SURMONT Thomas
NANDRINO Paolo
TD1
TP du 02/12/2021

TP2 - Lecture automatique de chiffres par analyse d'image

Introduction

Dans ce TP l'objectif est d'effectuer une analyse d'image en noir et blanc à l'aide d'un algorithme Python.

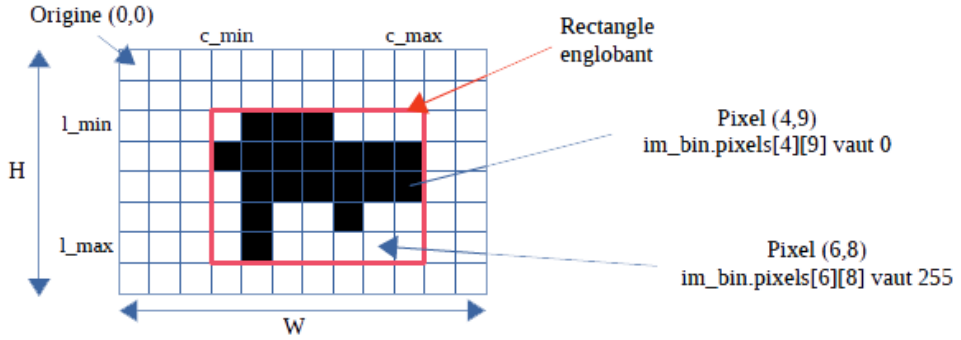
Travail préparatoire

La méthode binarisation :

Pour la binarisation de l'image, on crée un nouvel objet de type Image, dont on remplit chaque case par 0 (noir) ou 255 (blanc).
Concrètement, si le pixel de l'image a une valeur inférieure à celle d'un seuil choisi, le pixel de l'image binarisée sera noir, et s'il est au dessus de cette valeur, le pixel de l'image binarisée sera blanc. On obtiendra ainsi une image noire et blanche uniquement (sans nuances de "gris" comme on pouvait en avoir sur les images test)

```
#=============================================================================
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):
        for j in range (self.W):
            if self.pixels[i,j]<S:
                im_bin.pixels[i,j]=0
            else :
                im_bin.pixels[i,j]=255
    return im_bin
```

La méthode localisation :



Pour trouver les valeurs c_min, c_max, l_min et l_max, on a décidé de faire 4 boucles imbriquées différentes. Le but est de parcourir le tableau de pixels horizontalement puis verticalement, en partant du début puis de la fin, afin de trouver ces 4 points dès que l'on rencontre une case noire.

Après cela, on crée une nouvelle image im_loc, où l'on transpose notre image existante dans le format du rectangle englobant le plus restreint possible.

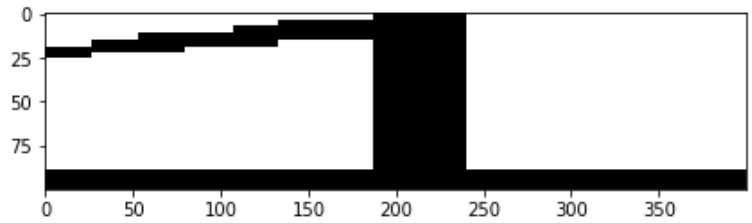
```
def localisation(self):
    c_min=self.W-1
    c_max=0
    l_min=self.H-1
    l_max=0
    step1=False
    step2=False
    step3=False
    step4=False

    for l in range (self.H):
        for c in range (self.W):
            if self.pixels[l,c]==0:
                l_min=l
                step1=True
                break
        if step1==True:
            break


    for c in range (self.W):
        for l in range (self.H):
            if self.pixels[l,c]==0:
                c_min=c
                step2=True
                break
        if step2==True:
            break

    for c in reversed (range (0,self.W)):
        for l in range (self.H):
            if self.pixels[l,c]==0:
                c_max=c
                step3=True
                break
        if step3==True:
            break


    for l in reversed (range (0,self.H)):
        for c in range (self.W):
            if self.pixels[l,c]==0:
                l_max=l
                step4=True
                break
        if step4==True:
            break

    im_loc=Image()
    im_loc.set_pixels(self.pixels[l_min:l_max+1,c_min:c_max+1])
    return im_loc
```

Reconnaissance automatique de chiffre

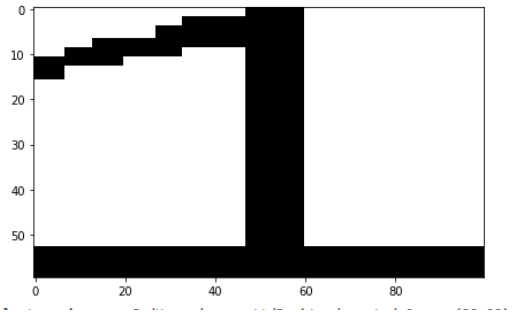
La méthode resize :

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

On teste ensuite cette fonction avec notre “1”, pour différentes valeurs



Avec un resize à (100,400)



Avec un resize à (60,100)

La méthode similitude :

On va commencer par compter le nombre de pixels de l’image. Puis, on va comparer chaque pixel de l’image sélectionnée avec l’image de référence.
L’objectif est ensuite d’obtenir un ratio entre le nombre de pixels que les 2 images ont en commun et le nombre de pixels total.

```
#=============================================================================
# Methode de mesure de similitude entre l'image self et un modele im
#=============================================================================
def similitude(self, im):
    pix_total=0
    pix_similitude=0

    #pas de resize à faire ici
    for i in range (im.H):
        for j in range (im.W):
            pix_total+=1

            if self.pixels[i,j]==im.pixels[i,j]:
                pix_similitude+=1


    ratio=pix_similitude/pix_total
    return ratio
```

La reconnaissance du chiffre :

On parcourt chaque image de la liste des modèles, en redimensionnant l'image test à la taille de l'image de référence, puis on applique la méthode similitude précédente afin d'obtenir un ratio pour chaque image.
L'image de référence qui obtient le ratio le plus élevé correspondra au chiffre détecté.

```
def reconnaissance_chiffre(image, liste_modeles, S):
    #attention a la valeur de S pour une bonne reconnaissance du chiffre
    rapport=0
    count=0
    object=image.binarisation(S)
    for ref in liste_modeles:
        object_resized=object.resize(ref.H,ref.W)
        sim=object_resized.similitude(ref)
        print(sim)

        if rapport<sim:
            rapport=sim
            modele=count

        count+=1

    return modele
```

Test de la fonction avec différentes valeurs de seuil :

avec S=12 : reconnaissance du chiffre 7 avec sim= 0.69
avec S=70 : reconnaissance du chiffre 7 avec sim=0.69
avec S=120 : reconnaissance du chiffre 1 avec sim=0.71
avec S=180 : reconnaissance du chiffre 1 avec sim=0.70
avec S=250 : reconnaissance du chiffre 1 avec sim=0.69

En dessous d'une certaine valeur de seuil, la reconnaissance ne s'effectue pas bien : l'algorithme nous retourne un "7" alors que l'image à tester correspond au chiffre 1.

Conclusion

Ce TP nous aura permis d'effectuer un algorithme de reconnaissance d'images avec le logiciel Python, ainsi que de se familiariser avec de nouvelles fonctions et d'approfondir le côté graphique du codage. Ainsi, à partir d'une image donnée en noir et blanc, nous sommes capables de calculer son taux de ressemblance avec une autre.