

Rapport de TP2 – Reconnaissance de chiffres.

I. Introduction

L'objectif de ce TP est de réaliser une lecture par nuance de gris afin de reconnaître automatiquement un caractère. Pour cela nous allons réaliser une classe **Image** avec plusieurs méthodes et d'écrire une fonction **reconnaissance_chiffre** qui permettra de reprendre les fonctions du TP afin d'atteindre notre but.

II. Travail Préparatoire.

Question 2

On souhaite réaliser une méthode **binarisation** à la classe **Image** afin de retourner une image constituée seulement de cases blanches ou noires. On a :

```
def binarisation(self, S):
    im_bin=Image()
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))
    for i in range(self.H):
        for j in range(self.W):
            if self.pixels[i][j]<S:
                im_bin.pixels[i][j]=0
            else:
                im_bin.pixels[i][j]=255
    return im_bin
```

On vérifie chaque case, et, si la valeur au sein de la case est inférieure à notre seuil alors la case est noire (0) sinon elle est blanche (255).

Question 3

On souhaite réaliser une méthode **localisation** permettant de recadrer l'image, pour cela nous avons dû déterminer **l_min**, **l_max**, **c_min**, **c_max**, qui sont les lignes minimum/maximum et les colonnes minimum/maximum.

```
def localisation(self):
    c_min=self.W-1
    l_min=self.H-1
    c_max=0
    l_max=0
    image_recadree=Image()
    for i in range(self.H):
        for j in range(self.W):
            if self.pixels[i][j]==0:
                if(l_min>i):
                    l_min=i
                if(l_max<i):
                    l_max=i
                if(c_min>j):
                    c_min=j
                if(c_max<j):
                    c_max=j

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

On initialise **c_min** et **l_min** comme étant la valeur maximale de colonnes et de lignes, et **c_max** et **l_max** à 0.

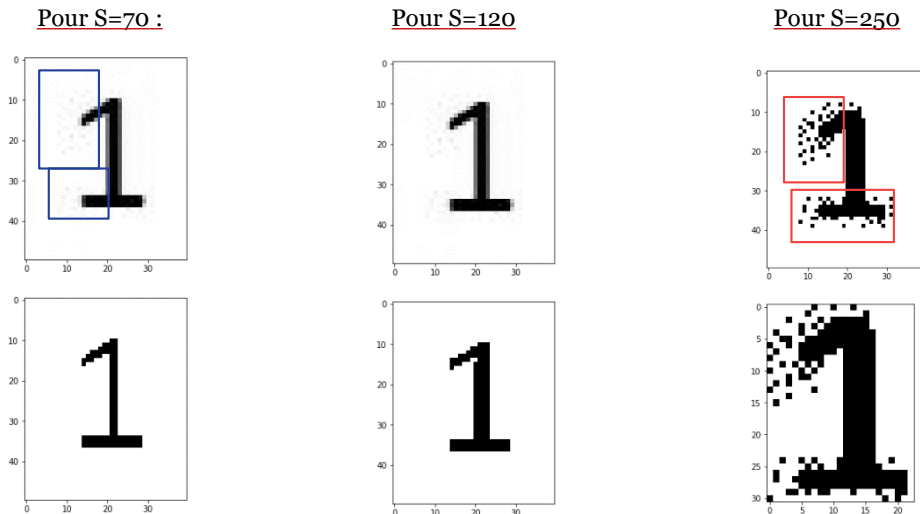
Ensuite, on fait une double boucle correspondant aux nombres de lignes et de colonnes et on regarde si le pixel considéré est noir (0), si oui alors on effectue des conditions pour déterminer les valeurs finales de **c_min**, **l_min**, **c_max** et **l_max**. On retourne l'image recadrée de l'image de départ. On peut désormais lancer le fichier **main** avec différentes valeurs de seuil.



III – Reconnaissance automatique de chiffre.

Question 1

On lance le fichier **main** avec différentes valeurs de seuil, et on a :

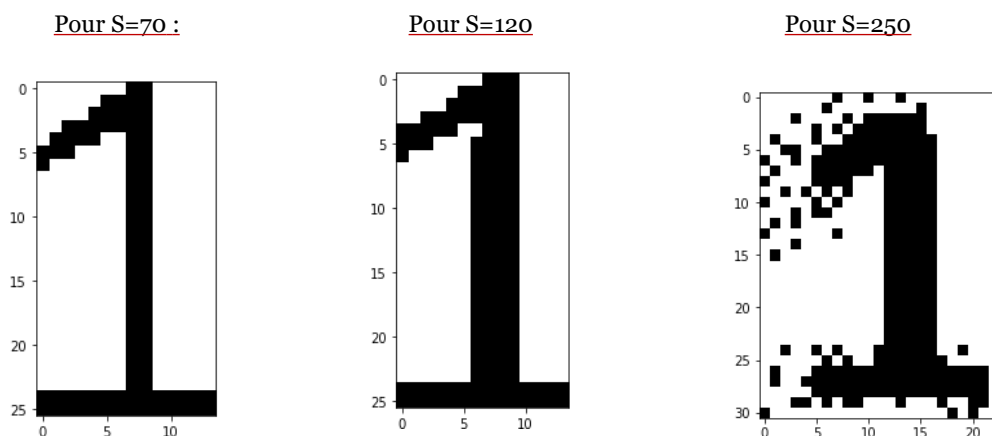


On remarque que plus on augmente la valeur de S plus on est sensible en terme de lecture du pixel. Ainsi pour une petite variation de couleur sur l'image de départ (zones encadrées en bleu), le pixel va être retourné en noir (zones encadrées en rouge). Il faut donc trouver un juste milieu permettant de retourner une image qui est la plus ressemblante à celle de départ.

En lançant le fichier **test_image** on observe qu'il n'y pas d'erreur pour les fonctions **binarisation** et **localisation**.

Question 2

Pour la localisation, on a :



On remarque que plus on augmente S , moins l'image est centrée sur le chiffre en question. En effet, comme nous l'avons vu à la question précédente, l'infime variation de couleur de l'image de départ nous renvoie un chiffre avec des pixels noirs qui ne correspond pas réellement à ce que l'on souhaite. On a donc un recadrage plus grand que les deux précédents ($S=70$ et $S=250$).

Question 3

Cette question nous demande de créer une méthode **resize** qui permet de redimensionner l'image à la taille voulue.

```
def resize(self, new_H, new_W):  
    image_resize2=Image()  
    image_resize2.set_pixels(np.uint8(resize(self.pixels, (new_H,new_W),0)*255))  
    return (image_resize2)
```

Nous avons donc créer **image_resize** qui sera notre image à la bonne dimension. Ensuite, nous avons utilisé la méthode **set_pixels** et convertit l'image **float** en **int**.

Question 4

Cette méthode permet de déterminer le pourcentage de similitude entre deux images en comparant le nombre de case noir qu'elles possèdent respectivement.

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

Cette méthode a été selon nous la plus difficile à mener, même si à première vue elle semble facile à comprendre, sa réalisation nous a posé quelques difficultés. On a commencé par créer une **image_resize** avec les dimensions de l'image **im**. On initialise un **compteur** à 0 qui sera le nombre de case noir égale, à la même position entre les deux images que l'on compare. On effectue donc une double boucle (nombre de lignes=**im.H**, nombre de colonnes=**im.W**). On vient vérifier si la case en question est noire, si oui, alors on incrémente **compteur** de 1. On retourne ainsi la valeur correspondant au ratio entre le nombre de cases noires égales et le nombre totale de case de l'image **im**.

Question 5

Dans cette question on nous demande de créer la fonction **reconnaissance_chiffre** qui va permettre de faire la reconnaissance de notre chiffre de départ par rapport à un modèle, en utilisant ce que nous avons fait jusque-là. Ainsi, cette fonction est primordiale pour savoir si notre TP a été mené correctement. Voici comment se compose la fonction :

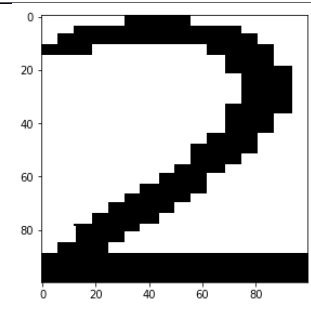
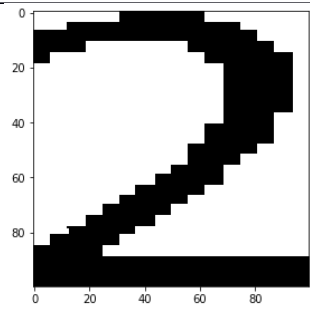
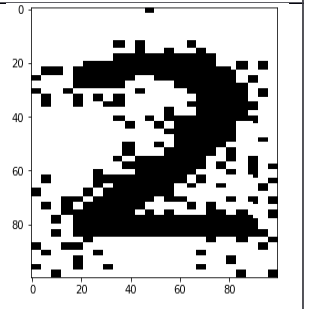
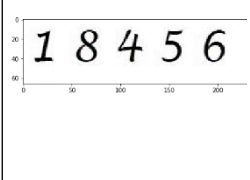
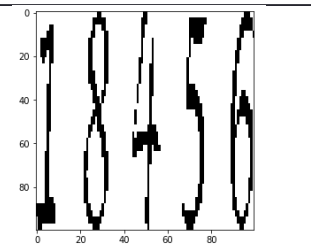
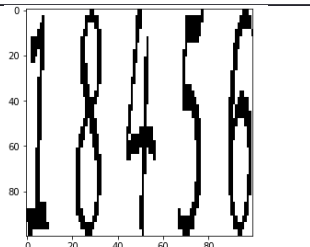
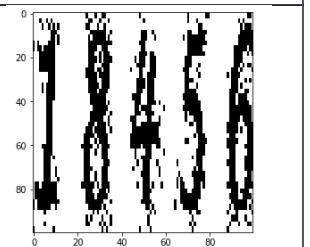
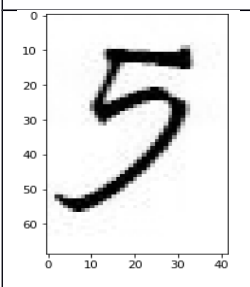
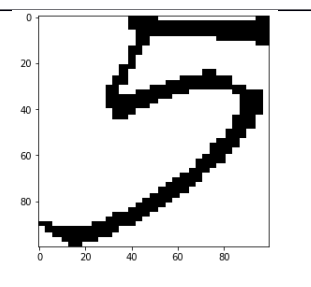
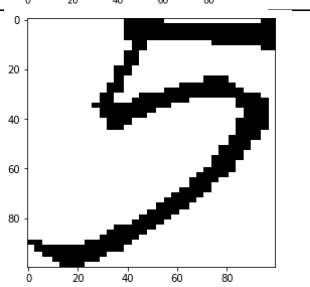
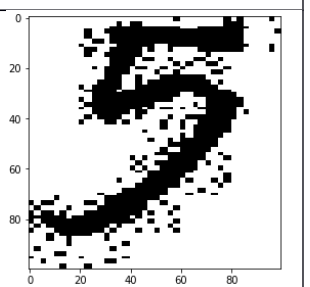
```
def reconnaissance_chiffre(image, liste_modeles, S):  
    image.binarisation(S)  
    image.localisation()  
    L=[]  
    for i in range(len(liste_modeles)):  
        L.append(image.similitude(liste_modeles[i]))  
    return(L.index(max(L)))
```

Cette fonction a été très intéressante à faire puisqu'elle reprend les fonctions que nous avons effectué précédemment. On a commencé d'abord par « binariser » l'image que nous avons au départ, par le seuil **S**, puis, nous avons localisé l'image. Nous avons ensuite crée une liste **L** que l'on remplira avec chaque valeur de la fonction **similitude** appliquée à chaque élément de la liste des modèles de caractères. Ensuite, on retourne l'indice de la liste **L** pour lequel la similitude entre les deux images est la plus grande.

Avant de faire la **Question 6**, nous avons décidé de lancer le programme **test_Image** et **test_reconnaissance** afin d'être sûr que nos fonctions étaient justes. Voir [Annexe 1](#), [2](#)).

Question 6

Pour cette question, nous avons modifié l'image de départ et le seuil. Voici un tableau avec quelques exemples de caractères :

Seuil S	70	120	250
Image initiale			
			
			

On remarque que cela fonctionne bien, même pour un groupement de caractères.

IV. Conclusion

Ce TP à été globalement très intéressant à mener tant pour développer ses compétences en python ou pour découvrir un domaine de l'informatique qui est courant. Nous avons eu quelques difficultés, notamment sur la fonction **similitude** mais elles étaient pas rédhibitoire. Nous avons aussi beaucoup apprécié de pouvoir réaliser des tests sur les fonctions que nous avons écrites, cela nous a permis de voir rapidement ou était nos erreurs.

Annexe :

1) Tests du fichier **test_Image**.

```
In [2]: runfile('E:/COURS/tp2-reconnaissance-chiffres-tp2_jollet_criscuolo-
main/tests/test_image.py', wdir='E:/COURS/tp2-reconnaissance-chiffres-
tp2_jollet_criscuolo-main/tests')
.....Reloading modules: image, reconnaissance
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/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 0.656s

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

2) Tests du fichier **test_reconnaissance**.

```
.....
C:\Anaconda3\lib\site-packages\skimage\transform\warps.py:105: UserWarning
The default mode, 'constant', will be changed to 'reflect' in skimage 0.15.
  warn("The default mode, 'constant', will be changed to 'reflect' in "
C:\Anaconda3\lib\site-packages\skimage\transform\warps.py:110: UserWarning
Anti-aliasing will be enabled by default in skimage 0.15 to avoid aliasing
artifacts when down-sampling images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
..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/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 3 tests in 0.303s

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