STEVENIN Valentin
Date du TP (03/12/2021)

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

I. Introduction

1. Représentation des images numériques¶

Les images numériques sont représentées par des tableaux à 2 dimensions, où chaque élément du tableau correspond à un pixel. Dans le cas d'une image en niveaux de gris, la valeur de chaque élément du tableau correspond à l'intensité du pixel, cette intensité étant généralement codée sur 8 bits, la valeur 0 correspondant au noir et la valeur 255 au blanc et les valeurs intermédiaires correspondant à différents niveaux de gris. Dans le cas d'une image couleur, chaque élément du tableau (c'est-à-dire chaque pixel) est caractérisé par trois grandeurs : une intensité de Rouge, une intensité de Vert et une intensité de Bleu, chacune de ces intensités étant comprises entre 0 et 255.

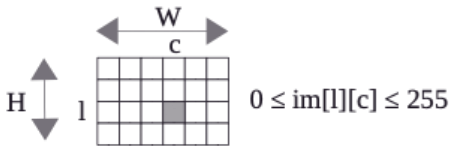


Image en niveaux de gris

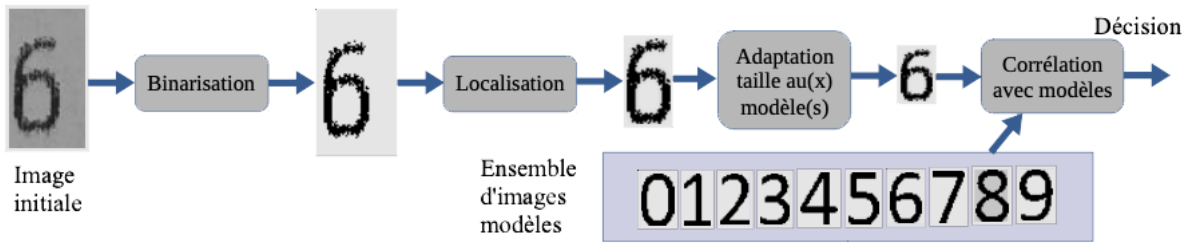
Image RVB couleur

Dans la suite de ce TP, nous n'utiliserons que des images en niveaux de gris.

2. La reconnaissance automatique de caractères¶

Aujourd'hui, l'analyse automatique d'images numériques est utilisée dans de très nombreux domaines : en médecine (assistance au geste chirurgical), dans l'industrie (contrôle qualité en bout de chaîne de fabrication), pour la surveillance de la surface du globe (suivi du déplacement des glaciers par imagerie satellitaire), en robotique (aide au déplacement des robots compagnons), en vidéo-surveillance (détection de comportements anormaux dans un lieu public), dans l'accès au savoir (numérisation du fonds des bibliothèques), etc. Parmi toutes les techniques utilisées dans ces différents domaines, la lecture automatique des caractères (OCR : Optical Character Recognition) a maintenant d'excellentes performances, même dans la reconnaissance de l'écriture manuscrite.

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 détaillé ci-dessous :



- **Étape 1 : binarisation** L'image initiale est en niveaux de gris, c'est-à-dire que chaque pixel a une intensité qui est comprise entre 0 (noir) et 255 (blanc). La binarisation transforme l'image en niveaux de gris en une image binaire où les pixels ne peuvent être que **noirs** (intensité = 0) ou **blancs** (intensité = 255). La décision se fait par comparaison à un seuil qui est le paramètre de la méthode et dont le choix peut s'avérer critique. Cette étape permet de simplifier l'image et de faire ressortir l'information utile.
- **Étape 2 : localisation**
  Pour simplifier la recherche par corrélation, on restreint l'image au **rectangle englobant** délimitant le chiffre à reconnaître.
- **Étape 3 : adaptation de la taille au(x) modèle(s)**
  Pour que la recherche par corrélation puisse avoir un sens, il faut que les deux images dont on veut mesurer la ressemblance par corrélation soient de la même taille. Il faudra donc **adapter la taille de l'image analysée** à la taille des différents modèles envisagés. Notons que les modèles peuvent avoir des tailles différentes.
- **Étape 4 : mesure de ressemblance par corrélation**
  La mesure de ressemblance par corrélation entre deux images va consister à compter la proportion de pixels de **même intensité** et situés au **même endroit** dans chacune des deux images. Cette mesure va donc varier de 0 à 1. Une valeur égale à 0 signifie qu'aucun des pixels d'une image n'a la même intensité que le pixel correspondant de l'autre image. Une valeur égale à 1 signifie que chaque pixel d'une image a la même intensité que le pixel correspondant de l'autre image (ressemblance parfaite).
- **Étape 5 : Décision.**
  Association de l'image analysée à l'image modèle de **corrélation maximale**.

II. Travail préparatoire

1. 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). Cela se fait en comparant pour chaque pixel de l'image la valeur du pixel à une valeur choisie par l'utilisateur (entrée **S** de la méthode). Ainsi en pratique, il faut itérer sur tous les pixels du tableau numpy et comparer la valeur au seuil. On veut également que le résultat soit donné sous la forme d'une nouvelle image que l'on crée dans la fonction afin de ne pas mmodifier l'image de base.

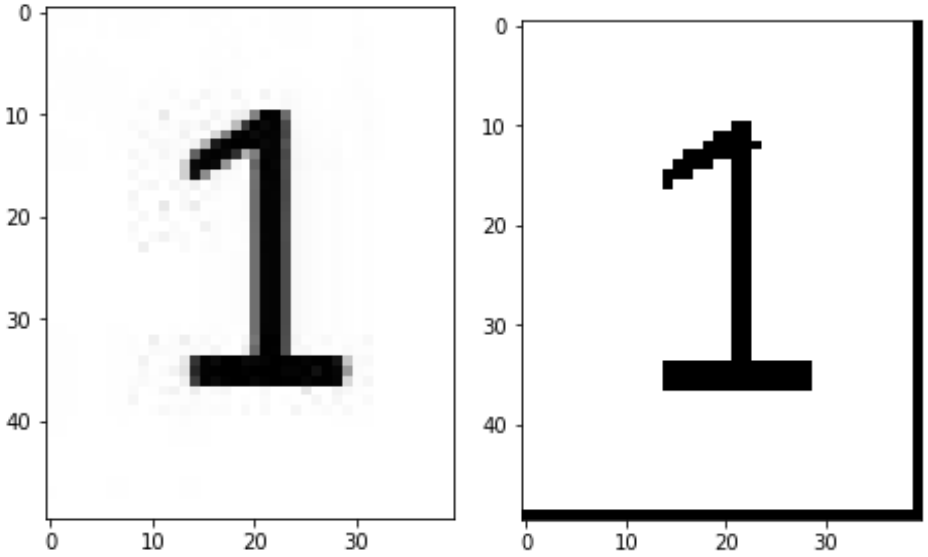
```python
def binarisation(self, S):
    # creation d'une image vide
    im_bin = Image()

    # affectation a l'image im_bin d'un tableau de pixels de meme taille
    # que self dont les intensites, de type uint8 (8bits non signes),
    # sont mises a 0
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))

    # TODO: boucle imbriquees pour parcourir tous les pixels de l'image im_bin
    # et calculer l'image binaire
    for i in range(self.H-1):
        for j in range(self.W-1):
            if self.pixels[i][j]>S:
                im_bin.pixels[i][j]=255

    return im_bin
```

Résultat :

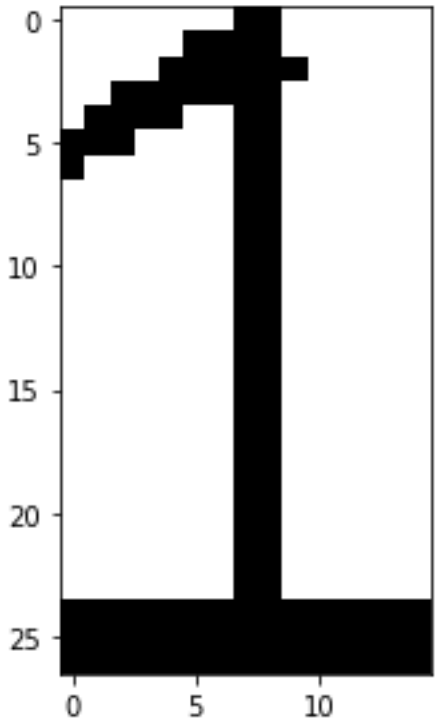


2. Écrire une méthode **localisation(self)** à la classe **Image** calculant et retournant l'image recadrée sur le chiffre à identifier. Le principe de ce recadrage consiste, sur une image binaire **im_bin** de dimension $H \times W$H×W, à déterminer les coordonnées l_{min}lmin, l_{max}lmax, c_{min}cmin et c_{max}cmax du rectangle englobant la forme noire (valeurs 0) et à construire l'image limitée à ce rectangle englobant.

```
def localisation(self):
    lmin=self.H-1
    lmax=0
    Cmin=self.W-1
    Cmax=0
    for i in range(self.H-1):
        for j in range(self.W-1):
            if self.pixels[i][j]<255:
                if i<lmin:
                    lmin=i
                elif j<Cmin:
                    Cmin=j
                elif i>lmax:
                    lmax=i
                elif j>Cmax:
                    Cmax=j
    im_loc=Image()
    im_loc.set_pixels(self.pixels[lmin:lmax+1,Cmin:Cmax+1])


    return im_loc
```

Résultat :



III. Reconnaissance automatique de chiffre

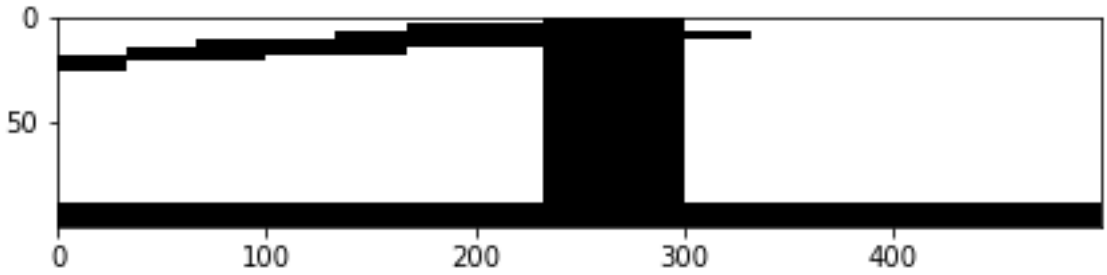
1. Ajouter à la classe Image la méthode **resize(self,new_H,new_W) **qui redimensionne l'image à la taille voulue et renvoie un autre objet de type Image en sortie. Tester cette méthode dans le fichier main.py sur l'image obtenue par l'étape de localisation. Pour ce test, on choisira des dimensions quelconques, (60,100) par exemple.

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


    return im_resized
```

Un problème rencontrer était le faîtes que l'image (Ici im) une fois redimentionner était un tableau numpy, il m'a donc fallu passer par la création d'une deuxieme image d'après le tableau numpay im.

Résultat :



2. 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 procédure pour le calcul de similitude est décrite dans la présentation du TP.

```
def similitude(self, im):
    score=0
    im1=Image()
    im2=Image()
    im1.set_pixels(self.pixels)
    im1=im1.binarisation(10).localisation()
    im2.set_pixels(im.pixels)
    im2=im2.binarisation(10).localisation()
    im1=im1.resize(im2.H, im2.W)

    for i in range(im2.H-1):
        for j in range(im2.W-1):
            if im1.pixels[i][j]==im2.pixels[i][j]:
                score=score+1
    score=(score/(im2.H*im2.W))
    return score
```

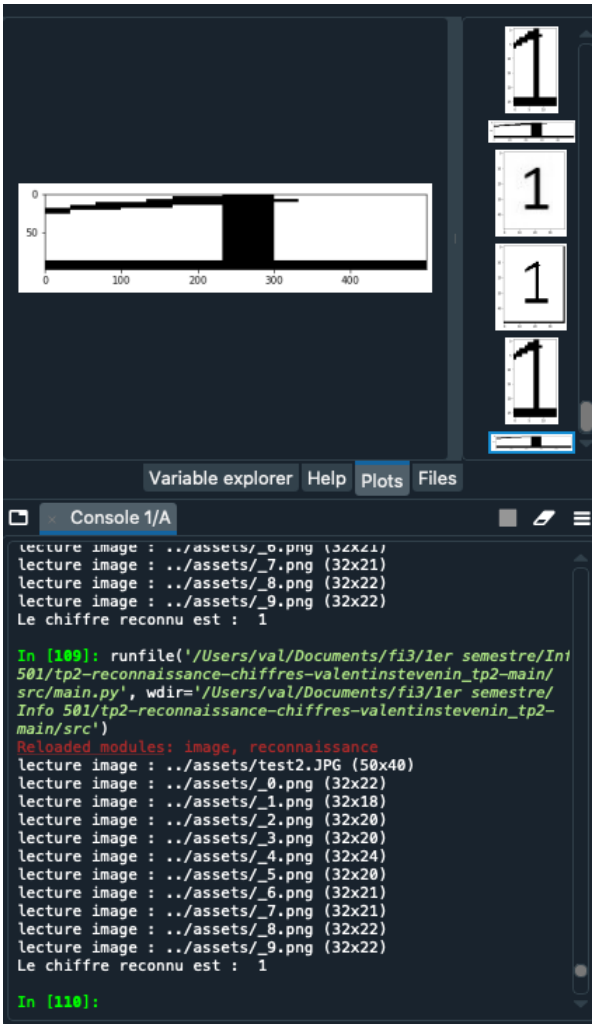
Cette fonction renvoie bien un score entre 0 et 1.

3. Dans le fichier reconnaissance.py, écrire la fonction reconnaissance_chiffre(image, liste_modeles, S)qui va effectuer la reconnaisance de chiffre sur l'image image donnée en entrée de la fonction. Pour cela il faudra dans la fonction tout à tour, binariser l'image et la localiser. Ensuite, il faut calculer sa simiitude à tous les modèles (en redimensionnant l'image à la taille du modèle) et trouver le modèle pour lequel il y a la plus grande similitude. Cela peut être fait en parcourant une liste d'images modèles et en sauvegardant la simimiltude

maximale ainsi que l'indice de l'image avec la similitude maximale. La fonction doit renvoyer un entier compris entre 0 et 9.

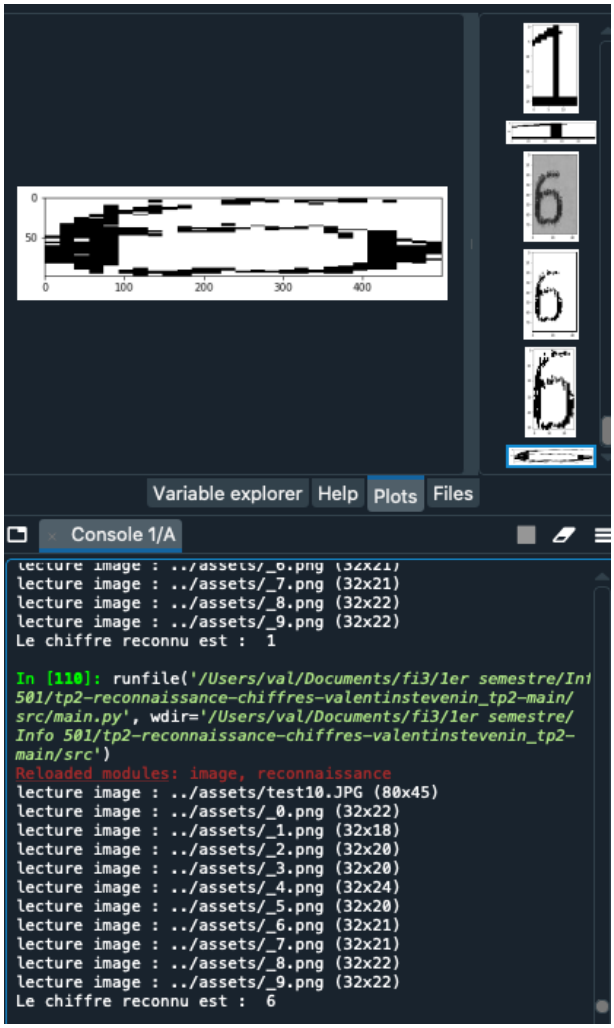
```
def reconnaissance_chiffre(image, liste_modeles, S):
    index=0
    sim=0
    for i in range(len(liste_modeles)-1):
        if image.similitude(liste_modeles[i])>sim:
            sim=image.similitude(liste_modeles[i])
            index=i
    return index
```

Résultat :



On remarque que l'image reconnu est bien le bon chiffre

4. 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…). Présenter les résultats dans le rapport sous forme de tableau (différentes images en ligne et différents seuil en colonne). Proposer une valeur de seuil qui marche le mieux selon vos expérimentations.



On réessaie avec un chiffre différent et on voie que le logiciel marche.

IV. Conclusion

J'ai pu terminer le TP en avance, aucune réelle difficulté n'a été rencontrer.