

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

Problème posé à la question, solution trouvée et résultats obtenus (illustration de sortie + résultat des tests).

Introduction

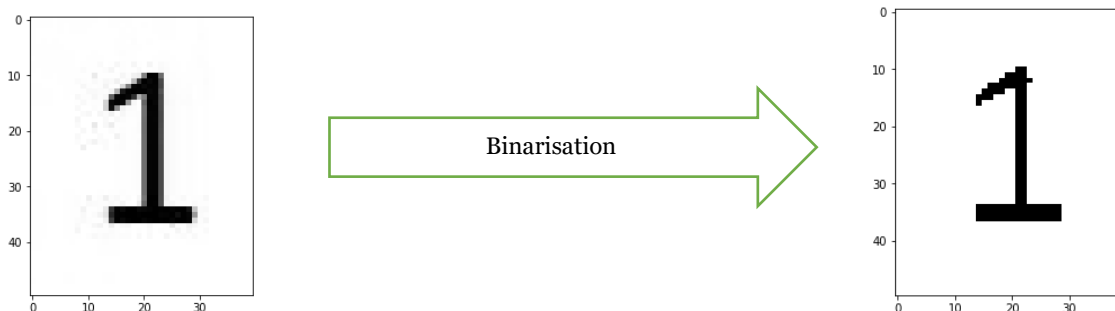
Nous allons vous présenter notre compte rendu de notre TP qui a pour objectif la reconnaissance automatique de caractères. Pour réaliser ceci, nous allons suivre une technique, la reconnaissance par corrélation avec des modèles. Elle se compose des 5 étapes suivantes que l'on vous présentera au cours de ce compte rendu : binarisation, localisation, adaptation de la taille, mesure de ressemblance par corrélation et décision.

I. Binarisation

Pour notre première partie nous devons réaliser une méthode binarisation dans la classe image qui doit recréer une nouvelle image dont chaque valeur de pixels doit être 0 ou 255 selon une valeur seuil passé en paramètre de notre méthode.

```
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))  
  
    #parcourt de chaque pixel de l'image à binariser par deux boucles for (une pour les  
    #lignes, puis une pour les colonnes), puis comparaison avec le seuil et  
    #détermination de la valeur finale du pixel.  
    for i in range(0,self.H,1):  
        for j in range(0,self.W,1):  
            if (self.pixels[i][j]<=S):  
                im_bin.pixels[i][j]=0  
            else:  
                im_bin.pixels[i][j]=255  
  
    #renvoie de l'image binarisée  
    return im_bin
```

A partir de cette méthode nous obtenons comme illustration de sortie :



II. Localisation

Nous avons ensuite réalisé une méthode localisation pour la classe image. Cette méthode nous permet de renvoyer une image recadrée d'une image binarisée en déterminant les coordonnées du rectangle englobant l'ensemble des pixels noir.

```
def localisation(self):
    #Initialisation de deux tableaux : L'un contenant les positions des lignes des
    #pixels noirs et l'autre les positions des colonnes des pixels noirs
    position_pixels_l = []
    position_pixels_c = []

    #Remplissage des tableaux position_pixels
    for l in range(0,self.H,1):
        for c in range(0,self.W,1):
            if (self.pixels[l][c]==0):
                position_pixels_l.append(l)
                position_pixels_c.append(c)

    #Initialisation des variables si l'image ne compte aucun pixel noir
    if (len(position_pixels_l)==0):
        l_min = None
        l_max = None
        c_min = None
        c_max = None

    #Initialisation des variables si l'image compte au moins un pixel noir
    l_min = position_pixels_l[0]
    l_max = position_pixels_l[-1]
    c_min = position_pixels_c[0]
    c_max = position_pixels_c[-1]

    #Détermination de la valeur l_max (l_min est déjà trouvée)
    for i in position_pixels_l:
        if (i>l_max):l_max=i

    #Détermination des valeurs de c_min et c_max
    for j in position_pixels_c:
        if (j<c_min):c_min=j
        if (j>c_max):c_max=j

    # creation d'une image vide et ajout du tableau recadré
    im_loc = Image()
    im_loc.set_pixels(self.pixels[l_min:l_max+1,c_min:c_max+1])

    #Renvoi de l'image recadrée
    return im_loc
```

A partir de cette méthode nous obtenons comme illustration de sortie :

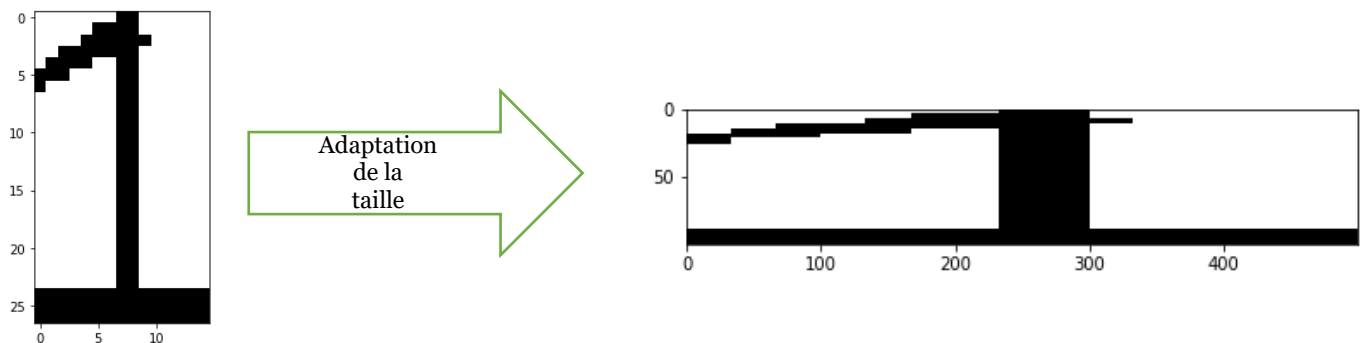


III. Adaptation de la taille

Nous allons désormais réaliser une méthode `resize` de la classe `Image` qui va permettre d'adapter la taille d'une image en modifiant la dimension du tableau correspond à l'image grâce aux nouvelles dimensions mis en paramètre de la méthode.

```
#Redimensionnement du tableau en fonction des dimensions passées en paramètres,  
#passage du tableau de float en int puis ajout du tableau dans l'image  
def resize(self, new_H, new_W):  
    im_res = Image()  
    tab_res = resize(self.pixels, (new_H, new_W), 0)  
    im_res.set_pixels(np.uint8(tab_res*255))  
    return im_res
```

A partir de cette méthode nous obtenons comme illustration de sortie pour un $H = 100$ et un $W = 500$:



IV. Ressemblance par corrélation

Pour cette partie nous avons réalisé une méthode `similitude` de la classe `Image` qui prend en paramètre une autre image qui sera notre image modèle. Cette méthode doit calculer la proportion de pixels de l'image à tester en comparaison avec ceux de l'image modèle et ainsi pourvoir déterminer plus tard à quel caractère correspond cette image.

```
def similitude(self, im):  
    #Initialisation d'un compteur  
    cpt = 0  
  
    #Vérification de l'intensité de chaque pixel de l'image à testée par rapport aux  
    #pixels de l'image modèle à la même position  
    for l in range(0, self.H, 1):  
        for c in range(0, self.W, 1):  
            if (self.pixels[l][c] == im.pixels[l][c]):  
                cpt += 1  
    return cpt / (self.H * self.W)
```

V. Décision

Cette partie correspond au fichier reconnaissance dans lequel nous avons dû réaliser une fonction reconnaissance_chiffre qui a pour paramètre une image à identifier, une liste d'images modèles et un seuil de binarisation. Cette fonction doit pouvoir nous renvoyer le caractère correspondant à notre image. Cette fonction utilise les différentes méthodes de la classe image.

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    #Binarisation de l'image de base  
    image_bin = image.binarisation(S)  
  
    #Recadrement de l'image binarisée  
    image_loc = image_bin.localisation()  
  
    #Initialisation des variables similitude_max et chiffre_final  
    similitude_max = 0  
    chiffre_final = None  
  
    #Redimensionnement de l'image recadrée en fonction de tous les modèles  
    for i in range(0, len(liste_modeles), 1):  
        image_res = image_loc.resize(liste_modeles[i].H, liste_modeles[i].W)  
  
        #Recherche de la similitude la plus élevée parmi tous les modèles  
        if (similitude_max < image_res.similitude(liste_modeles[i])):  
            similitude_max = image_res.similitude(liste_modeles[i])  
            chiffre_final = i  
  
    #Renvoi du chiffre dont la similitude est la plus élevée  
    return chiffre_final
```

Après exécution de cette fonction nous obtenons l'illustration suivante :

Le chiffre reconnu est : 1

VI. Test

1. Test Image

Dans cette partie-là, nous vérifions que les tests sur les méthodes binarisation, localisation, resize et similitude soient tous validés.

```
.....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)
```

```
-----
Ran 22 tests in 0.162s
```

```
OK
```

```
An exception has occurred, use %tb to see the full traceback.
```

```
SystemExit: False
```

On peut observer que nos méthodes répondent aux spécifications demandées.



POLYTECH[®]
ANNECY-CHAMBERY



UNIVERSITÉ
SAVOIE
MONT BLANC

2. Test reconnaissance

Dans cette partie-là, nous vérifions que les tests sur la fonction reconnaissance_chiffre soit validée.

```
....lecture image : ../assets/test2.JPG (50x40)
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)
lecture image : ../assets/test2.JPG (50x40)
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)
lecture image : ../assets/test1.JPG (54x43)
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)
lecture image : ../assets/test3.JPG (43x38)

-----

Ran 25 tests in 0.224s









OK
An exception has occurred, use %tb to see the full traceback.

SystemExit: False
```

On peut observer que la fonction répond aux spécifications demandées.

Conclusion

Tableau des similitudes en fonctions du seuil.

Image\Seuil	10	40	70	100	130	160	190	220	250
	0,9028	0,9132	0,9219	0,9635	0,9201	0,8715	0,8646	0,9167	0,7049
	0,8620	0,8593	0,8854	0,9010	0,9453	0,9375	0,9662	0,8958	0,6875
	0,8547	0,9	0,9062	0,9062	0,9094	0,9031	0,8906	0,8906	0,5977
	0,8187	0,8312	0,8344	0,8219	0,8703	0,8766	0,875	0,8719	0,6027
	0,7906	0,8062	0,7687	0,7734	0,7844	0,7953	0,8078	0,8203	0,6984
	0,7513	0,7318	0,7409	0,7357	0,7201	0,7161	0,72	0,7343	0,6706
	0,6830	0,6845	0,6741	0,6771	0,6696	0,6607	0,6667	0,6637	0,6994
	Indéfini Ne reconnaît aucun pixel noir à ce seuil	0,6302	0,7455	0,7917	0,8036	0,6549	0,4659	0,4659	0,4659

Pour les fichiers test 8 et 9, on n'a pas eu le temps de réaliser une reconnaissance sur un chiffre de plusieurs caractères. Nous avons pu constater que le seuil a une certaine importance sur la précision de la similitude de plus il ne faut pas choisir un seuil ni trop haut ni trop bas de risque d'avoir une similitude trop faible. Selon nos expérimentations, nous pensons que le seuil qui fonctionne le mieux se situe entre 100 et 160, théoriquement nous pensons que le meilleur seuil serait de 128 soit la moitié de 256.

Pour conclure, nous avons réalisé l'ensemble des méthodes et fonctions demandées. Nous avons rencontré certaines difficultés dans la réalisation de la méthode localisation. Grâce à ce TP nous avons pu apprendre et comprendre comment était réalisé une reconnaissance automatique d'un caractère en suivant plusieurs étapes.