

Rapport de TP2 – Lecture automatique de chiffres par analyse d'image

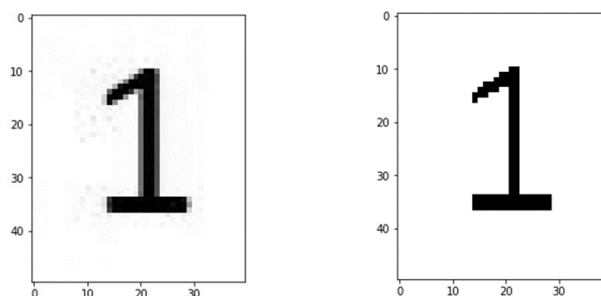
I. Introduction

L'objectif de ce TP est d'illustrer, parmi les différentes techniques de lecture automatique de chiffres, une des solutions les plus simples : la reconnaissance par corrélation avec des modèles.

II. Travail préparatoire

1) H correspond à la hauteur de l'image et W correspond à la largeur de l'image. Pixel est un tableau 2D.

2) La méthode binarisation(self, S) prend en attribut une valeur fixée par l'utilisateur. Tous les pixels ayant une valeur supérieure à S prennent la valeur 255 et deviennent blanc. Tous les pixels ayant une valeur inférieure à S prennent la valeur 0 et deviennent noir. On obtient ainsi une image uniquement en noir et blanc.

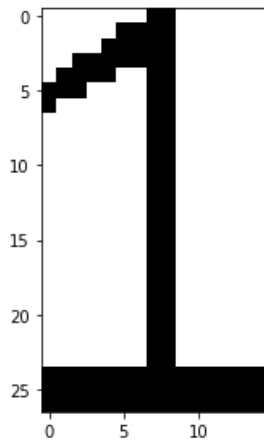


A gauche, il s'agit de l'image avant l'exécution de la méthode binarisation, à droite il s'agit de l'image obtenue après l'exécution de la méthode.

Voici, notre code :

```
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 imbriquee 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
```

3) Nous avons écrit la méthode localisation, qui permet de retourner l'image recadrée. Cela permet ainsi de supprimer les parties blanches qui n'ont pas une grande utilité. Voici l'image que nous avons obtenue :



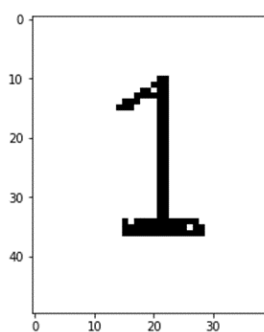
Voici notre code :

```
def localisation(self):
    c_min= self.W
    c_max=0
    l_min=self.H
    l_max=0
    for l in range(self.H):
        for c in range(self.W):
            if self.pixels[l][c]==0:
                if c<=c_min:
                    c_min=c
                if c>=c_max:
                    c_max=c
                if l<=l_min:
                    l_min=l
                if l>=l_max:
                    l_max=l

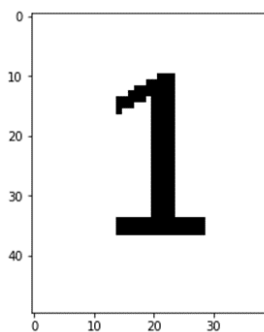
    res= Image()
    res.set_pixels(self.pixels[l_min:l_max+1,c_min:c_max+1])
    return res
```

III. Reconnaissance automatique de chiffre

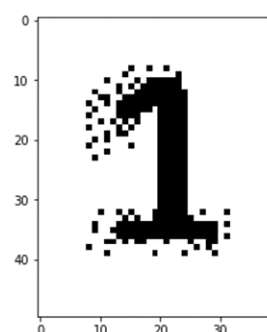
1) Nous avons testé la méthode binarisation pour plusieurs valeurs de S, voici les images que nous avons obtenues :



S=10



S=130



S=250

Nous remarquons que les valeurs les plus adaptées pour S sont les valeurs situées autour de 100. De plus, pour les valeurs plus faibles (environ 10), il n'y a pas assez de pixels qui deviennent noirs. Au contraire, pour les valeurs les plus grandes (aux alentours de 250), il y a trop de pixels qui deviennent noirs.

Nous avons également lancé le fichier `test_image`, nous obtenons ce résultat :

```
-----  
Ran 25 tests in 0.156s
```

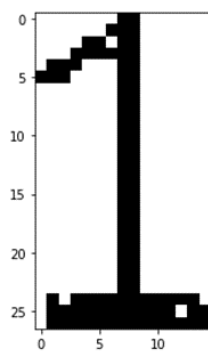
```
FAILED (failures=8, errors=3)
```

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

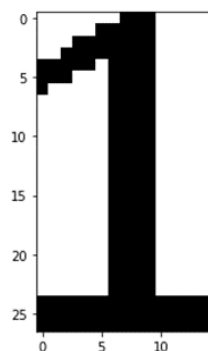
```
SystemExit: True
```

Nous avons 11 tests qui ne fonctionnent pas car nous n'avons pas encore écrit les méthodes correspondantes. Donc nos 14 tests pour nos 2 fonctions fonctionnent bien.

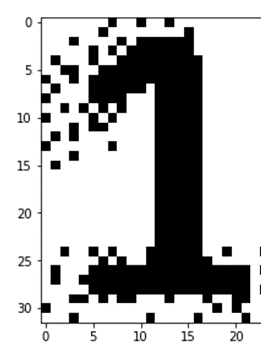
2) Nous avons testé la méthode localisation pour plusieurs valeurs de seuil :



$S=10$



$S=130$



$S=250$

Comme pour la méthode binarisation, nous remarquons que les valeurs les plus adaptées pour S sont les valeurs situées autour de 100. De plus, pour les valeurs plus faibles (environ 10), il n'y a pas assez de pixels qui deviennent noirs. Au contraire, pour les valeurs les plus grandes (aux alentours de 250), il y a trop de pixels qui deviennent noirs.

Nous avons également lancé le fichier `test_image`, nous obtenons ce résultat :

```
-----  
Ran 25 tests in 0.156s
```

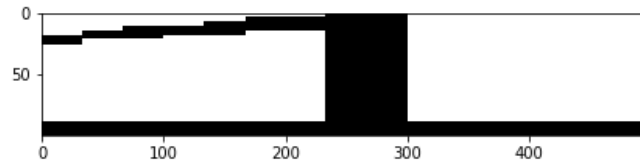
```
FAILED (failures=8, errors=3)
```

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

```
SystemExit: True
```

Nous avons 11 tests qui ne fonctionnent pas car nous n'avons pas encore écrit les méthodes correspondantes. Donc nos 14 tests pour nos 2 fonctions fonctionnent bien.

3) Nous avons écrit la méthode `resize` qui permet d'imposer des dimensions à l'image. En imposant $H=100$ et $W=500$, nous obtenons l'image suivante :



Voici notre code :

```
def resize(self, new_H, new_W):  
    im_resize=Image()  
    im_resize.H=new_H  
    im_resize.W=new_W  
    im_resize.pixels=resize(self.pixels,(new_H,new_W),0)  
    im_resize.pixels=np.uint8(im_resize.pixels*255)  
    return im_resize
```

Nous avons lancé le fichier `test_image`. Nous n'avons plus que 7 erreurs qui correspondent aux fonctions que nous n'avons pas encore écrites. Ainsi, nous voyons bien que les 4 tests qui correspondent à la méthode `resize` fonctionnent bien.

Ran 25 tests in 0.202s

FAILED (failures=6, errors=1)

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

SystemExit: True

4) Nous avons écrit la méthode `similitude` qui permet de comparer 2 images. Le résultat est un chiffre compris entre 0 et 1. Plus le chiffre est proche de 1, plus les images se ressemblent. Pour que la méthode fonctionne, il faut imposer la même taille pour chaque image grâce à la fonction `resize`. Nous avons choisi d'imposer $H=150$ et $W=150$. Voici notre code :

```
def similitude(self, im):  
    pixels=0  
    res=0  
    simili=0  
    self.resize(150,150)  
    im.resize(150,150)  
    for i in range(self.H):  
        for j in range(self.W):  
            if self.pixels[i][j]==im.pixels[i][j]:  
                simili+=1  
            pixels =pixels+1  
    res=simili/pixels  
    return res
```

Nous avons lancé le fichier `test_image`. IL ne nous reste plus que 3 erreurs donc les tests qui correspondent à la fonction `similitude` fonctionnent correctement.

Ran 25 tests in 0.187s

FAILED (failures=2, errors=1)

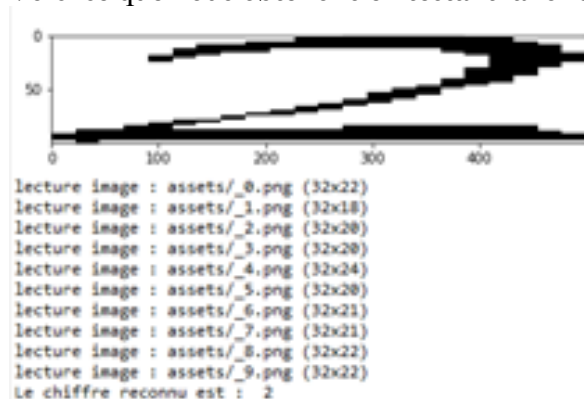
An exception has occurred, use `%tb` to see the full traceback.

SystemExit: True

5) Nous avons écrit la fonction reconnaissance_chiffre dans le fichier reconnaissance. Cette fonction permet de comparer deux images de chiffres et de dire s'il s'agit du même chiffre sur les deux images. Voici notre code :

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    res=0  
    similimax=0  
    im=image.localisation()  
    for i in range (len(liste_modeles)):  
        H=liste_modeles[i].H  
        W=liste_modeles[i].W  
        im=im.resize(H,W)  
        simili=im.similitude(liste_modeles[i])  
        if simili> similimax:  
            similimax=simili  
            res=i  
    return res
```

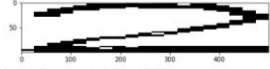
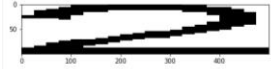
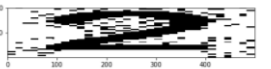
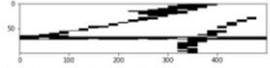
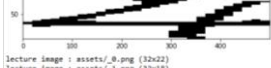
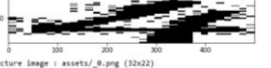
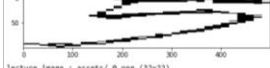
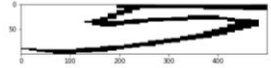
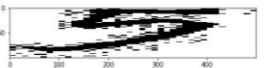
Voici ce que nous obtenons en testant la fonction :



Nous avons également lancé le fichier test_reconnaissance.py, nous n'avons plus d'erreur donc les tests associés à notre méthode fonctionnent correctement.

```
Ran 25 tests in 0.297s  
  
OK  
An exception has occurred, use %tb to see the full traceback.  
  
SystemExit: False
```

6)

Seuil	10	130	250
Image test-4.JPG	 <pre>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) Le chiffre reconnu est : 2</pre>	 <pre>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) Le chiffre reconnu est : 2</pre>	 <pre>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) Le chiffre reconnu est : 2</pre>
Image test-6.JPG	 <pre>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) Le chiffre reconnu est : 4</pre>	 <pre>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) Le chiffre reconnu est : 4</pre>	 <pre>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) Le chiffre reconnu est : 4</pre>
Image test-7.JPG	 <pre>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) Le chiffre reconnu est : 7</pre>	 <pre>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) Le chiffre reconnu est : 7</pre>	 <pre>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) Le chiffre reconnu est : 7</pre>

On remarque que pour les tests 4 et 6, peu importe le seuil, le chiffre est bien reconnu. Cependant pour le test 7, l'ordinateur reconnaît un 7 au lieu d'un 5, cela est sûrement dû à la netteté de l'image qui n'est pas très bonne.

IV. Conclusion

Lors de ce TP, nous avons appris à utiliser le traitement d'image. Nous avons pu comparer et modifier des images en créant différentes méthodes.



POLYTECH[®]
ANNECY-CHAMBERY



UNIVERSITÉ
SAVOIE
MONT BLANC