DEVIGNE Lisa
EMELI Ese
Date du TP (02/12/2021)

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

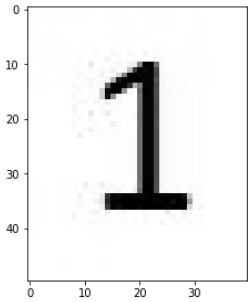
I. Introduction

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. Le principe de cette approche est composé en 5 étapes :

- Étape 1 : binarisation
- Etape 2 : localisation
- Étape 3 : adaptation de la taille au(x) modèle(s)
- Étape 4 : mesure de ressemblance par corrélation
- Étape 5 : Décision

II. Prise en main de l'environnement

Suite à l'exécution du fichier main.py nous avons obtenu l'image suivante :



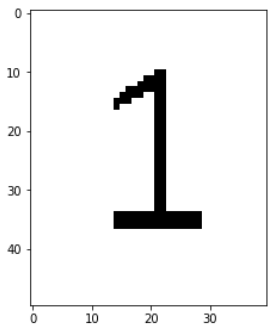
III. Travail préparatoire

1. Question (1)

H représente la hauteur (height), W la largeur (weigh), ils sont initialisés à 0. Self.pixels tableau 2D numpy contenant les valeurs de l'image en pratique, qui est d'abord nul car l'image est de dimension 0,0 ;

2. Question (2)

Pour comparer pour chaque pixel de l'image à une valeur choisie par l'utilisateur (entrée S de la méthode), nous avons fait deux boucles imbriquées, une qui parcours d'abord les lignes, et une qui parcours les colonnes. Ensuite nous avons comparé chaque valeur de pixels à la valeur S, si le pixel est supérieur à S, il devient blanc, sinon il devient noir. Le résultat est donné sous la forme d'une nouvelle image que l'on crée afin de ne pas modifier l'image de base, c'est pourquoi, en début de fonction on crée une image vide (im_bin=Image()).
Nous avons obtenu l'image suivante suite à cette méthode binarisation :



3. Question (3)

Nous n'avons pas utilisé la méthode binarisation car le test était effectué avec une image binarisée dans le main. Nous avons d'abord initialisé les variables cmin, cmax, lmin et lmax qui représentent respectivement la colonne min, colonne max, ligne min, ligne max. Ensuite, on a parcouru l'image de pixels en pixels, en changeant les valeurs des lignes et des colonnes seulement pour les pixels concernés, c'est-à-dire, les pixels noirs. Au fur et à mesure de l'itération, on comparait les indices des colonnes et lignes des pixels pour trouver les minimums et maximums.
Pour finir, on a créé une nouvelle image qui a pour dimensions les colonnes et lignes (min et max) ;

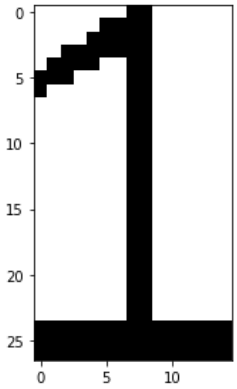
Voici notre code :

```
def localisation(self):

    cmin = self.W
    cmax = 0
    lmin = self.H
    lmax = 0

    for i in range (self.H):
        for j in range (self.W):
            if self.pixels[i,j] == 0:
                if i <= lmin:
                    lmin = i
                if i >= lmax:
                    lmax = i
                if j <= cmin:
                    cmin = j
                if j >= cmax:
                    cmax = j
    img=self.pixels[lmin:lmax+1,cmin:cmax+1]
    imagette = Image()
    imagette.set_pixels(img)
    return imagette
```

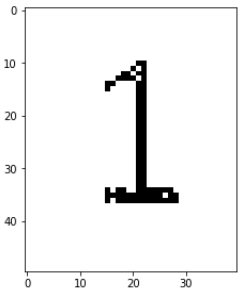
Voici notre résultat :



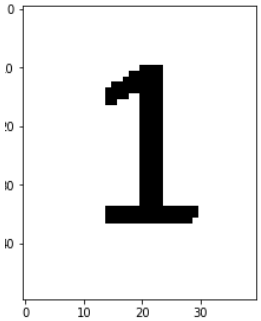
IV. Reconnaissance automatiques de chiffres

1. Question (1)

Premier test binaire : S=7.

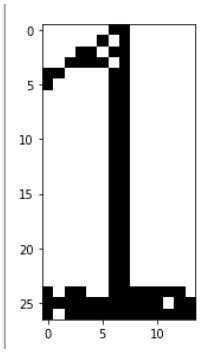


Deuxième test binaire : S=200 .

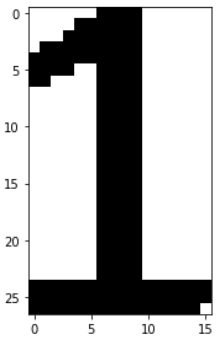


2. Question (2)

Premier test localisation: S=7.



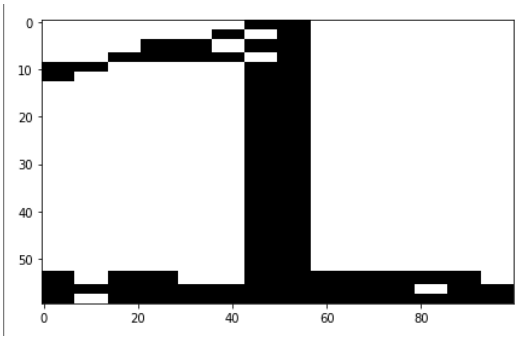
Deuxième test localisation : S=200



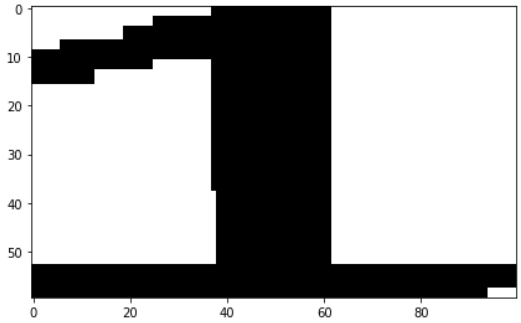
3. Question (3)

Le but de cette méthode est de redimensionner une image à la taille voulue en imposant une longueur (H) et une largeur (W). Cela renvoie un autre objet de type Image en sortie. Les tests sont effectués sur les images localisées de la question précédente. On a choisi les dimensions (60,100).

Premier test resize: S=7.



Deuxième test resize : S=200



Fichier test :

```
.................FFFFReloaded 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)

======================================================================
FAIL: test_similitude_is_not_none (__main__.Test_Image_similitude)
Teste la méthode similitude ne renvoie pas None (oublie de return...).
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:/Users/emeklie/Documents/Cours/INFO501/tp2-reconnaissance-chiffres-info501-tp2_devigne_emekli-main/tests/
test_Image.py", line 206, in test_similitude_is_not_none
    assert image.similitude(image) is not None
AssertionError
```

4. Question (4)

Cette méthode 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. Il faut coller les deux images. Pour cela on redimensionne les images localisées à la même dimensions grâce à la méthode resize (nous avons choisis (60,60)). Ensuite on parcourt les pixels et on compare ceux des 2 images. Si 2 les pixels ont la même intensité, le nombre de similitudes augmente de 1, sinon, rien ne se passe. Pour obtenir, le degré de similitude, il faut diviser le nombre de similitudes par le nombre de pixels.

Voici notre code :

```python
#=============================================================================
# Methode de mesure de similitude entre l'image self et un modele im
#=============================================================================
def similitude(self, im):
    im_bin = self.binarisation(127)
    nb_similitude = 0
    nb_pixels = 0
    res = 0
    self.resize(60, 60)
    im.resize(60, 60)
    for i in range (im_bin.H):
        for j in range (im_bin.W):
            if self.pixels[i][j] == im.pixels[i][j] :
                nb_similitude = nb_similitude+1
            nb_pixels = nb_pixels+1
    res = nb_similitude / nb_pixels
    return res
```

5. Question (5)

Voici notre code :

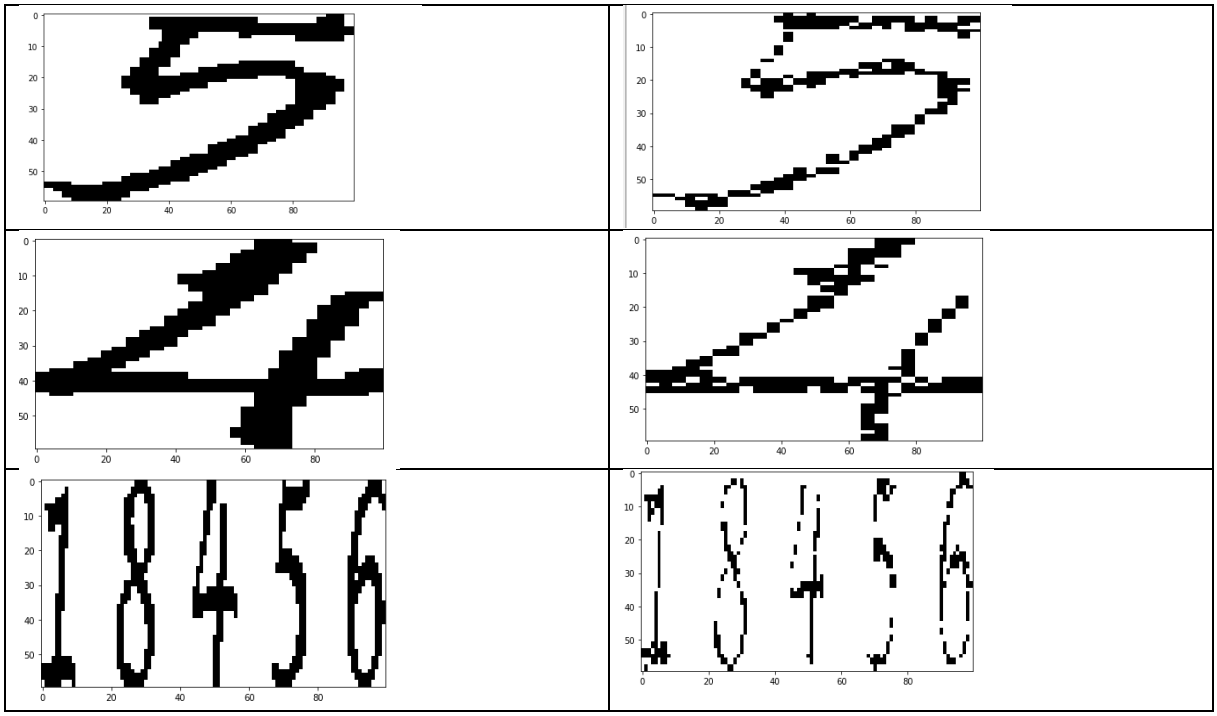
```
def reconnaissance_chiffre(image, liste_modeles, S):
    im_loc= image.localisation()
    similitude_max = 0
    for i in range(0,len(liste_modeles)):
        longueur = liste_modeles[i].H
        largeur = liste_modeles[i].W
        im_loc = im_loc.resize(longueur, largeur)
        similitude = im_loc.similitude(liste_modeles[i])
        if similitude > similitude_max:
            res = i

    return res
```

Nous avons suivi les instructions dans la question.

6. Question (6)

Valeur de seuil S = 200 Valeur de seuil S = 7



V. Conclusion

Expliquer ici l'état d'avancement du TP actuel, les difficultés principales que vous avez rencontrées ainsi que ce que vous avez appris.

Nous avons réussi à finir ce TP. Nous avons principalement rencontré des difficultés pour la fonction localisation. Au début, la fonction ne retournait rien car au départ, notre code ne fonctionnait pas pour le return et l'initialisation de nos variables n'allait pas (il fallait prendre les plus grandes valeurs pour les minimums, et les plus petites pour les maximums pour pouvoir ensuite les comparer). Nous avons aussi rencontré des difficultés pour la fonction reconnaissance chiffre car, au départ, nous n'avions pas pensé à diviser par le nombre de pixels pour obtenir le degré de similitude. Cependant, nous avons appris à utiliser la fonction resize. De plus, ce Tp nous a permis de nous familiariser avec les tableaux 2D.