NOM1 Prénom1
NOM2 Prénom2
Date du TP (jj/mm/aaa)

Rapport de TPX – Titre du TP

I. Introduction

L'objectif de ce TP est d'effectuer une reconnaissance par corrélation avec des modèles.
La reconnaissance s'effectue en 4 étapes :

- Etape 1 : Binarisation
  Il s'agit d'obtenir à partir d'une image grisée, une image seulement composée de noir et de blanc.

- Etape 2 : Localisation
  Cette méthode permet de « rogner » une image uniquement sur l'ensemble de pixel composant l'objet/ le chiffre voulu.

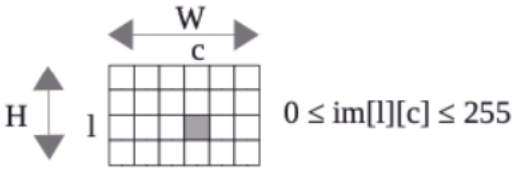
- Etape 3 : Adaptation de la taille
  Comme le nom l'indique, il faut adapter la taille de l'objet analysé à la taille de différents modèles.

- Etape 4 : Mesure de ressemblance par corrélation
  On mesure la proportion de pixel de même intensité et au même endroit entre 2 images.

II. Travail préparatoire

1. Analyse de la classe Image.

On analyse la nature/type de chaque attribut de notre classe Image, le « H » correspond à la hauteur de l'image, le « W » correspond à la largeur de l'image et enfin « pixels » est tableau numpy qui possède les informations relatives à la position des pixels dans l'image choisis ainsi que leur couleur (couleur compris entre 0 et 255).



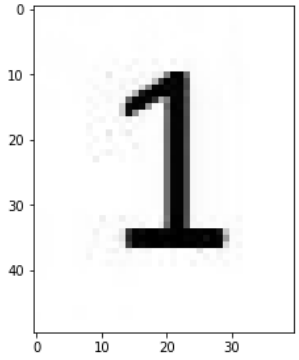
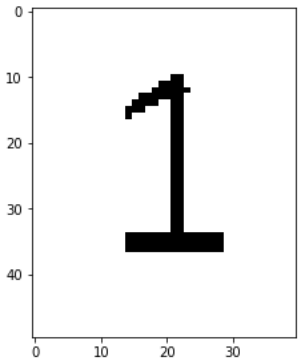
2. Binarisation.

On souhaite créer une nouvelle méthode relative à l'étape 1 (Binarisation).
Tout d'abord la binarisation permet de remplacer tous les pixels autres que noir (0) ou blanc(255) en pixels noir ou blanc. Il faut donc créer une fonction qui puissent analyser chaque pixel individuellement et ensuite comparer chaque pixel pour savoir s'ils sont en dessus ou en dessous du seuil choisis.

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

Initialisation d'une image noire

On obtient 2 images en exécutant le programme principal (main.py) :

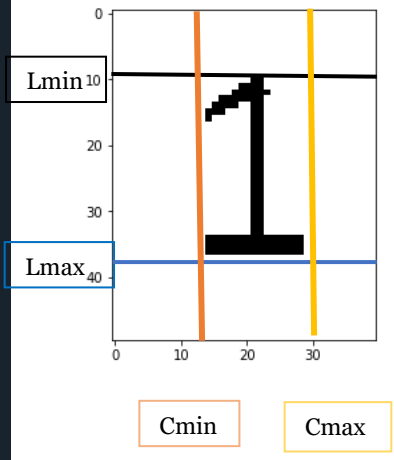
L'image de gauche est l'image original, et l'image de droite est l'image modifié après avoir utilisé la méthode binarisation pour un seuil S = 70.

3. Localisation.

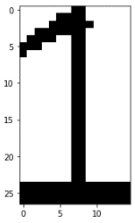
On créer une méthode « localisation » qui permet d'effectuer l'étape 2 (« rogner » limage), pour ce faire il faut aussi analyser l'image comme pour la binarisation, mais cette fois ci il faut la parcourir de chaque côté pour trouver les limites délimitant le chiffre en noir (Lmin,Lmax,Cmin,Cmax).

```
def localisation(self):
    image = Image()
    image.set_pixels(self.pixels)
    lmin = 0
    lmax = 0
    cmin = 0
    cmax = 0
    for x in range(self.H):
        for y in range(self.W):
            if self.pixels[x][y] == 0:
                lmax = x+1
    for y in range(self.W):
        for x in range(self.H):
            if self.pixels[x][y] == 0:
                cmax = y+1
    for x in range(self.H-1,0,-1):
        for y in range(self.W-1,0,-1):
            if self.pixels[x][y] == 0:
                lmin = x
    for y in range(self.W-1,0,-1):
        for x in range(self.H-1,0,-1):
            if self.pixels[x][y] == 0:
                cmin = y

    image.pixels = image.pixels[lmin:lmax,cmin:cmax]
    return image
```



La localisation donne alors à partir de la binarisation cette image :

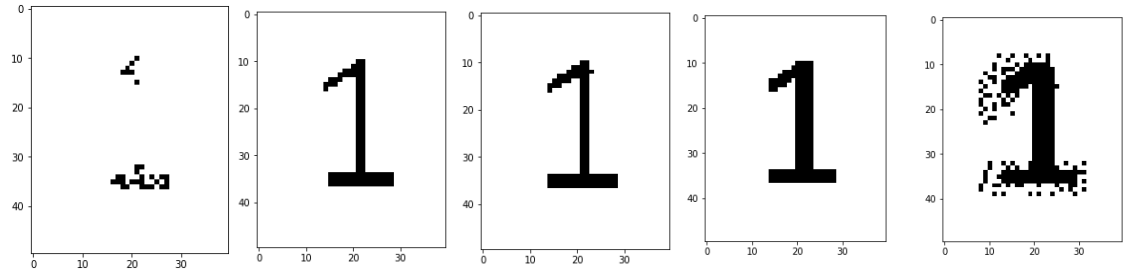


III. Reconnaissance automatique de chiffre

1. Test binarisation avec différent seuil

En testant avec différent seuil l'image binarisée est plus ou moins net par rapport à l'image de départ, si le seuil est trop bas(1) ou trop élevé(250) l'image n'est plus lisible (Voir les images ci-dessous).

Seuil = 1 **Seuil = 20** **Seuil = 70** **Seuil = 150** **Seuil = 250**



2. Test localisation avec différent seuil

Seuil = 1

IV. Conclusion

Expliquer ici l'état d'avancement du TP actuel, les difficultés principales que vous avez rencontrées ainsi que ce que vous avez appris.