

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

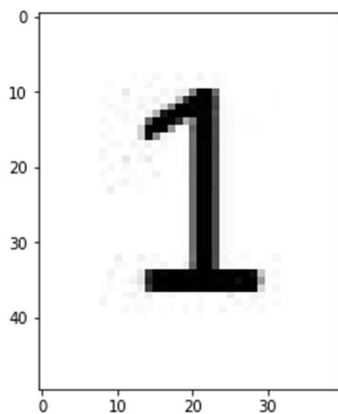
I. Introduction

Les images numériques sont représentées par des tableaux à 2 dimensions, où chaque élément du tableau correspond à un pixel. 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. Prise en main de l'environnement

Voici ce que nous affiche l'exécution du fichier main :

```
In [1]: runfile('C:/Users/schermat/Desktop/tp2-reconnaissance-chiffres-tp2_barroux_scherrerm-main/src/main.py',  
wdir='C:/Users/schermat/Desktop/tp2-reconnaissance-chiffres-tp2_barroux_scherrerm-main/src')  
lecture image : ../assets/test2.JPG (50x40)
```



Traceback (most recent call last):

III. Travail préparatoire

1. Analyser attentivement la classe `Image` et remarquer les attributs `H`, `W` qui indiquent la taille de l'image et l'attribut `pixels` qui contient un tableau 2D numpy contenant les valeurs de l'image en pratique.


`H` correspond à la hauteur de l'image et `W` correspond à la largeur de l'image. Pour ce qui est de l'attribut `pixels`, il correspond à un tableau que l'on crée avec tous les pixels constituant l'image. L'attribut `shape` de la fonction `set_pixels` permet de connaître la taille de l'image.

2. Écrire une méthode `binarisation(self, S)` à la classe `**Image**` qui permet de passer d'une image codée sur 256 valeurs à une image avec seulement deux valeurs (0 ou 255).

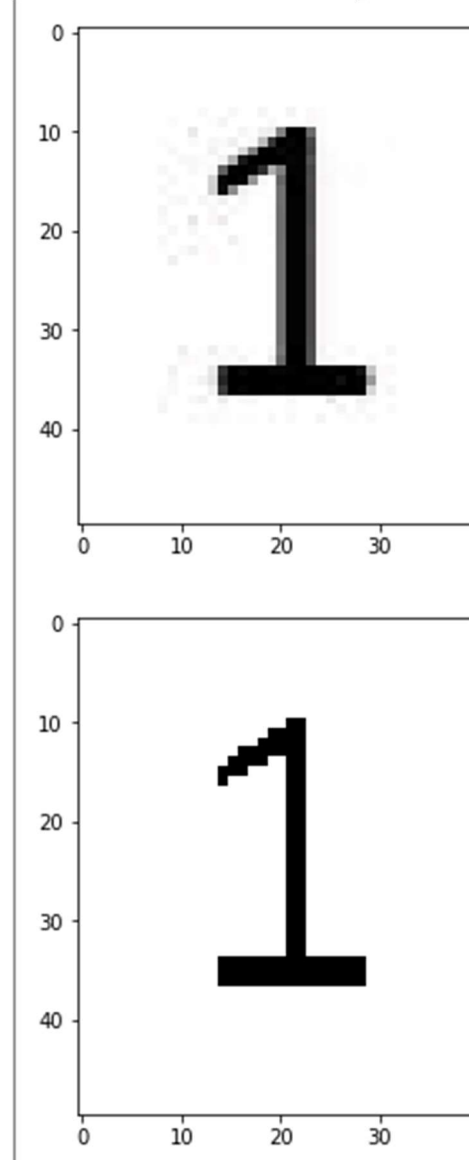
On doit créer une double boucle qui va se balader dans l'image en largeur et en hauteur et comparer si la valeur du pixel en position `[i ; j]` est supérieur (dans ce cas il prendra la valeur 255) ou inférieur (il prendra la valeur 0).

```
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] = 255  
            else :  
                im_bin.pixels[i][j] = 0  
    return im_bin
```

En testant dans GitHub, on obtient donc :

 **Autograding complete**
Points 8/25

On note aussi, la différence entre la première image de la partie II et maintenant, la binarisation a eu lieu :



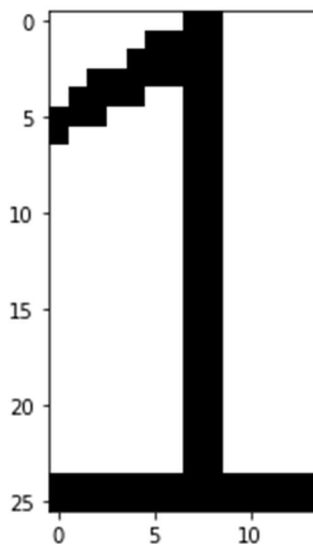
3. Écrire une méthode localisation(self) à la classe Image calculant et retournant l'image recadrée sur le chiffre à identifier.

Pour cette méthode, on va repartir avec le même système de boucle imbriquée avec deux for pour se balader dans tous les lignes et colonnes de l'image. Ensuite, on crée une condition sur le fait de tomber sur un pixel de valeur 0 (couleur noir) et affecter à la valeur de `c_min` et `l_min` les valeurs respectives de `i` et `j`. On comparera à chaque fois `i` et `j` à aux valeurs `c_min` et `l_min` pour que ce soit toujours la plus petite valeur stockée.

Pour les valeurs des max, on fait la comparaison inverse entre `i`, `j` et `c_max`, `l_max`.

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

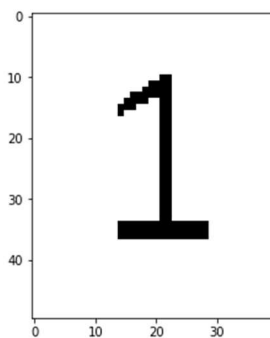
Voici ce que nous affiche l'exécution du programme :



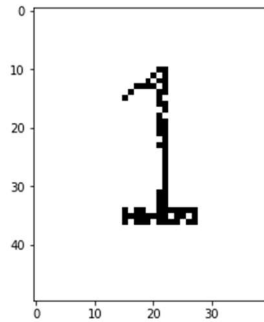
IV. Reconnaissance automatique de chiffre

1. Essayer de lancer le fichier main.py et de voir le résultat de la méthode de binarisation. Essayer avec différentes valeurs de seuil et observez le résultat.

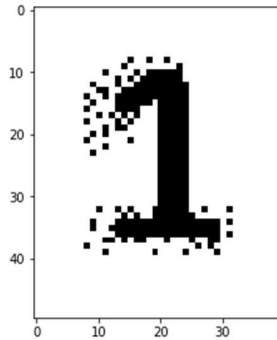
S=70 : le 1 est fidèlement reproduit .



S=5 : certaines cases noires sont devenues de couleurs blanches.



S=250 : les cases noires restent noires et les cases grises deviennent noires.



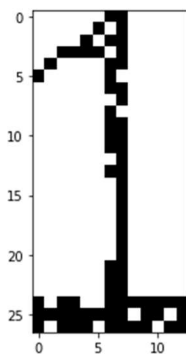
Lorsque l'on lance le fichier de test image, les erreurs de binarisation ne s'affiche plus :

```
Ran 22 tests in 0.141s
```

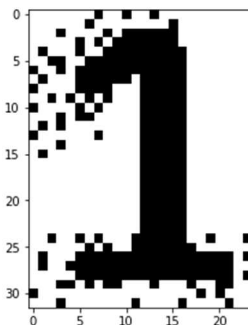
```
FAILED (failures=6, errors=2)
```

2. Essayer de lancer le fichier main.py et de voir le résultat de la méthode de localisation. Essayer avec différentes valeurs de seuil et observez le résultat.

Voici l'affichage du main de la méthode localiasation pour un seuil de 5 :



Pour un seuil de 250 :



On lance aussi le test image, et les erreurs de localisation ne s'affiche pas.

```
Ran 22 tests in 0.141s
```

```
FAILED (failures=6, errors=2)
```

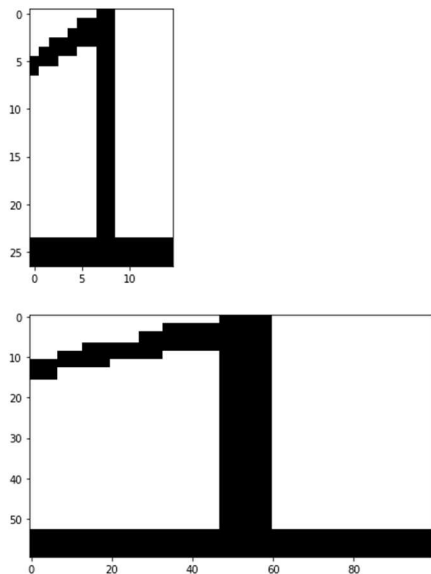
3. Ajouter à la classe Image la méthode **`**resize(self,new_H,new_W)`** qui redimensionne l'image à la taille voulue et renvoie un autre objet de type Image en sortie. Tester cette méthode dans le fichier main.py sur l'image obtenue par l'étape de localisation.

On crée une nouvelle image. On réduit l'image self avec la méthode que l'on nous donne puis insère le tableau de l'image que l'on vient de réduire dans la nouvelle image.

Cela nous retourne la nouvelle image. Voici notre code :

```
#####  
# Methode de redimensionnement d'image  
#####  
def resize(self, new_H, new_W):  
    new_im = Image()  
    a = resize(self.pixels, (new_H, new_W), 0)  
    new_im.set_pixels(np.uint8(a*255))  
    return new_im
```

Le main nous affiche donc ça :



Le nombre d'erreur a diminué et les test resize ne s'affiche plus :

Ran 22 tests in 0.141s

FAILED (failures=4)

4. Ajouter à la classe Image, la méthode **`similitude(self, image)`** qui mesure la similitude par corrélation d'images entre l'image représenté par l'objet courant (self) et un objet de type Image entrée en paramètre.

La méthode similitude va retourner le nombre de pixel similaire entre l'image self et une autre image divisée par le nombre de pixel de l'image. Pour comparer les pixels, on fait deux boucles for imbriquées qui vont regarder si les pixels à la position i,j sont identiques entre les images.

Voici notre script :

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

Il n'y a donc plus d'erreur :

Ran 22 tests in 0.172s

OK

Nous arrivons à 22/25 test validés :

 **Autograding complete**
Points 22/25

5. Dans le fichier reconnaissance.py, écrire la fonction reconnaissance_chiffre(image, liste_modeles, S) qui va effectuer la reconnaissance de chiffre sur l'image image donnée en entrée de la fonction.

On binarise, localise et resize l'image à l'aide des méthodes que l'on a développées précédemment. Ensuite, on crée une liste avec toutes les similitudes entre l'image et les modèles. Enfin, on retourne la valeur maximale de cette liste :

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    sim=[]  
    im_bin = image.binarisation(S)  
    im_loc = im_bin.localisation()  
    for i in liste_modeles:  
        im_res = im_loc.resize(i.H, i.W)  
        sim.append(im_res.similitude(i))  
    return sim.index(max(sim))
```

On fait les tests reconnaissance : pas d'erreurs.

Ran 25 tests in 0.234s

OK

On obtient :

Le chiffre reconnu est : 1

Ainsi, l'image la plus similaire est la seconde dans la liste de modèle.

6. Essayer la fonction de reconnaissance en modifiant l'image de test dans main.py avec différentes images disponibles dans assets/ et en modifiant également le seuil avec quelques valeurs (pas besoin d'en faire 100...).

Nom fichier	Maximum de la mesure de corrélation	Chiffre reconnu	Seuil S
Test1.JPG	0.884	4	70
Test1.JPG	0.86	4	45
Test3.JPG	0.901	2	95
Test3.JPG	0.906 (maximum que l'on peut obtenir)	2	90
Test5.JPG	0.775	2	90
Test5.JPG	0.776	2	85

On note que la mesure de corrélation maximale est obtenue pour des seuils différents suivant les fichiers.

V. Conclusion

Pour compléter tous les tests, il faut renvoyer un entier dans la méthode reconnaissance_chiffre sans la mesure de corrélation. Le système de reconnaissance fonctionne puisque nous obtenons 25/25 pour les tests sur Github et que les mesures de corrélation sont proches de 1 (plus ou moins selon les fichiers).

Durant ce TP, nous avons pu donc apprendre à faire de la reconnaissance d'image.