

Rapport de TPX – Lecture automatique de chiffres par analyse d'image

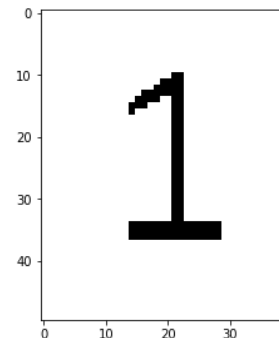
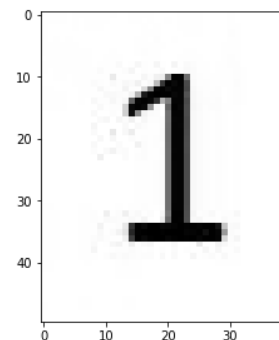
I. Présentation du TP

II. Prise en main de l'environnement

III. Travail préparatoire

- (1) H et W correspondent à la taille de l'image et à chaque attributs pixels correspond à une valeur qui est soit 0 pour le noir soit 255 pour le blanc.
- (2)

```
#####  
# Methode de binarisation  
# 2 parametres :  
# self : l'image a binariser  
# S : le seuil de binarisation  
# on retourne une nouvelle image binarisee  
#####  
def binarisation(self, S):  
    # creation d'une image vide  
    im_bin = Image()  
  
    # affectation a l'image im_bin d'un tableau de pixels de meme taille  
    # que self dont les intensites, de type uint8 (8bits non signes),  
    # sont mises a 0  
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))  
  
    # TODO: boucle imbriquees pour parcourir tous les pixels de l'image im_bin  
    # et calculer l'image binaire  
    for i in range(self.H):  
        for j in range(self.W):  
            if self.pixels[i][j] >= S:  
                im_bin.pixels[i][j] = 255  
            else:  
                im_bin.pixels[i][j] = 0  
  
    return im_bin  
pass
```



On réalise une methode binarisation(self, S) qui va nous servir à passer d'une image codée sur 256 valeurs, à une image codée avec deux valeurs (0 ou 255)

(3)

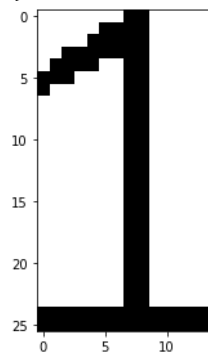
Code :

```
def localisation(self):
    L_max = 0
    L_min = self.H
    C_max = 0
    C_min = self.W
    for i in range(self.H):
        for j in range(self.W):
            if (self.pixels[i][j]==0):
                if i<L_min:
                    L_min = i
                if j<C_min:
                    C_min = j
                if i>L_max:
                    L_max = i
                if j>C_min:
                    C_max = j
    nbl=L_max-L_min
    nbc=C_max-C_min
    im_loc = Image()
    im_loc.set_pixels(np.zeros((nbl,nbc), dtype=np.uint8))
    for i in range(L_max-L_min):
        for j in range(C_max-C_min):
            im_loc.pixels[i][j]=self.pixels[L_min+i][C_min+j]
    return(im_loc)
pass
```

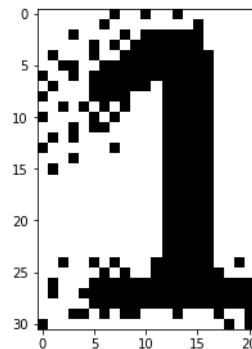
On réalise une méthode Localisation(self) qui permet de recadrer l'image au plus proche du chiffre

IV – Reconnaissance automatique de chiffre

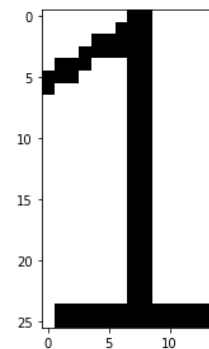
(1)



S = 70



S = 250



S = 20

On exécute le fichier main.py en faisant varier la valeur du seuil S et on analyse les différents résultats. On peut voir qu'il ne faut pas avoir un seuil trop bas pour ne pas perdre trop de pixels, ni un seuil trop haut pour obtenir une image correspondante à l'image originale.

```
In [31]: runfile('E:/INFO TP/src/test_Image.py', wdir='E:/INFO TP/src')
.....Reloaded modules: image
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test5.JPG (54x61)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test4.JPG (58x40)
lecture image : ../assets/test3.JPG (43x38)
```

Ran 7 tests in 0.074s

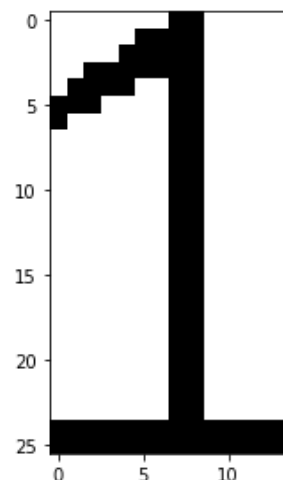
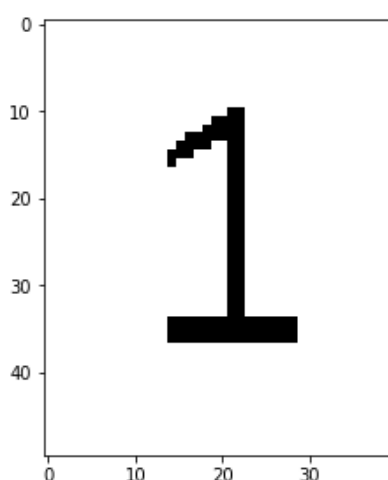
OK
An exception has occurred, use %tb to see the full traceback.

SystemExit: False

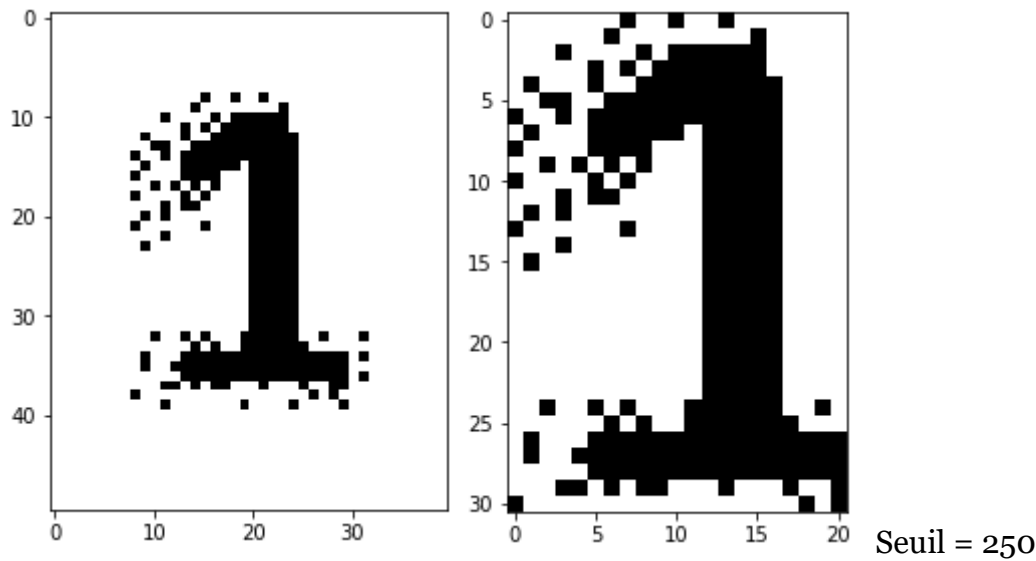
La méthode répond aux spécifications demandées, on obtient bien un « OK » à la suite du test.

(2)

Résultat après test :



Seuil = 70



On remarque que lorsque l'on lance la méthode localisation, l'image est bien recadrée. Mais lorsque le seuil est grand, on remarque que l'on perd des pixels lors du rognage.

Cette méthode sert à recadrer l'image.

(3)

```
#####
# Methode de redimensionnement d'image
#####
def resize(self, new_H, new_W):
    im_resized=resize(self.pixels, (new_H,new_W), 0)
    im_resized=np.uint8(im_resized*255)
    im_res = Image()
    im_res.set_pixels(np.zeros((new_H,new_W), dtype=np.uint8))
    for i in range(new_H):
        for j in range(new_W):
            im_res.pixels[i][j]=im_resized[i][j]
    return(im_res)
pass
```

```
In [49]: runfile('E:/INFO TP/src/test_Image.py', wdir='E:/INFO TP/src')
.....Reloaded modules: image
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test5.JPG (54x61)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test4.JPG (58x40)
lecture image : ../assets/test3.JPG (43x38)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test4.JPG (58x40)
lecture image : ../assets/test3.JPG (43x38)
```

Ran 13 tests in 0.189s

OK

An exception has occurred, use %tb to see the full traceback.

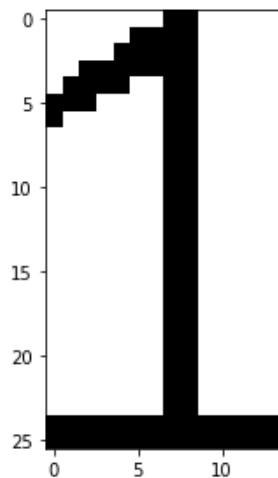
SystemExit: False

Résultat du test Image.py

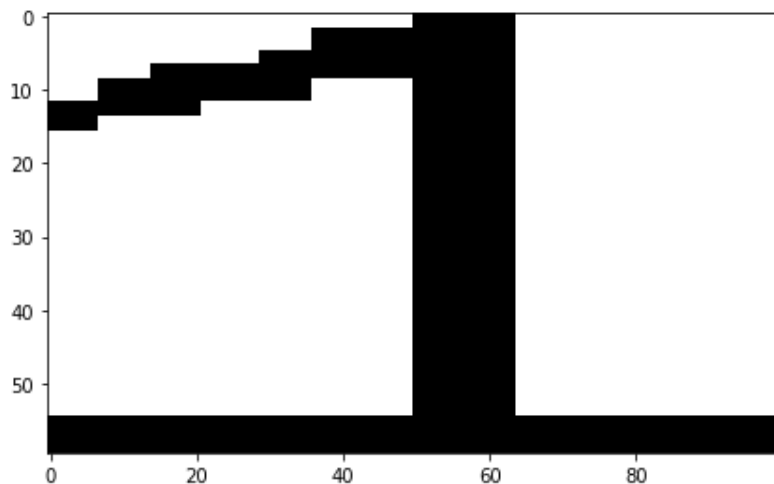
On crée et on teste la fonction resize et en réalisant le test avec les valeurs de (60, 100) pour (new_H, new_W)

Cette fonction sert à redimensionner l'image avec des dimensions données.

On obtient donc l'image redimensionnée suivante :



(25, 15)



(60, 100)

(4)

```
#####
# Methode de mesure de similitude entre l'image self et un modele im
#####
def similitude(self, im):
    sim=0
    for i in range(self.H):
        for j in range(self.W):
            if self.pixels[i][j]==im.pixels[i][j]:
                sim+=1
    return(sim/(im.H*im.W))
    pass
```

Résultat :

```
In [62]: runfile('E:/INFO TP/src/test_Image.py', wdir='E:/INFO TP/src')
....lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test3.JPG (43x38)
lecture image : ../assets/test2.JPG (50x40)
```

 Ran 4 tests in 0.029s

OK

An exception has occurred, use %tb to see the full traceback.

SystemExit: False

On réalise la méthode `similitude(self, im)`, qui renvoie la ressemblance entre deux images. Pour cela, la fonction analyse l'image, et prend en compte les pixels similaires. Elle retourne la similitude totale divisée par la aire totale de l'image.

(5)

```

4 def reconnaissance_chiffre(image, liste_modeles, S):
5     image=image.binarisation(S)
6     image=image.localisation()
7     sim=0
8     chiffre=0
9     for i in range(len(liste_modeles)):
10         image=image.resize(liste_modeles[i].H,liste_modeles[i].W)
11         if image.similitude(liste_modeles[i])>sim:
12             sim=image.similitude(liste_modeles[i])
13             chiffre=i
14     return(chiffre)
15     pass

```

Résultat :

```

In [75]: runfile('E:/INFO TP/src/
test_reconnaissance.py', wdir='E:/INFO TP/src')
.....Reloading modules: image
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test5.JPG (54x61)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test4.JPG (58x40)
lecture image : ../assets/test3.JPG (43x38)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test4.JPG (58x40)
lecture image : ../assets/test3.JPG (43x38)
.....lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/test3.JPG (43x38)
lecture image : ../assets/test2.JPG (50x40)
lecture image : ../assets/_0.png (32x22)
lecture image : ../assets/_1.png (32x18)
lecture image : ../assets/_2.png (32x20)
lecture image : ../assets/_3.png (32x20)
lecture image : ../assets/_4.png (32x24)
lecture image : ../assets/_5.png (32x20)
lecture image : ../assets/_6.png (32x21)
lecture image : ../assets/_7.png (32x21)
lecture image : ../assets/_8.png (32x22)
lecture image : ../assets/_9.png (32x22)
lecture image : ../assets/test1.JPG (54x43)
lecture image : ../assets/_0.png (32x22)
lecture image : ../assets/_1.png (32x18)
lecture image : ../assets/_2.png (32x20)
lecture image : ../assets/_3.png (32x20)
lecture image : ../assets/_4.png (32x24)
lecture image : ../assets/_5.png (32x20)
lecture image : ../assets/_6.png (32x21)
lecture image : ../assets/_7.png (32x21)
lecture image : ../assets/_8.png (32x22)
lecture image : ../assets/_9.png (32x22)
lecture image : ../assets/test3.JPG (43x38)

-----
Ran 25 tests in 0.464s

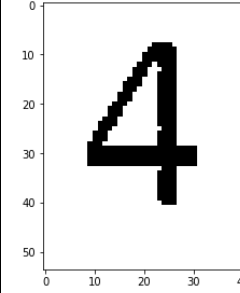
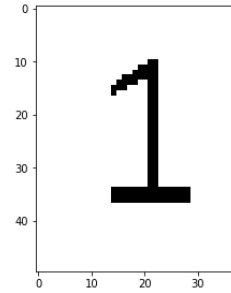
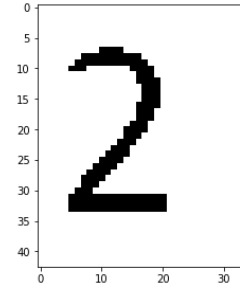
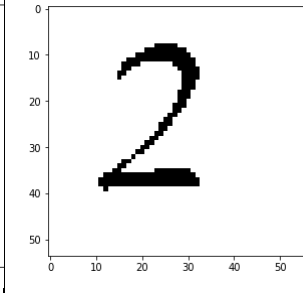
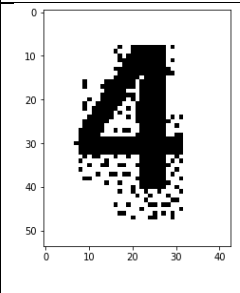
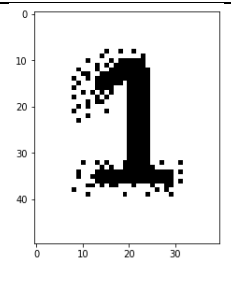
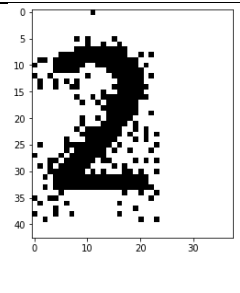
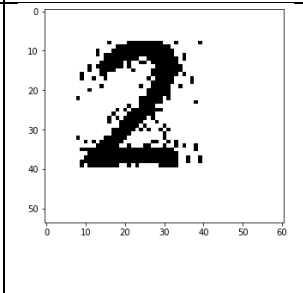
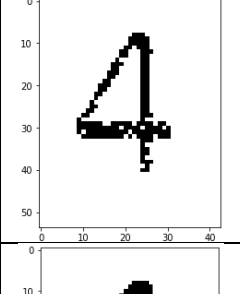
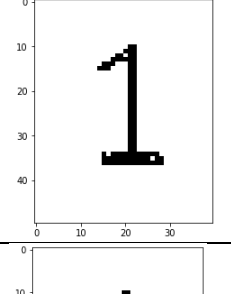
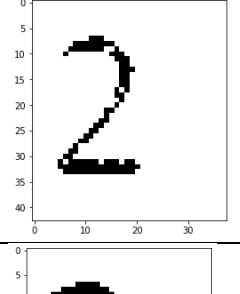
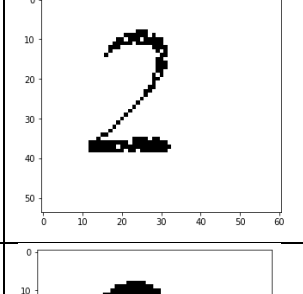
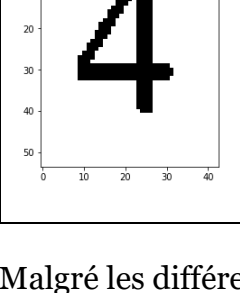
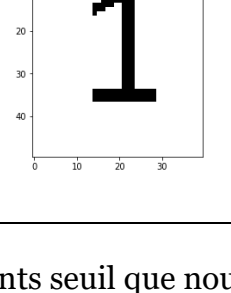
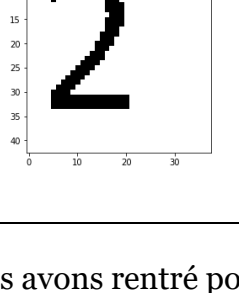
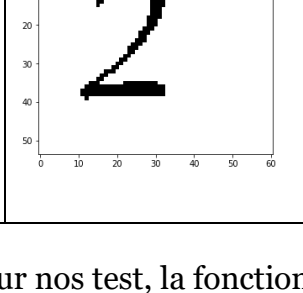
OK
An exception has occurred, use %tb to see the full
traceback.

SystemExit: False

```

Cette fonction exploite toutes les fonctions précédentes créées.
 L'objectif de la fonction est de trouver à quel chiffre correspond l'image prise en paramètre. La fonction appelle les méthodes binarisation et localisation.
 Le résultat est comparé grâce à la méthode similitude, aux modèles existants.
 On ajoute et on mémorise le chiffre qui à le plus de similitude et on le renvoie.

(6)

Test 1	Test 2	Test 3	Test 5	S
				70
				250
				10
				100

Malgré les différents seuil que nous avons rentré pour nos test, la fonction reconnaissance arrive à associer chaque image au bon chiffre.