Compte rendu

Fonction binaris :

def binaris(self, S):

# preparaton du resultat : creation d'une image vide

im\_modif = image()

# affectation de l'image resultat par un tableau de 0, de meme taille

# que le tableau de pixels de l'image self

# les valeurs sont de type uint8 (8bits non signes)

im\_modif.set\_pixels(np.zeros((self.H,self.W), dtype=np.uint8))

# boucle imbriquees pour parcourir tous les pixels de l'image

for l in range(self.H):

for c in range(self.W):

# modif des pixels d'intensite >= a S

if self.pixels[l][c] >= S:

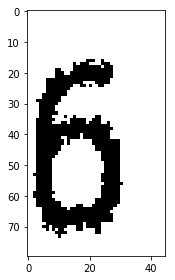
im\_modif.pixels[l][c] = 255

else :

im\_modif.pixels[l][c] = 0

return im\_modif

Image resultant : seuil de 150



seuil = 150

im\_bin = im.binaris(seuil)

im\_bin.display("Image binarisee")

Fonction localisation :

def localisation(self):

#creation de la nouvelle image recadree

im\_loc = image()

# declaration des coordonnees de localisation

l\_min = l\_max = c\_min = c\_max = -10

# determination des coordonnees min/max des lignes

for l in range(self.H) :

for c in range(self.W) :

if self.pixels[l][c] == 0 : # om tombe sur un pixel noir

if(l\_min == -10) : # si la valeur n'a pas deja ete trouve, on note les coords

l\_min = l

l\_max = l # on note les coords max jusqu'a ce qu'on ait plus que des pixels blancs

# determination des coordonnee min/max des colonnes

for c in range(self.W) :

for l in range(self.H) :

if self.pixels[l][c] == 0:

if(c\_min == -10) :

c\_min = c

c\_max = c

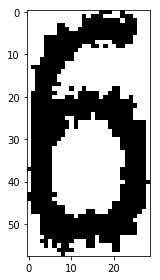
#on affetce les pixels

im\_loc.set\_pixels(self.pixels[l\_min:l\_max+1, c\_min:c\_max+1])

print(l\_min, l\_max, c\_min, c\_max)

return im\_loc

Image resultant :



im\_loc = im\_bin.localisation()

im\_loc.display("Localisation du chiffre")

Fonction redimensionnement de l’image :

def resize\_im(self,new\_H,new\_W) :

# creation de la nouvelle image redimensionnee

im\_resized = image()

# affectation des nouvelles dimensions

im\_resized.H = new\_H

im\_resized.W = new\_W

# redimensionnement de l'image

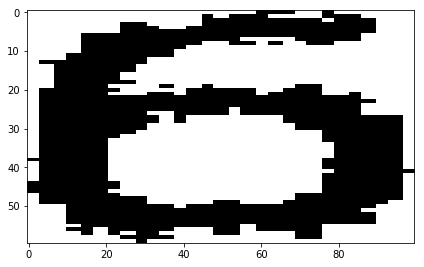
im\_resized.pixels = resize(self.pixels, (new\_H, new\_W), 0)

# normalisation sur l'intervalle de pixels [0, 255] et conversion en uint8

im\_resized.pixels = np.uint8(im\_resized.pixels\*255)

return im\_resized

Image resultant :



new\_H = 60

new\_W = 100

im\_resized = im\_loc.resize\_im(new\_H, new\_W)

im\_resized.display("Image redimensionnee")

Fonction de similitude :

def simil\_im(self, im) :

# Nombre de pixels correspondant entre les deux images

pixels\_simil = 0

#Nombre total de pixels dans l'image de base

px\_total = self.H \* self.W

if self.H==im.H and self.W==im.W :

for l in range(self.H) :

for c in range(self.W) :

if self.pixels[l][c] == im.pixels[l][c] :

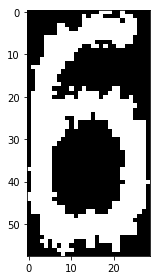
pixels\_simil += 1

else :

print("Dimensions differentes")é

return pixels\_simil / px\_total

Fonction de transformation d’une image en son image negative :

def negative(self) :

#creation de l'image negative

im\_neg = image()

im\_neg.pixels = 255 - self.pixels

return im\_neg

Similitudes résultants : Rapport de similitudes entre l'image localisee et son image negative : 0.0

#creation de l'image negative de im\_loc

im\_loc\_neg = im\_loc.negative()

im\_loc\_neg.display("Image negative")

# test de similitudes

simi = im\_loc.simil\_im(im\_loc\_neg)

print("Rapport de similitudes entre l'image localisee et son image negative :", simi)

Similitude maximum entre l’image de base traitée (binarisation et localisation) et la liste chargée d’images :

list\_modif = [0]\*len(list\_model) #liste des models traités

simil\_max = 0 #similitude maximum

id\_im\_max\_simil = 0 #id de l'image avec la similitude maximum

new\_im = image() #nouvelle image

#Pour chaque model dans la liste, on traite et

#on compare les similitudes avec l'image de base (localisee)

for i in range( len(list\_model) ) :

new\_im = list\_model[i].resize\_im(im\_loc.H, im\_loc.W)

list\_modif[i] = new\_im

s = im\_loc.simil\_im(new\_im)

if s > simil\_max :

simil\_max = s

id\_im\_max\_simil = i

print("L'image la plus similaire est l'image n°", id\_im\_max\_simil, " avec un rapport de " + str(simil\_max))

im\_loc.display("Image de base")

list\_modif[id\_im\_max\_simil].display("Image avec le plus de similitude ")

Resultats : L'image la plus similaire est l'image n° 6 avec un rapport de 0.7538644470868014

