

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

I. Introduction

Le but de ce TP est d'utiliser la reconnaissance par corrélation des modèles afin d'effectuer une lecture automatique des chiffres. Ce principe se décompose en différentes étapes, à savoir la binarisation, la localisation, puis l'adaptation de la taille au modèle et enfin la mesure de ressemblance par corrélation pour permettre la prise de décision.

II. Prise en main de l'environnement :

```
1  '''
2  File: main.py
3  Created Date: Friday August 27th 2021 - 02:35pm
4  Author: Ammar Mian
5  Contact: ammar.mian@univ-smb.fr
6  -----
7  Last Modified: Mon Aug 30 2021
8  Modified By: Ammar Mian
9  -----
10 Copyright (c) 2021 Université Savoie Mont-Blanc
11 '''
12 import matplotlib.pyplot as plt
13 from image import Image
14 from reconnaissance import reconnaissance_chiffre, lecture_modeles
15
16
17 if __name__ == '__main__':
18
19     # Variables utiles
20     path_to_assets = '../assets/'
21     plt.ion() # Mode interactif de matplotlib pour ne pas bloquer l'exécution lorsque l'on fait display
22
23     #=====
24     # Lecture image et affichage
25     #=====
26     image = Image()
27     image.load(path_to_assets + 'test2.JPG')
28     image.display("Exemple d'image")
29
30     #=====
31     # Binarisation de l'image et affichage
32     #=====
33     S = 70
34     image_binarisee = image.binarisation(S)
35     image_binarisee.display("Image binarisee")
36
37     #=====
38     # Localisation de l'image et affichage
39     #=====
40     image_localisee = image_binarisee.localisation()
41     image_localisee.display("Image localisee")
42
43     #=====
44     # Redimensionnement de l'image et affichage
45     #=====
46     image_resizee = image_localisee.resize(100, 500)
47     image_resizee.display("Image redimensionnee")
48
49     #=====
50     # Lecture modeles et reconnaissance
51     #=====
52     liste_modeles = lecture_modeles(path_to_assets)
53     chiffre = reconnaissance_chiffre(image, liste_modeles, 70)
54     print("Le chiffre reconnu est : ", chiffre)
```

Figure 1 : fichier main.py

Afin de démarrer le TP par la première étape du principe de reconnaissance, il a été nécessaire d'isoler (mettre en commentaire) les étapes suivantes. En effet, au fur et à mesure des étapes il sera nécessaire de retirer ces commentaires afin d'effectuer les méthodes. Une fois à l'étape de reconnaissance (dernière étape à réaliser), toutes les requêtes du fichier *main* seront donc actives.

III. Travail préparatoire :

- 1.
2. Question 2 :

```
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 j in range(self.W):  
        for i in range(self.H):  
            if self.pixels[i,j] >= S :  
                im_bin.pixels[i,j] = 255  
            else:  
                im_bin.pixels[i,j] = 0  
  
    return im_bin
```

Figure 2 : méthode de binarisation

Remarque :

Avec cette méthode on compare les valeurs des pixels du tableau avec une valeur de seuil définie par S. Et donc si la valeur du pixel est supérieure à S le pixel devient noir et si la valeur est inférieure, le pixel est blanc. Cela nous permet donc d'obtenir une image en noir et blanc.

Images obtenues :

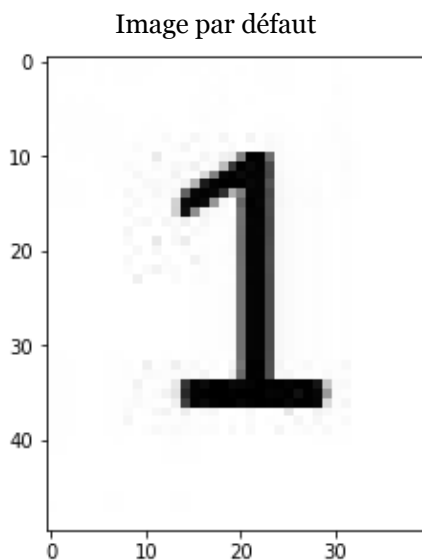


Figure 3 : "Test2"

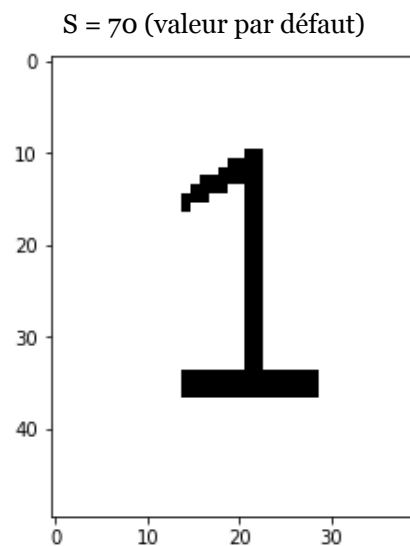


Figure 4 : "Test2" - S = 70

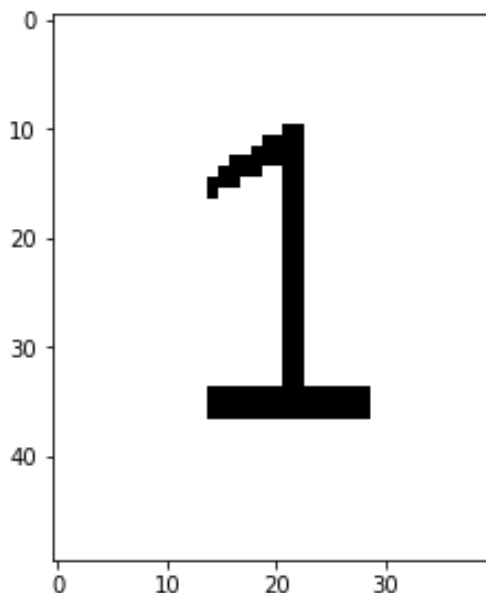
Il est donc ici visible que la réalisation de cette méthode permet d'obtenir une image bicolore plus nette. Il a ici été appliqué un seuil de 70 étant que cette valeur est celle par défaut. L'influence de la valeur du seuil sur l'image sera expliquée dans la partie suivante.

3. Question 3 :

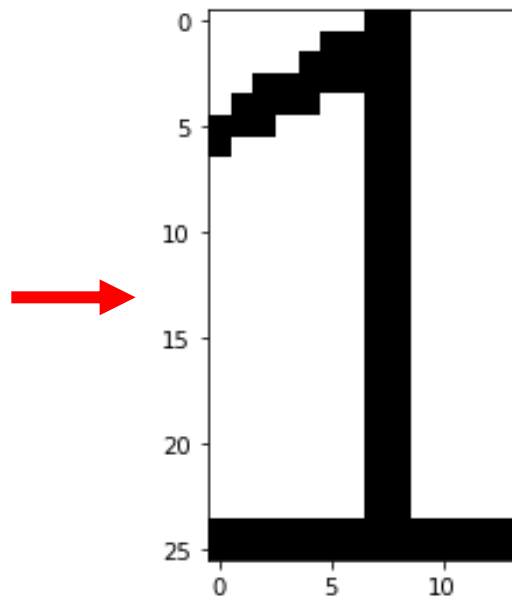
```
def localisation(self):  
  
    im_loc = Image()  
    l_max = 0  
    l_min = self.H-1  
    c_min = self.W-1  
    c_max = 0  
  
    for l in range(self.H):  
        for c in range(self.W):  
  
            if l < l_min:  
                if self.pixels [l][c] == 0:  
                    l_min = l  
            elif l > l_max:  
                if self.pixels [l][c] == 0:  
                    l_max = l  
  
            if c < c_min:  
                if self.pixels [l][c] == 0:  
                    c_min = c  
            elif c > c_max:  
                if self.pixels [l][c] == 0:  
                    c_max = c  
  
    im_loc.set_pixels(self.pixels[l_min:l_max,c_min:c_max])  
    return im_loc
```

Figure 5 : méthode de localisation

Avant :



Après :



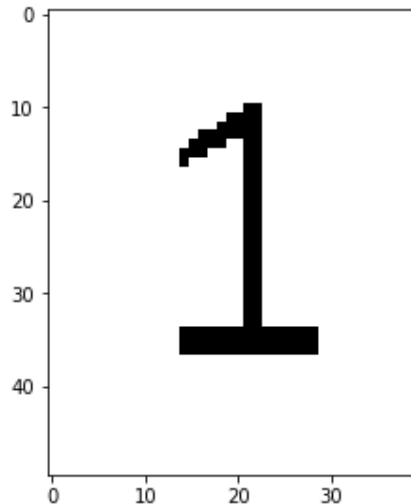
Cette méthode permet donc de recadrer l'image en prenant comme contour les valeurs minimums et maximums en abscisse et en ordonnée (ligne/colonne).

IV. Reconnaissance automatique de chiffre :

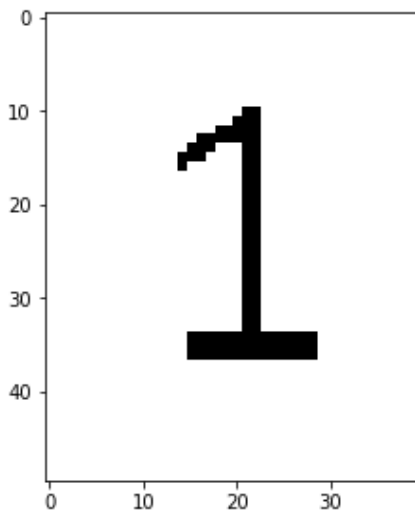
1. Question 1 :

Lors de la partie précédente, l'application de la méthode de binarisation a déjà été effectuée. Il s'agit ici de faire varier le seuil (S) et d'en étudier son influence.

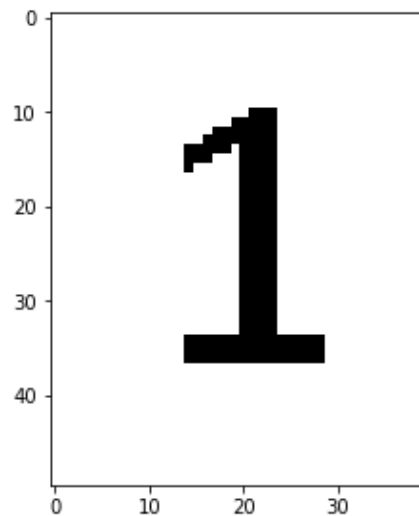
Pour S = 70 :



Pour S = 30 :



Pour S = 130 :



Remarque :

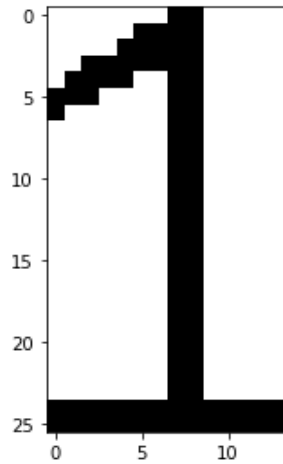
En faisant varier la valeur du seuil, l'épaisseur du chiffre et donc la différenciation entre un pixel considéré comme blanc et un autre comme noir, va être impactée. Plus la valeur du seuil est importante, plus le chiffre est « épais » et inversement, plus S diminue, plus le chiffre est fin.

Pour ce qui est de lancer le fichier de test « test_Image.py », aucune erreur n'est apparue dans la console, l'ensemble des tâches ont pu être effectuée.

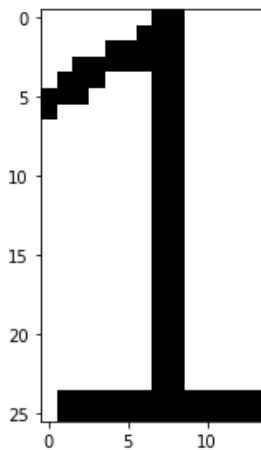
2. Question 2 :

Lors de la partie précédente, l'application de la méthode de localisation a déjà été effectuée. IL s'agit ici de faire varier le seuil (S) et d'en étudier son influence.

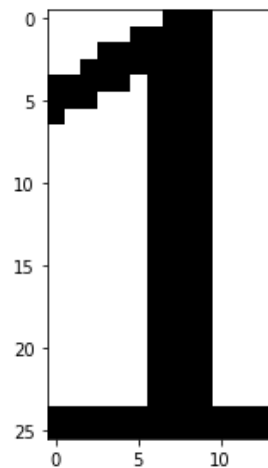
Pour S = 70 :



Pour S = 30 :



Pour S = 130 :



Remarque :

Il est ici visible que la méthode « binarisation » est toujours active, car plus S augmente, plus le chiffre s'appaisse et inversement. Par contre on remarque que cette fois-ci, la fonction « localisation » permet en plus de recadrer l'image autour du symbole, en l'occurrence ici, le chiffre 1 (conclusion de la partie précédente). Ce qui permet ici de voir davantage l'impact du seuil sur l'image.

Pour ce qui est de lancer le fichier de test « test_Image.py », aucune erreur n'est apparue dans la console, l'ensemble des tâches ont pu être effectuée.

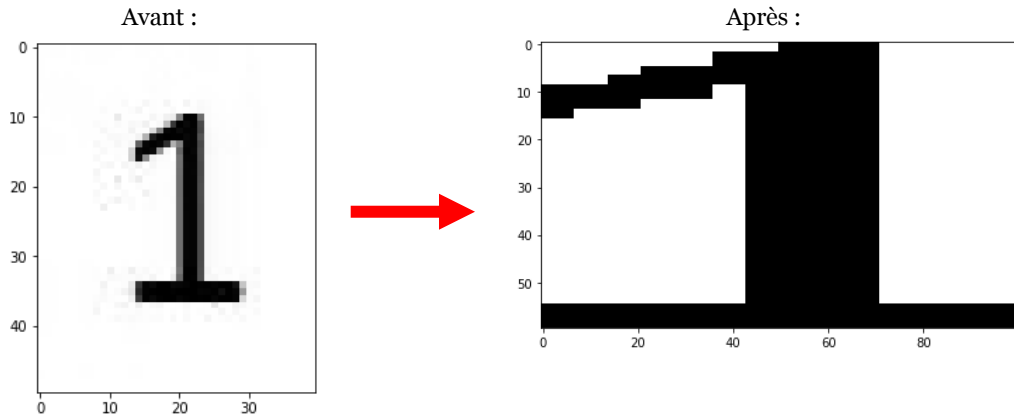
3. Question 3 :

Il est ici question de redimensionner l'image en fonction de dimensions choisies (ligne,colonne) :

```
def resize(self, new_H, new_W):  
    im_res = Image()  
    tab0= resize(self.pixels, (new_H,new_W),0)  
    tab_res = np.uint8(tab0*255)  
    im_res.set_pixels(tab_res)  
  
    return im_res
```

Figure 6 : méthode de resize

Le résultat suivant est obtenu :



Remarque :

L'image « Avant » représente l'image par défaut et l'image « Après » est l'image après les méthodes de binarisation, localisation et resize. En plus d'être recadrer, celle-ci possède les dimensions choisies (60,100). L'image est donc logiquement déformée par rapport à l'originale.

Pour ce qui est de lancer le fichier de test « test_Image.py », aucune erreur n'est apparue dans la console, l'ensemble des tâches ont pu être effectuée.

4. Question 4 :

```
def similitude(self, im):  
    n=0  
    for l in range(self.H):  
        for c in range(self.W):  
            if self.pixels[l][c] == im.pixels[l][c]:  
                n+=1  
    return n/(self.H*self.W)
```

Figure 7 : méthode de similitude

Remarque :

Cette méthode permet de mesurer la similitude entre deux images (par corrélation).

5. Question 5 :

```
def reconnaissance_chiffre(image, liste_modeles, S):
    image = image.binarisation(S)
    image = image.localisation()
    similitude = 0

    for x in range(len(liste_modeles)):
        im = image.resize(liste_modeles[x].H, liste_modeles[x].W)
        if im.similitude(liste_modeles[x]) > similitude:
            similitude = im.similitude(liste_modeles[x])
            indiceimage = x

    return indiceimage
```

Figure 8 : méthode reconnaissance

```
In [76]: runfile('D:/Théo/POLYTECH/INFO501/tp2-reconnaissance-chiffres-tp2_pik_camus-main/src/main.py',
wdir='D:/Théo/POLYTECH/INFO501/tp2-reconnaissance-chiffres-tp2_pik_camus-main/src')
Reloaded modules: image
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)
Le chiffre reconnu est : 1
```

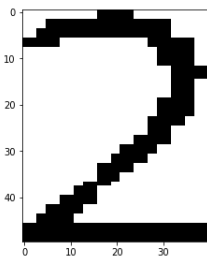
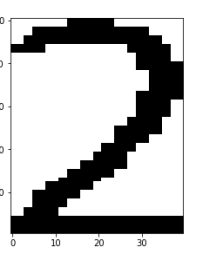
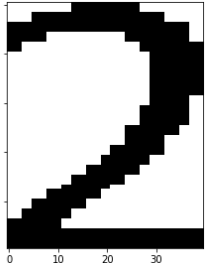
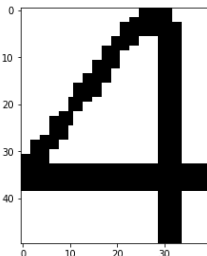
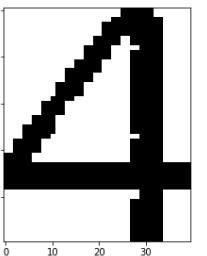
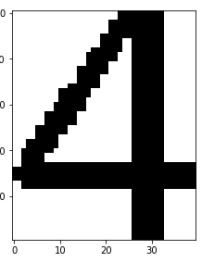
Figure 9 : résultat console

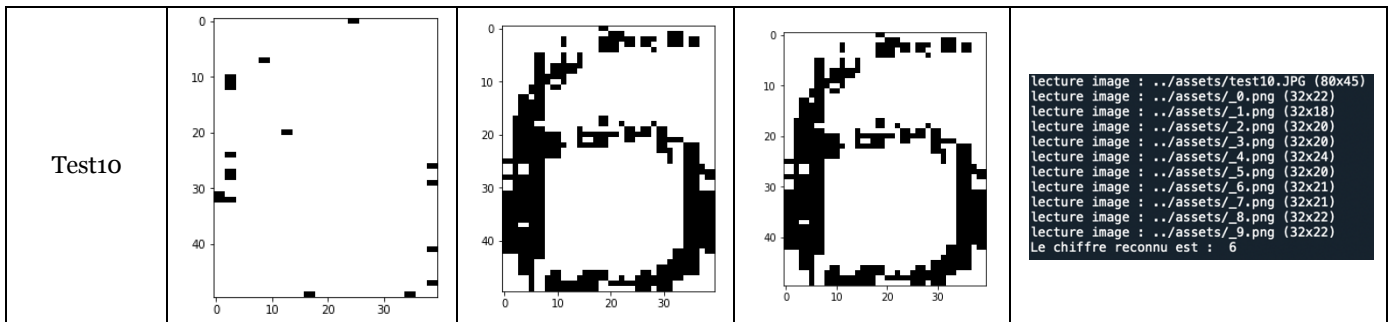
Remarque :

Il est ici affiché que le chiffre reconnu est « 1 », c'est donc ici correct car l'image forme bien le chiffre 1.

Pour ce qui est de lancer le fichier de test « test_renaissance.py », aucune erreur n'est apparue dans la console, l'ensemble des tâches ont pu être effectuée.

6. Question 6 :

	Seuil			
Fichier	30	70	130	Résultat
Test2				lecture image : ../assets/test3.JPG (43x38) 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
Test1				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) Le chiffre reconnu est : 4



Il semblerait qu'après les tests effectués sur trois autres chiffres, d'une part le code de la méthode de ressemblance fonctionne dans chacun des cas, et de plus le seuil le plus élevé (130) permet d'avoir la meilleure forme/visibilité. Il serait donc judicieux de choisir un seuil aux alentours de 130. Il faudra être vigilant qu'un seuil trop élevé (trop supérieur à 130) rendra illisible l'image.