LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

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

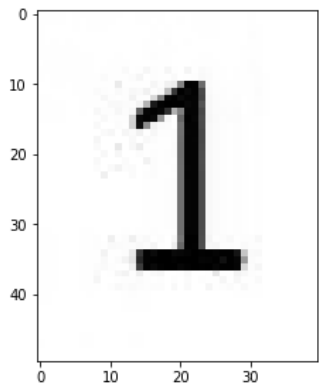
I. Introduction

L'objectif de ce TP est de se familiariser avec la notion d'analyse d'image. Pour ce faire nous allons modéliser des images de chiffres.

On effectue cette analyse par plusieurs étapes: Premièrement la binarisation, ensuite la localisation, après l'adaptation de la taille aux modèles, enfin la mesure de ressemblance par corrélation et pour conclure la décision.

II. Prise en main de l'environnement

```
In [1]: runfile('G:/A3/info python/tp2-recor
info python/tp2-reconnaissance-chiffres-tp2_
lecture image : ../assets/test2.JPG (50x40)
```




```
Traceback (most recent call last):
```

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

En exécutant le fichier main.py, on obtient bien une image, ici on aperçoit que cette image affiche le chiffre 1.

III. Travail préparatoire

1. Question (1).

```
def __init__(self):
    """Initialisation d'une image composee d'un tableau numpy 2D vide
    (pixels) et de 2 dimensions (H = height et W = width) mises a 0
    """
    self.pixels = None
    self.H = 0
    self.W = 0
```

Cette classe initialise notre code en mettant la longueur (H) et la largeur (W) à (0;0), l'origine se trouve en haut à gauche de l'image et l'attribut pixels qui contient un tableau 2D numpy contenant les valeurs de l'image en pratique

2. Question (2).

```
def binarisation(self, S):
    """ creation d'une image vide"""
    im_bin = Image()
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))
    for i in range (len(im_bin.pixels)):
        for k in range (len(im_bin.pixels[0])):
            if self.pixels[i][k]<= S:
                im_bin.pixels[i][k]=0
            else:
                im_bin.pixels[i][k]=255

    return im_bin
```

On choisit une valeur seuil S .

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

On a pour objectif de ne plus que afficher 2 valeur : 0 et 255, qui correspondent aux couleurs noir et blanc. On veut donc binariser les valeurs. On a donc complété le code donné dans l'énoncé avec 2 boucles for pour la taille de la valeur puis on a utilisé un si inférieur à la seuil (=S) afin de déterminer si la valeur est 0 ou 255.

3. Question (3).

```
def localisation(self):
    im_loc = Image()
    lmin = self.H-1
    lmax = 0
    cmin = self.W-1
    cmax = 0
    for l in range(len(self.pixels)):
        for c in range (len(self.pixels[0])):
            if l<lmin:
                if self.pixels [l][c] ==0:
                    lmin = l
            elif l>=lmax:
                if self.pixels [l][c] ==0:
                    lmax = l
            if c<cmin:
                if self.pixels [l][c] ==0:
                    cmin = c
            elif c>=cmax:
                if self.pixels [l][c] ==0:
                    cmax = c
    im_loc.set_pixels(self.pixels[lmin:lmax,cmin:cmax])
    return im_loc
```

Cette méthode permet de calculer et de retourner l'image recadrée sur le chiffre identifié. Les valeurs des coordonnées limites de l'image noir lmin, lmax, cmin, cmax forment un rectangle m. L'image est donc limitée par le rectangle des limites.

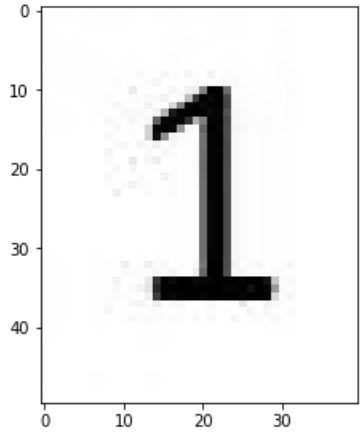
Pour cela on vérifie les valeurs maximales lmin /lmax et cmin/cmax avec des conditions if/elif. Le tout est contenu dans une double boucle for afin de rentrer dans un format de tableau 2D.

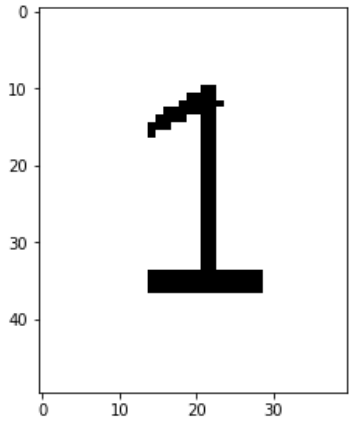
LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

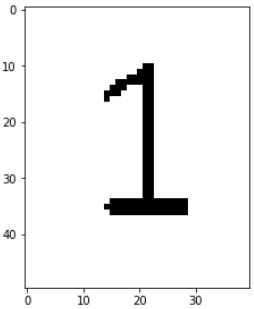
IV. Reconnaissance automatique de chiffre

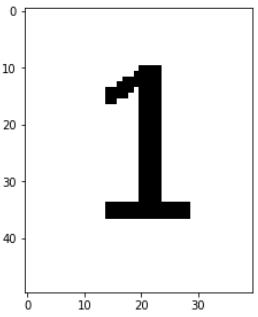
1. Question (1).

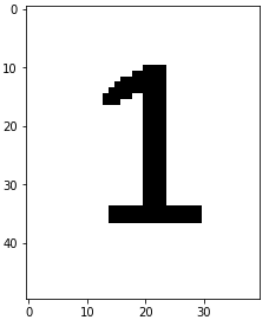
On teste la binarisation avec différente valeur du seuil S :

Original → 

S=70→ 

S=40→ 

S=150→ 

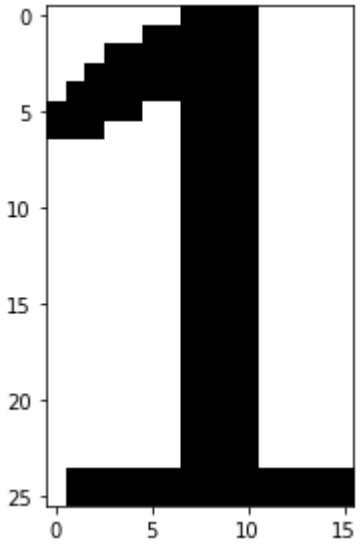
S=220 

Après obtention des différents essais avec différente valeur de seuil, on fait un test avec test_Image.py et comme prévu, nous n'obtenons aucune erreur.

2. Question (2).

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

On teste ici la localisation du chiffre tout en entrant des valeurs de seuil, au bout de plusieurs essai on se rend compte que la localisation du chiffre est inchangée.



Après obtention des différents essais avec différente valeur de seuil, on fait un test avec test_Image.py et comme prévu, nous n'obtenons aucune erreur.

3. Question (3).

Ici, on nous demande de redimensionner l'image, on créer donc un class resize avec les nouvelles valeurs qui vont initialiser l'origine.

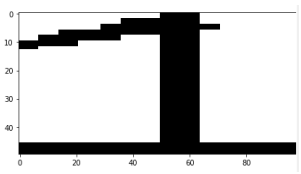
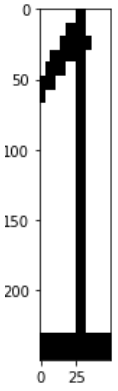
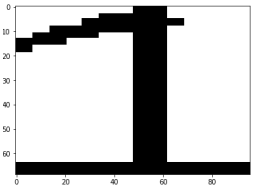
```
def resize(self, new_H, new_W):
    newIm = Image()
    pixels_resized = resize(self.pixels, (new_H,new_W), 0)
    newIm.set_pixels(np.uint8(pixels_resized*255))

    return newIm
```

On renseigne donc les nouvelles valeurs de la largeur (new_W) et de la longueur (new_H).

On test ensuite en faisant varier les valeurs de resize, on voit que l'image se modifie avec la nouvelle largeur et la nouvelle longueur

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

(50;100) →  (250;50) →  (69;96) → 

On fait bien attention à repasser resize en entier qui aurait pu devenir un float avec laide de (np..uint8(pixels_resized*255)).

Après obtention des différents essais avec différente valeur de seuil, on fait un test avec test_Image.py et comme prévu, nous n'obtenons aucune erreur.

4. Question (4).

On nous demande de faire une classe similitude afin de voir les similitudes entre les images.

```
def similitude(self, im):
    simi = 0
    compte = 0
    for l in range (len(self.pixels)):
        for c in range (len(self.pixels[0])):
            compte += 1
            if self.pixels[l][c] == im.pixels[l][c]:
                simi+= 1
     return(simi/compte)
```

On fait donc une boucle pour différencier chaque pixel des 2 images, à chaque test on ajoute +1 a la variable compte. Dès que l'on obtient un pixel identique dans les 2 images, on ajoute +1 à la variable simi. Ensuite on finit cette classe en faisant simi / compte.

5. Question (5).

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021

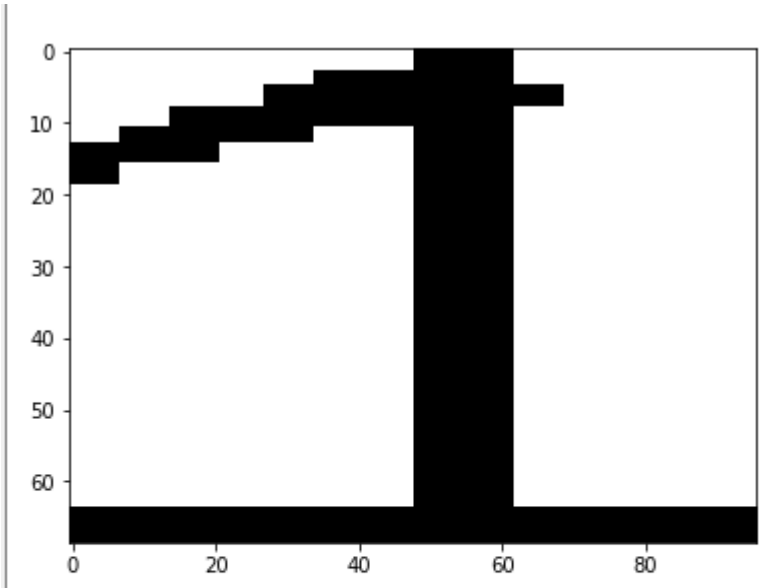
On nous demande de faire une fonction reconnaissance de chiffre, on utilise une variables correspondant à l'image, une autre aux différents modèles et une dernière avec le seuil.

```
def reconnaissance_chiffre(image, liste_modeles, S):
    image =image.binaisation(S)
    image = image.localisation()
    sim = 0
    chiffre = 0
    for i in range (len(liste_modeles)):
        image2 = liste_modeles[i]
        image = image.resize(image2.H, image.W)
        im = image.similitude(image2)
        if im > sim:
            sim = im
            chiffre = 1
    return chiffre
```

On binarise puis localise l'image. Ensuite on test la similitude entre les chiffres tout en les redimensionnant l'image afin qu'ils aient les mêmes références. Pour finir on test le pourcentage de similitude pour savoir de quelle chiffre on se rapproche le plus/

6. Question (6).

LE BOUILLE Brice
PASQUIER Paul
MM3-TDB2
Date du TP : 1 Décembre 2021




```
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
```

En modifiant le fichier de test on obtient le bon nombre à chaque fois. Pour test2 on obtient bien 1 et pour test3 on obtient bien 2.

La valeur de seuil optimal est de 70.

V. Conclusion

Grâce à ce Tp, il nous a été possible de comprendre et d'appliquer les différentes méthodes et paramètres pour analyser une image. De plus l'aspect reconnaissance de l'image est très pratique pour comprendre les limites de ces méthodes.