TP3
Prétraitements

1. **Lecture de l'image de texte**

Par défaut, OpenCV ouvre une image en couleurs.

```
# Lire image de texte et la rendre en nivaux de gris
img=cv2.imread('data/Page3.png')
print(img.shape)
```

(1277, 3940, 3)

La passer en niveaux de gris.

```
# rendre l'image en niveaux de gris
img= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

(1277, 3940)

On peut lire directement l'image en niveaux de gris, de deux maniéres :

```
# Lire image de texte directement en nivaux de gris
img=cv2.imread('data/Page3.png', cv2.IMREAD_GRAYSCALE)
print(img.shape)
```

(1277, 3940)

```
# Lire image de texte directement en nivaux de gris
img=cv2.imread('data/Page3.png', 0)
print(img.shape)
```

(1277, 3940)

2. **Binarisation de l'image et l'inverser**

On peut procéder de plusieurs manières :
Remarque :
La fonction cv2.threshold donne deux sorties : le seuil de binarisation et l'image binarisée.

Ci-dessous, on ne récupère que l'image binarisée.

1. Seuil de binarisation global (écriture noirs sur fond blanc)

```
#  binarisation seuil global
bw=cv2.threshold(img,130,255,cv2.THRESH_BINARY)[1]
# inverser l'image
bw = cv2.bitwise_not(bw)
```

2. Seuil de binarisation global avec inversion (écriture blanche sur fond noir)

```
#  binarisation et inversion par un seul appel (seuil global)
bw=cv2.threshold(img,130,255,cv2.THRESH_BINARY_INV)[1]
```

3. Seuil de binarisation global avec inversion et méthode d'Otsu

```
#  binarisation et inversion par un seul appel (méthode OTSU)
bw=cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)[1]
```

4. On peut définir des fonctions pour les différents prétraitements

i) Niveaux de gris

```
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

ii) Binarisation

```
def binariser(image):
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
```

iii) Elimination du bruit

```
def remove_noise(image,p):
    return cv2.medianBlur(image, p)
```

iv) Dilatation

```
def dilate(image):
    kernel = np.ones((5, 5), np.uint8)
    return cv2.dilate(image, kernel, iterations=1)
```

v) Erosion

```
def erode(image):
    kernel = np.ones((5, 5), np.uint8)
    return cv2.erode(image, kernel, iterations=1)
```

On définit la **fermeture morphologique** comme une dilatation suivie d'une érosion. La fermeture a pour effets :

1. de faire disparaître les trous de petite taille dans les structures
2. de connecter les structures proches.

vi) Fermeture

```
def closing(image):
    kernel = np.ones((5, 5), np.uint8)
    return cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
```

On définit l'**ouverture morphologique** comme une érosion suivie d'une dilatation. L'ouverture a pour effets :

1. de faire disparaître les petites particules (dont la taille est inférieure à celle de l'élément structurant)
2. de séparer les grosses particules aux endroits où elles sont plus fines.

vii) Ouverture

```
def opening(image):
    kernel = np.ones((5, 5), np.uint8)
    return cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
```

viii) Application du filtre de Prewitt en utilisant les masques de ce dernier

Sachant que les masques sont les suivants :
Masque horizontal :

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Masque vertical :

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

```
def prewitt(image):
    kernelx = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
    kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
    prewittx  = cv2.filter2D(image, -1, kernelx)
    prewitty  = cv2.filter2D(image, -1, kernely)
    prewitt_filter = prewittx + prewitty
    return prewitt_filter
```

ix) Filtres de Canny, Sobel, Laplacien

```
def canny(bw, p1, p2):
    return cv2.Canny(bw, p1, p2)
```

```
def = Sobel(bw, p1, p2, p3)
    return cv2.Sobel(bw, p1, p2, p3)
```

```
def = Laplacien(bw, p)
    return cv2.Laplacian(bw,p)
```

5. Création des Trackbars

Les trackbars nous permettent de faire varier les paramètres de fonctions de façon interactive tout en visualisant en même temps les résultats.

Pour créer les trackbars, nous utilisons la fonction createTrackbar(). Cette méthode créera une barre de suivi, l'attachera à la fenêtre fournie et spécifiera l'objet image et la plage de la barre de suivi dans la fonction.
Dans notre cas, la plage sera de 0 à 255. Il prend également un paramètre OnChange contenant une fonction qui doit être appelée à chaque fois que la position du curseur change.

a) Trackbars pour la binarisation

Les paramètres que nous allons faire varier sont :

i) Le type de binarisation (de 0 à 4)

THRESH_BINARY (0) Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV (1) Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC (2) Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO (3) Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV (4) Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$

ii) Le seuil de binarisation

De 0 à 255

```python
import cv2
import numpy as np

img = cv2.imread('data/texte1.jpg',cv2.IMREAD_GRAYSCALE)
bw = np.zeros(img.shape,np.uint8)
th = 0
type = 0
def afficher():
    cv2.threshold(img,th,255,type,bw)
    cv2.imshow('result',bw)
def changeTh(x):
    global th
    th = x
    afficher()
```

```
def changeType(x):
    global type
    type = x
    afficher()
cv2.namedWindow('result')
cv2.createTrackbar('threshold','result',0,255,changeTh)
cv2.createTrackbar('type','result',0,4,changeType)
afficher()
cv2.waitKey()
cv2.destroyAllWindows()
```

b) Dilatation/Erosion

```
import cv2

sizeDilate = 1
sizeErode = 1
cv2.namedWindow("Erosion")
cv2.namedWindow("Dilatation")
img = cv2.imread("data/Page3.png",cv2.IMREAD_GRAYSCALE)
cv2.threshold(img,130,255,0,img)
img=cv2.bitwise_not(img)
img = cv2.resize(img, dsize=(0,0), fx=0.25, fy=0.25)

def dilate_func():
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(sizeDilate * 2 +
1,sizeDilate * 2 + 1))
    img_dilate = cv2.dilate(img,kernel,iterations=1)
    cv2.imshow("Dilatation",img_dilate)

def erode_func():
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS,(sizeErode * 2 + 1,sizeErode
* 2 + 1))
    img_erode = cv2.erode(img,kernel,iterations=1)
    cv2.imshow("Erosion",img_erode)

def changeErosionsize(x):
    global sizeErode
    sizeErode = x
    erode_func()

def changeDilationsize(x):
    global sizeDilate
    sizeDilate = x
    dilate_func()

cv2.createTrackbar("Size Erode","Erosion",1,21,changeErosionsize)
cv2.createTrackbar("Size Delate","Dilatation",1,21,changeDilationsize)
erode_func()
dilate_func()
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

c) Ouverture/Fermeture

```
import cv2
import numpy as np


cv2.namedWindow("Fermeture")
cv2.namedWindow("Ouverture")
img = cv2.imread("data/Page3.png",cv2.IMREAD_GRAYSCALE)
cv2.threshold(img,120,255,0,img)
img=cv2.bitwise_not(img)
# img = cv2.resize(img, dsize=(0,0), fx=0.25, fy=0.25)
sizeFermeture = 1
sizeOuverture = 1

def fermeture_fun():
    kernel = np.ones((sizeFermeture,sizeFermeture),np.uint8)
    img_ferme = cv2.morphologyEx(img,cv2.MORPH_CLOSE,kernel,iterations=1)
    cv2.imshow("Fermeture",img_ferme)
def ouverture_fun():
    kernel = np.ones((sizeOuverture,sizeOuverture),np.uint8)
    img_ouverte = cv2.morphologyEx(img,cv2.MORPH_OPEN, kernel,iterations=1)
    cv2.imshow("Ouverture",img_ouverte)
def changeFermeSize(x):
    global sizeFermeture
    sizeFermeture = x
    fermeture_fun()
def changeOuvertureSize(x):
    global sizeOuverture
    sizeOuverture = x
    ouverture_fun()
cv2.createTrackbar("FermeTrack","Fermeture",1,21,changeFermeSize)
cv2.createTrackbar("OuvertureTrack","Ouverture",1,21,changeOuvertureSize)
fermeture_fun()
fermeture_fun()

cv2.waitKey(0)
cv2.destroyAllWindows()
```