M, dtype=np.int16)
    Mg = np.zeros(M.shape)
    Mg[1:-1, 1:-1] = Mc[2:, 1:-1] - Mc[:-2, 1:-1]
    return np.array(np.abs(Mg), dtype=np.uint8)
```

[35]:
```
def gradi(M):
    Mc = np.array(M, dtype=np.int16)
    Mg = np.zeros(M.shape)
    n, m, k = M.shape
    for i in range(1, n-1):
        for j in range(m):
            Mg[i, j] = Mc[i+1, j] - Mc[i-1, j]
    return np.array(np.abs(Mg), dtype=np.uint8)
```

[36]:
```
test('exemple.png', gradi)
```

[37]:
```
def gradj(M):
    Mc = np.array(M, dtype=np.int16)
    Mg = np.zeros(M.shape)
    Mg[1:-1, 1:-1] = Mc[1:-1, 2:] - Mc[1:-1, :-2]
    return np.array(np.abs(Mg), dtype=np.uint8)
```

[38]:
```
test('exemple.png', gradj)
```

[39]:
```
def cont(M):
    return gradi(M) + gradj(M)
```

[40]:
```
test('exemple.png', cont)
```

9 Rotation

[41]:
```
from math import sqrt, cos, sin, pi
```

[42]:
```
def R(alpha, u):
    ux, uy = u
    c = cos(alpha)
    s = sin(alpha)
    return c*ux -s*uy, s*ux + c*uy
```

[43]:
```
x, y = u = (2, 1)
x2, y2 = u2 = R(pi/3, u)
fig, ax = plt.subplots()
ax.plot([0, x], [0, y], color='red')
ax.plot([0, x2], [0, y2], color='blue')
ax.axis('equal')
fig.savefig('rotation_vect.png')
```

[44]:
```
def pad(M):
    n, m, k = M.shape
    N = round(max(n, m) * sqrt(2))
    M2 = np.zeros((N, N, 3), dtype=np.uint8)
    mil = N//2
    imin, jmin = mil - n//2, mil - m//2
    imax, jmax = imin + n, jmin + m
    M2[imin:imax, jmin:jmax] = M
    return M2
```

[45]:
```
test('exemple.png', pad, 'pad.png')
```

[46]:
```
def rot(M, alpha=pi/3):
    # padding = on créé une matrice plus grande pour permettre la rotation
    n, m, k = M.shape
    N = round(max(n, m) * sqrt(2))
    M2 = np.zeros((N, N, 3), dtype=np.uint8)
    mil = N//2
    imin, jmin = mil - n//2, mil - m//2
    imax, jmax = imin + n, jmin + m
    M2[imin:imax, jmin:jmax] = M

    # rotation
    C = np.array([mil, mil]) # axe de rotation
    M3 = np.zeros((N, N, 3), dtype=np.uint8)
    for i in range(imin, imax):
        for j in range(jmin, jmax):
            P = np.array([i, j])
            i2, j2 = np.array(C + R(alpha, P - C), dtype=int)
```

```
            M3[i2, j2] = M2[i, j]
    return M3
```

[47]:
```
test('exemple.png', rot, 'rot.png')
```

10 Composition de traitements

[48]:
```
def o(f, g):
    return lambda x:f(g(x))
```

[49]:
```
def O(lif):
    if lif == []:
        return lambda x:x
    else:
        return o(lif[0], O(lif[1:]))
```

[50]:
```
sobel = O([seuil2, cont, ng2, flou2])
```

[51]:
```
test('exemple.png', sobel)
```

11 Histogramme

[54]:
```
def plot_hist(M):
    hR = np.zeros(256)
    hG = np.zeros(256)
    hB = np.zeros(256)
    n, m, k = M.shape
    for i in range(n):
        for j in range(m):
            r, g, b = M[i, j]
            hR[r] += 1
            hG[r] += 1
            hB[r] += 1
    fig, ax = plt.subplots(1, 3, figsize=(10, 2))
    ax[0].plot(hR, color='red')
    ax[1].plot(hR, color='green')
    ax[2].plot(hR, color='blue')
    plt.show()
```

[55]:
```
plot_hist(cont(M))
```

[ ]:
```

```