GUILBERT Jérémy
MONTUORI François
Date du TP (03/12/1998)

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

I. Introduction

Pour ce TP, nous voyons une méthode pour la reconnaissance automatique de caractères appelée reconnaissance par corrélation avec des modèles. Cette reconnaissance sera effectuée en cinq étapes détaillées dans ce TP, que nous effectuerons sous Python.

II. Travail préparatoire

1. Question (1)

On remarque effectivement que les attribus H et W, pour Height et Width déterminent la taille de l'image. Pixel est quand à lui un tableau bidimensionnel, chaque dimension correspondant à H et W.

2. Binarisation

L'idée de cette méthode est de recréer une image binaire, c'est-à-dire contenant exclusivement des pixels tout blancs ou tout noirs et en sélectionnant manuellement la limite de nuances de gris voulues pour trancher entre le blanc ou le noir.

La méthode fonctionne avec deux paramètres : self, l'image que l'on veut binariser et S la valeur arbitrairement choisie du seuil.

On crée d'abord une matrice remplie de 0 de dimension HxW en initialisation.

On balaye ensuite l'image de base : si le pixel rencontré possède un code de couleur supérieur à S, le pixel devient blanc (code 255). Sinon le pixel devient ou reste noir, ce qui correspond au code 0.

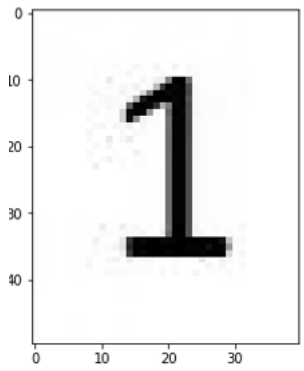
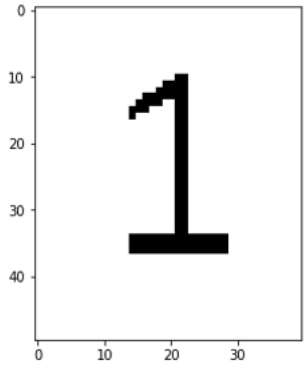


Image originale



Binarisation avec S=70

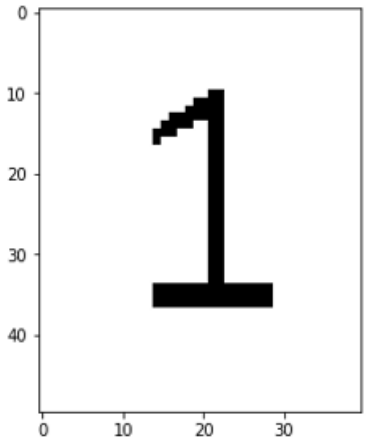
3. Localisation

L'idée de la localisation est d'obtenir une image de taille plus petite possible contenant notre caractère. Ainsi on obtient une image rectangulaire où les bords de l'image sont tangeants aux bords du caractère.

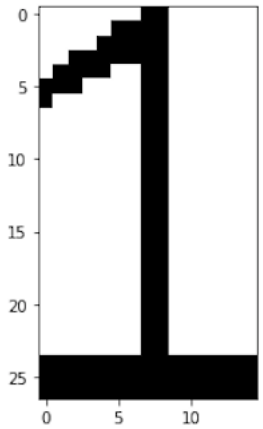
La méthode fonctionne avec un seul paramètre qui est self, l'image d'entrée.

Pour les valeurs initiales, on se place dans les extremums (dans l'hypothèse peu probable ou on ne pourrait pas effectuer un recadrage meilleur). Soit self.H-1 et self.W-1. Comme le comptage de pixels commence à 0, on doit en effet soustraire 1 à self.H et self.W pour avoir la hauteur et la largeur effectives.

En parcourant l'image, si on rencontre un pixel de couleur noire, et si sa position est à l'intérieur du contour, alors sa position devient une nouvelle limite du contour de l'image.



Cadrage initial après binarisation



Localisation

III. Reconnaissance automatique de chiffre

1. Tests binarisation (1)

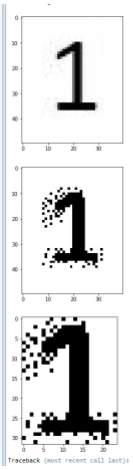
2. Test localisation (2)

On effectue le test de binarisation et localisation avec deux valeurs de seuil « extrêmes » : S=250, où tout pixel non blanc est noirci au maximum et S=10, cas où seulement les valeurs de noir très foncées sont conservés, et les autres sont blanchis.

Les résultats de ces tests suivants sont affichés ci-dessous :

S=250

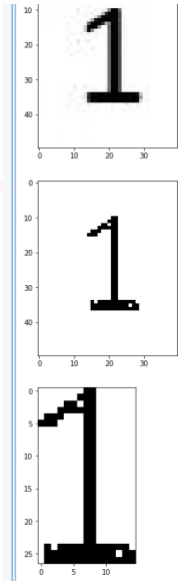
```
import matplotlib.pyplot as plt
from image import Image
from reconnaissance import reconnaissance_chiffre, lecture_modeles

if __name__ == '__main__':

    # Variables utiles
    path_to_assets = "../assets/"
    plt.ion() # Mode interactif de matplotlib pour ne pas bloquer l'éxécutions lorsque l'on fait display

    #-----------------------------------------------------------------------
    # Lecture image et affichage
    #-----------------------------------------------------------------------
    image = Image()
    image.load(path_to_assets + "test2.JPG")
    image.display("Exemple d'image")

    #-----------------------------------------------------------------------
    # Binarisation de l'image et affichage
    #-----------------------------------------------------------------------
    S = 250
    image_binarisee = image.binarisation(S)
    image_binarisee.display("Image binarisee")

    #-----------------------------------------------------------------------
    # Localisation de l'image et affichage
    #-----------------------------------------------------------------------
    image_localisee = image_binarisee.localisation()
    image_localisee.display("Image localisee")

    #-----------------------------------------------------------------------
    # Redimensionnement de l'image et affichage
    #-----------------------------------------------------------------------
    image_resizee = image_localisee.resize(100, 500)
    image_resizee.display("Image redimensionnee")

    #-----------------------------------------------------------------------
    # Lecture modeles et reconnaissance
    #-----------------------------------------------------------------------
    liste_modeles = lecture_modeles(path_to_assets)
    chiffre = reconnaissance_chiffre(image, liste_modeles, 70)
    print("Le chiffre reconnu est : ", chiffre)
```



Traceback (most recent call last):

S=10

```python
if __name__ == '__main__':

    # Variables utiles
    path_to_assets = '../assets/'
    plt.ion() # Mode interactif de matplotlib our ne pas bloquer l'éxécutions lorsque l'on fait display

    #=========================================================================
    # Lecture image et affichage
    #=========================================================================
    image = Image()
    image.load(path_to_assets + 'test2.JPG')
    image.display("Exemple d'image")

    #=========================================================================
    # Binarisation de l'image et affichage
    #=========================================================================
    S = 10
    image_binarisee = image.binarisation(S)
    image_binarisee.display("Image binarisee")

    #=========================================================================
    # Localisation de l'image et affichage
    #=========================================================================
    image_localisee = image_binarisee.localisation()
    image_localisee.display("Image localisee")

    #=========================================================================
    # Redimensionnement de l'image et affichage
    #=========================================================================
    image_resizee = image_localisee.resize(100, 500)
    image_resizee.display("Image redimensionee")

    #=========================================================================
    # Lecture modeles et reconnaissance
    #=========================================================================
    liste_modeles = lecture_modeles(path_to_assets)
    chiffre = reconnaissance_chiffre(image, liste_modeles, 70)
    print("Le chiffre reconnu est : ", chiffre)
```



3. Resize

Pour la méthode resize, on veut simplement redimensionner modifiant notre image de base de dimensions H*W par une image de dimensions new_H*new*W : Pour ce faire on utilise les fonction et commande fournies par le sujet : On obtient la syntaxe et l'image nouvellement dimensionnée suivante :

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

4. Similitude

Le but est de connaitre la similitude d'une image avec une autre. On balaie comme dans les autres méthodes décrites, on introduit une variable de comptage h on pose la condition d'égalité : si les pixels sont égaux h devient h+1.
On divise ensuite h final par le nombre total de pixels, ce qui nous renvoie le pourcentage de similitude entre les deux images.

5. Reconnaissance

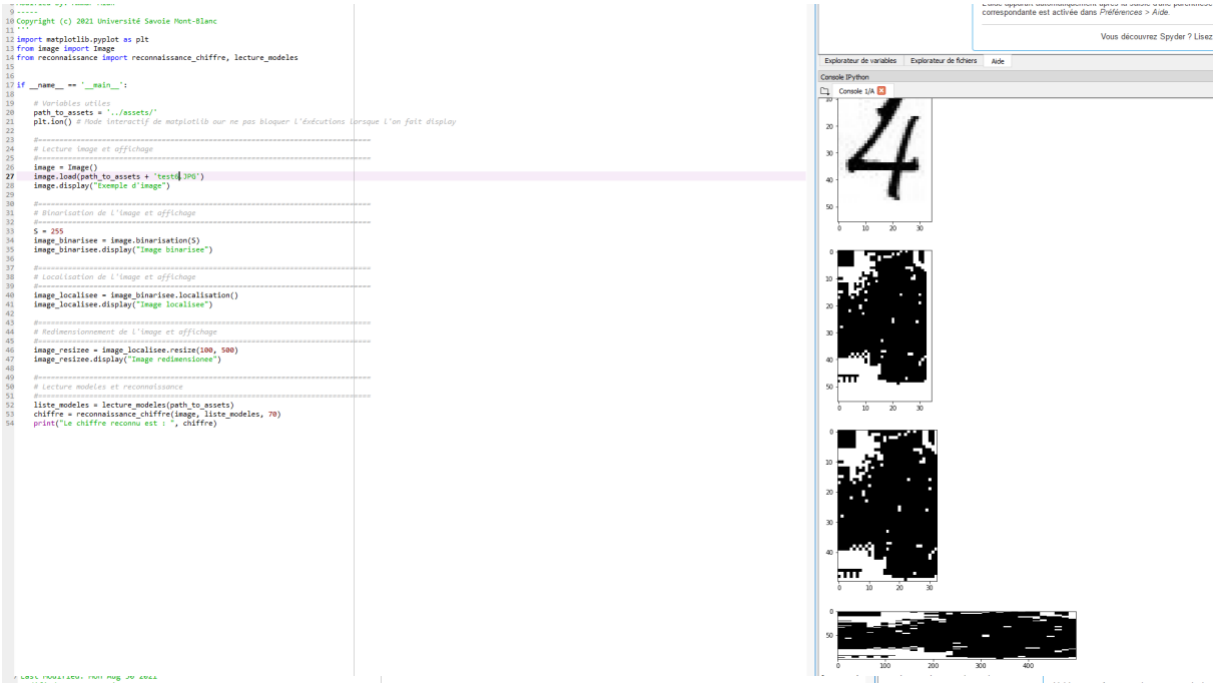
Avec la fonction **reconnaissance_chiffre(image, liste_modeles, S)**, on veut que notre programme puisse associer un des chiffres avec une image en entrée. Les chiffres sont eux associés à des images correspondantes contenues dans une base de données. Pour cela on utilise les méthodes écrites dans la partie II. Ensuite, on parcours la liste d'images et on compare la similitude de chacune telle que dd2crite dans la partie III.4
L'image ayant la plus grande similitude est sélectionnée avec son chiffre associé.

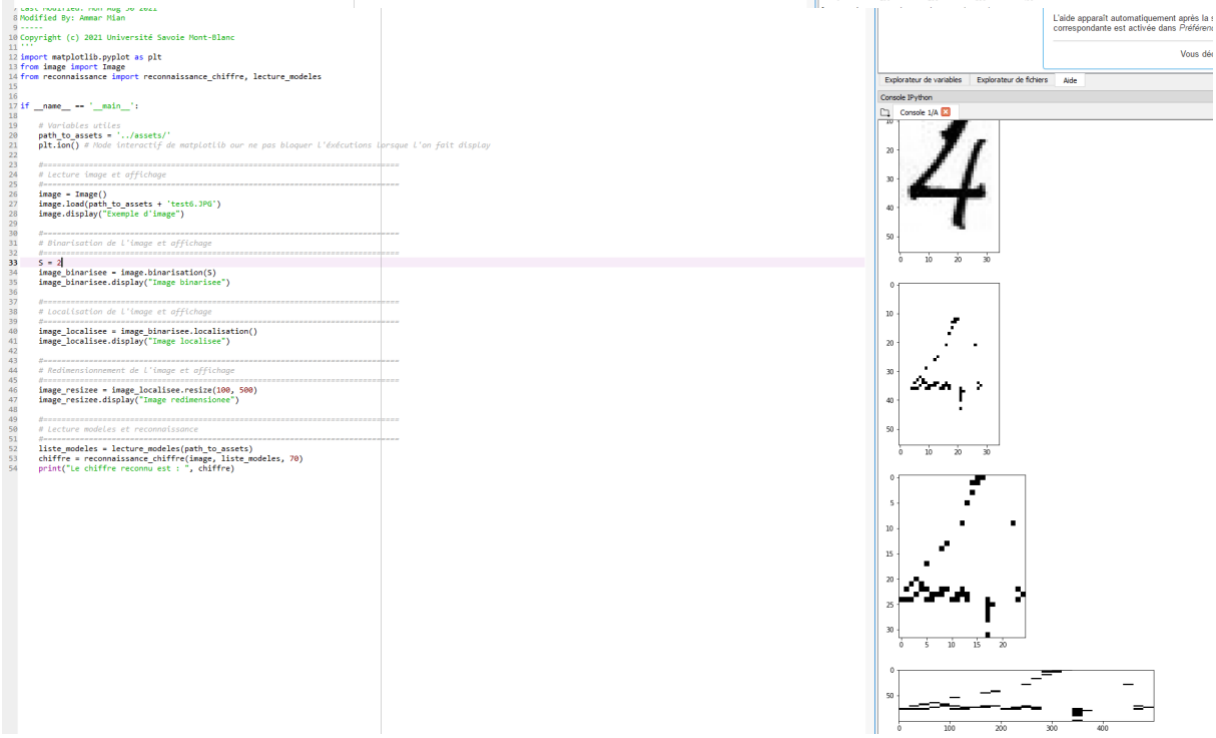
6. Tests

Pour les test, nous avons procédé expérimentalement, entrant manuellement des seuils croissants très bas et très hauts pour connaitre les valeurs limites. Nous avons effectué ces tests sur trois figures et avons été assez surpris par la faculté du programme à trouver les bonnes similitudes malgré des valeurs de seuil extrêmes.

On retrouve ci-dessous un tableau pour les 3 figures testées ainsi que deux valeurs de seuils extrêmes pour la figure 6.

Nom Image	caractère	Valeur min S	Valeur max S
Test1	4	2	255
Test4	2	3	250
Test6	4	2	255





IV. Conclusion

Sur ce TP, notre principal frein a été une nouvelle fois la syntaxe Python qui nous a beaucoup ralenti. Nous aurions aimé faire plus de tests avec d'autres figures mais nous avons manqué de temps sur la fin. On sent tout de même que l'on a un peu progressé depuis le TP 1 et que nous sommes moins perdus dans les syntaxes.
Au niveau de la méthode, bien que le concept soit appréhendé sur un cas simple, cela nous donne une idée de comment peuvent fonctionner de nombreux systèmes de reconnaissance et cette proximité avec le monde « réel » nous a plus.