Traitement d'image

1. Fonctions élémentaires

Question 1.

```
def symetrie(im):
    n, p, _ = im.shape
    im_sym = np.zeros_like(im)
    for i in range(n):
        for j in range(p):
            im_sym[i, j] = im[i, p-1-j]
    return im_sym
```

Question 2.

```
def rotation(im):
    n, p, s = im.shape
    im_rot = np.zeros((n, p, s), dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            im_rot[p-1-j, i] = im[i, j]
    return im_rot
```

Question 3.

```
def negatif(im):
    n, p, _ = im.shape
    im_neg = np.zeros_like(im)
    for i in range(n):
        for j in range(p):
            im_neg[i, j] = [255, 255, 255] - im[i, j]
    return im_neg
```

Question 4.

```
def luminance(p):
    return np.round(0.2126*p[0] + 0.7152*p[1] + 0.0722*p[2])


def niveaudegris(im):
    n, p, _ = im.shape
    im_lum = np.zeros((n, p), dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            im_lum[i, j] = luminance(im[i, j])
    return im_lum
```

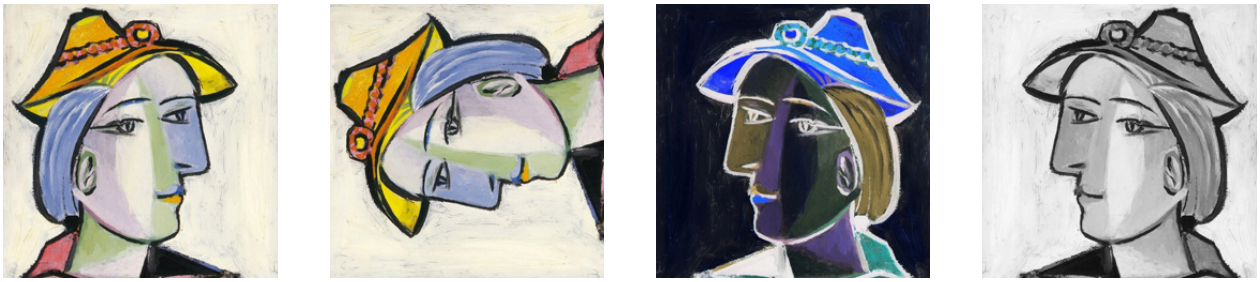

Figure 1 – Le résultat des quatre fonctions ci-dessus sur l'image test.

2. Traitement d'image

Question 5.

```python
def convolution(m, c):
    n, p, q = m.shape
    mm = np.zeros_like(m)
    for i in range(1, n-1):
        for j in range(1, p-1):
            s = 0
            for u in range(3):
                for v in range(3):
                    s += c[u, v] * m[i-1+u, j-1+v]
            for k in range(q):
                if s[k] < 0:
                    mm[i, j, k] = 0
                elif s[k] > 255:
                    mm[i, j, k] = 255
                else:
                    mm[i, j, k] = s[k]
    return mm
```

Question 6.

```python
def lissage(im):
    c = 1/9*np.ones((3, 3), dtype=float)
    return convolution(im, c)

def contraste(im):
    c = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]], dtype=float)
    return convolution(im, c)

def repoussage(im):
    c = np.array([[-2, -1, 0], [-1, 1, 1], [0, 1, 2]], dtype=float)
    return convolution(im, c)
```

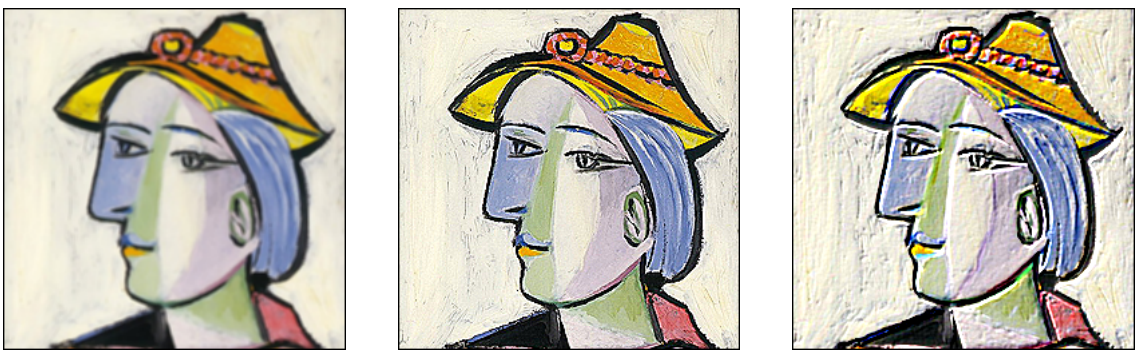

Figure 2 – Le résultat des trois filtres définis ci-dessus sur l'image test.

3. Détection de contours

Question 7.

```
def gradient(m):
    n, p = m.shape
    gr = np.zeros((n, p), dtype=float)
    c1 = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=float)
    c2 = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=float)
    for i in range(1, n-1):
        for j in range(1, p-1):
            s1 = s2 = 0
            for u in range(3):
                for v in range(3):
                    s1 += c1[u, v] * m[i-1+u, j-1+v]
                    s2 += c2[u, v] * m[i-1+u, j-1+v]
            gr[i, j] = s1**2+s2**2
    return gr
```

Question 8.

```
def contour(im, seuil):
    n, p, _ = im.shape
    im_lum = niveaudegris(im)
    gr = gradient(im_lum)
    im_c = np.zeros_like(im_lum)
    for i in range(n):
        for j in range(p):
            if gr[i, j] < seuil:
                im_c[i, j] = 255
    return im_c
```

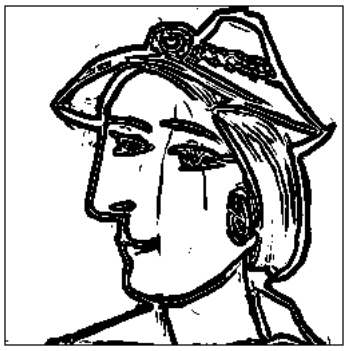

FIGURE 3 – La détection des contours de l'image test. Un lissage a d'abord été appliqué pour réduire le bruit.

4. Stéganographie d'une image

Les deux fonctions demandées sont très simples à écrire lorsqu'on connait les opérateurs de décalage sur les bits : `n >> k` décale les bits de n de k bits vers la droite et `n << k` de k bits vers la gauche.

`n :`	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
`n >> 4 :`	0	0	0	0	a_0	a_1	a_2	a_3
`n << 4 :`	a_4	a_5	a_6	a_7	0	0	0	0

En outre, ces fonctions sont vectorielles, c'est-à-dire qu'elles peuvent s'appliquer à un tableau NUMPY contenant des entiers.

Question 9.

```
def encodage(a, b):
    return ((a >> 4) << 4) + (b >> 4)
```

Question 10.

```
def decodage(c):
    return (c << 4)
```