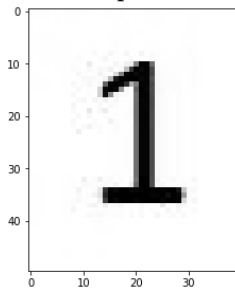


Rapport de TP2 – Lecture automatique de chiffres par analyse d'image

I. Introduction

Dans ce TP, nous nous intéresserons à l'analyse d'une image avec un chiffre pour le retrouver numériquement. Cela se fait en plusieurs étapes que nous développerons tout le long de ce TP.



Nous ferons nos tests avec cette image.

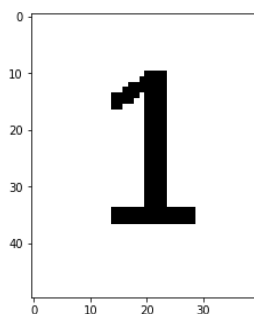
II. Travail préparatoire

1. Question (2)

On s'intéresse à rendre l'image binaire, c'est-à-dire supprimer tous les niveaux de gris et ne garder que du noir (0) sur fond blanc (255).

```
def binarisation(self, S):  
    im_bin = Image()  
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))  
    for i in range(self.H):  
        for j in range(self.W):  
            if self.pixels[i][j] > S:  
                im_bin.pixels[i][j] = 255  
            else :  
                im_bin.pixels[i][j] = 0  
    return im_bin
```

On obtient le résultat suivant :

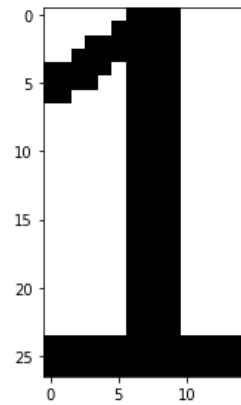


2. Question (3)

Ensuite, nous recadre l'image pour enlever les bordures blanches autour du chiffre. Nous avons quatre fois la même structure pour chaque élément. A chaque fois, on parcourt l'image soit en ligne, soit en colonne. Dès qu'on trouve un pixel noir, on passe à la ligne/colonne suivante. Chaque boucle renvoie donc la dernière itération où il y a eu un pixel noir.

```
def localisation(self):
    for i in range(self.H):
        for j in range(self.W):
            if self.pixels[i][j] == 0:
                l_max = i
                break
    for i in range(self.H-1,-1,-1):
        for j in range(self.W):
            if self.pixels[i][j] == 0:
                l_min = i
                break
    for i in range(self.W):
        for j in range(self.H):
            if self.pixels[j][i] == 0:
                c_max = i
                break
    for i in range(self.W-1,-1,-1):
        for j in range(self.H):
            if self.pixels[j][i] == 0:
                c_min = i
                break

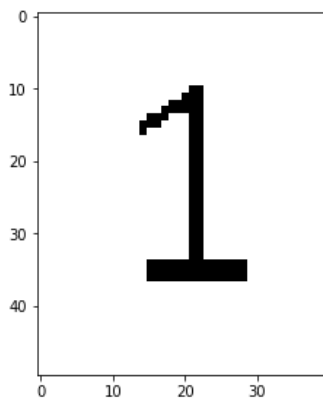
    im_l = Image()
    im_l.set_pixels(np.zeros((self.H,self.W) , dtype=np.uint8))
    im_l.pixels = self.pixels[l_min:l_max+1, c_min : c_max+1]
    return im_l
```



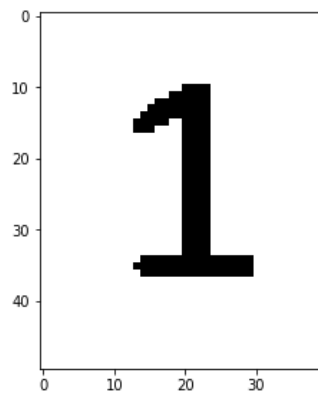
III. Reconnaissance automatique de chiffre

1. Question (1)

Avec un seuil trop bas, nous perdons de l'information car il y a moins de pixels noir. Au contraire, avec un seuil trop haut nous avons trop d'informations.



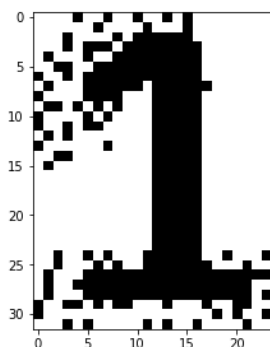
Avec un seuil de 20



Avec un seuil de 230

2. Question (2)

En modifiant le seuil, il se peut qu'il y ait des pixels noirs autour du chiffre. Ainsi le recadrage ne centrera pas le chiffre correctement.



On observe que les pixels parasites empêchent le bon centrage du chiffre. On peut obtenir ce résultat avec un seuil très proche de 255.

3. Question (3)

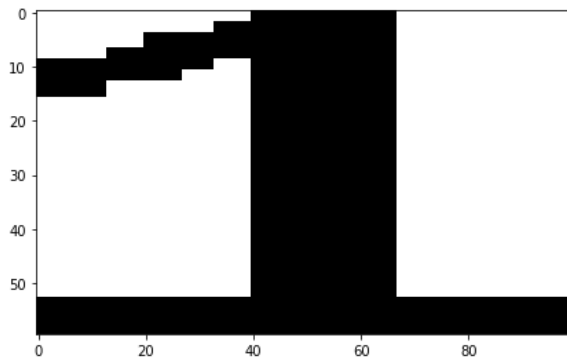


POLYTECH[®]
ANNECY-CHAMBÉRY



UNIVERSITÉ
SAVOIE
MONT BLANC

Pour redimensionner l'image, nous avons utilisé la ligne qui nous est donnée en modifiant bien les valeurs pour qu'elle s'adapte à notre code. De plus, la ligne passe l'image à une échelle de 0 à 1. Il nous a donc fallu de multiplier les pixels par 255 pour revenir sur notre échelle.



En redimensionnant en 60 par 100, on obtient l'image ci-contre.

4. Question (4)

Pour similitude, on compare un à un les pixels de notre image à celle d'une autre image, qui est de même taille. Puis, on renvoie le rapport entre les bons pixels comptés et l'aire de l'image.

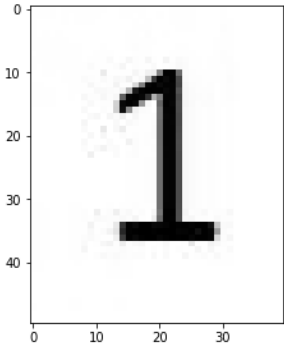
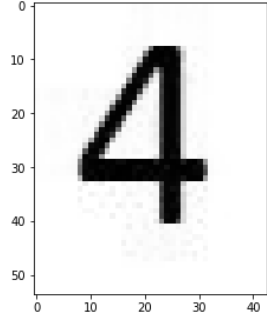
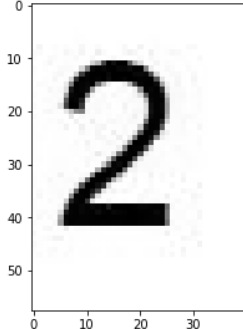
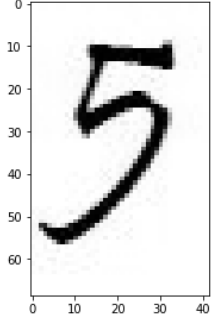
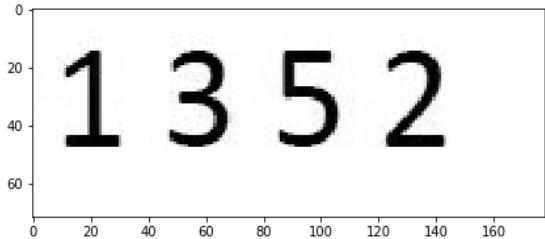
```
def similitude(self, im):  
    nb=0  
    for i in range(self.H):  
        for j in range(self.W):  
            if self.pixels[i][j]==im.pixels[i][j]:  
                nb+=1  
    return (nb/(self.H*self.W))
```

5. Question (5)

Pour faire la reconnaissance, on fait chaque étape qui ont détaillées dans la partie préliminaire pour l'image et pour chaque modèle. Puis, on relève le modèle pour lequel la similitude est la plus élevée.

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    trouve=0  
    seuil=0  
    im=image.binarisation(S).localisation()  
    for i in range(len(liste_modeles)):  
        modele=liste_modeles[i].binarisation(S).localisation()  
        im_r=im.resize(modele.H, modele.W)  
        test_seuil=im_r.similitude(modele)  
        if seuil < test_seuil:  
            trouve=i  
            seuil=test_seuil  
    return trouve
```

6. Question (6)

Image	Résultat
	Le chiffre reconnu est : 1
	Le chiffre reconnu est : 4
	Le chiffre reconnu est : 2
	<p>Avec un seuil à 180 : Le chiffre reconnu est : 5</p> <p>Avec un seuil à 160 : Le chiffre reconnu est : 2</p> <p>Ici, l'écriture de ce 5 est différent que notre modèle. Il est donc « normal » d'avoir un résultat faux.</p>
	<p>Avec un seuil à 160 : Le chiffre reconnu est : 5</p> <p>Avec un seuil à 70 : Le chiffre reconnu est : 6</p> <p>En effet, notre algorithme n'a jamais prévu qu'il puisse y avoir plusieurs chiffres à l'écran.</p>

IV. Conclusion

Dans l'ensemble, le TP s'est bien passé, nous avons des résultats cohérents. Cependant, nous avons rencontré quelques difficultés pour la localisation car nous avons essayé de l'optimiser pour éviter de parcourir toute la matrice.