TP 6- Proposition de solutions

Solution 1 Négatif de l'image

```
def negatif(f_in,f_out):
    f=plt.imread(f_in)
    l,c,d=np.shape(f)
    g=np.zeros((l,c,3))
    for i in range(l):
        for j in range(c):
            g[i,j,:]=np.ones(3)-f[i,j,:3]
    plt.close()
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```

Solution 2 Rotation d'une image

• La nouvelle taille de l'image :

Si (l,c) est la taille de l'image initiale alors les nouvelles dimensions sont (L,C) :

$$L = \sin(a)c + \cos(a)l \qquad C = \sin(a)l + \cos(a)c$$

• Déplacement du point $(0,0)$ en haut à gauche :

Le pixel $(0,0)$ de l'image initial est à la position (x,y) dans l'image transformée avec :

$$(x,y) = \big(\sin(a)c, 0\big)$$

Le pixel (I,J) de l'image transformée est à la position (i,j) dans l'image initiale avec :

$$\begin{pmatrix} i \\ j \end{pmatrix} = \mathrm{Rot}_{(x,y),-a}\left(\begin{pmatrix} I-x \\ J-y \end{pmatrix}\right) = \begin{pmatrix} \cos(-a) & -\sin(-a) \\ \sin(-a) & \cos(-a) \end{pmatrix}\begin{pmatrix} I-x \\ J-y \end{pmatrix}$$

Il convient de vérifier que $i \in [\![0; l-1]\!]$ et $j \in [\![0; c-1]\!]$.

```
def rotation90(f_in,f_out,a):
 a=a/360*2*np.pi # radian
 f=plt.imread(f_in)
 t=np.shape(f)
 L=int(np.sin(a)*t[1]+np.cos(a)*t[0])
 C=int(np.sin(a)*t[0]+np.cos(a)*t[1])
 g=np.zeros((L,C,t[2]))
 x,y=int(np.sin(a)*t[1]),0 # centre de rot
 for I in range(L):
  for J in range(C):
      i=int(round(np.cos(-a)*(I-x)-np.sin(-a)*J))
      j=int(round(np.sin(-a)*(I-x)+np.cos(-a)*J))
      if i<0 or i>=t[0] or j<0 or j>=t[1]:
          g[I,J]=np.array([0,0,0,0])
      else:
                g[I,J][:t[2]]=f[i,j]
                g[I,J][3]=1
 plt.imshow(g)
 plt.axis('off')
 plt.show()
 plt.imsave(f_out,g)
```

Solution 3 Rotation d'un angle quelconque :

```
def rotation(f_in,f_out,a):
    q=int((a%360)/90) # nb de 1/4 tours
    f=plt.imread(f_in)
    for i in range(q):
        # rotation 90%
        t=np.shape(f)
        g=np.zeros((t[1],t[0],t[2]))
        for j in range(t[1]):
            g[j]=f[:,-j-1]
        f=1*g
    plt.imsave(f_out,g)
    # rotation a degre (<90)
    a=a%90
    rotation90(f_out,f_out,a)
```

Solution 4 Ajustement des coefficients

```
def ajust(g):
    l,c,p=np.shape(g)
    for i in range(l):
        for j in range(c):
            for k in range(3):
                g[i,j,k]=min(max(g[i,j,k],0),1)
    return(g)
```


Lycée Saint-Exupery – maffre.saintexupery@gmail.com – MPSI

Solution 5 Filtre par convolution

```
def convulution(M,f_in,f_out):
 f=plt.imread(f_in)
 l,c,p=np.shape(f)
 r=len(M)//2
 g=np.zeros((l-2*r,c-2*r,3))
 s=np.sum(M)
 if s==0: s=1
 for i in range(r,l-r):
  for j in range(r,c-r):
   g[i-r,j-r]=np.array([np.sum(f[i-r:i+r+1,
        j-r:j+r+1,k]*M)/s for k in range(3)])
 g=ajust(g)
 plt.close()
 plt.imshow(g)
 plt.show()
 plt.imsave(f_out,g)
```

Solution 6 Détection des contours

Le laplacien est : $\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}$.

Une dérivée seconde se discrétise par :

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i-1,j}-2T_{i,j}+T_{i+1,j}}{h^2}$$

On peut donc considérer la matrice de convolution :

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

```
M=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
convulution(M,'Etretat.png','T.png')
negatif('T.png','Etretat_dessin.png')
```