BECHARD Thomas

VEDIE Max

Date du TP (01/12/2021)

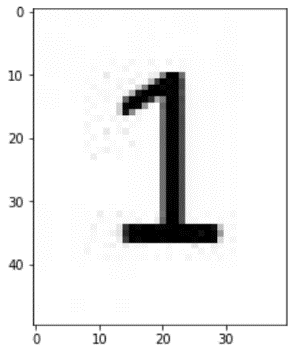
# Rapport de TP2 – Lecture automatique de chiffre par analyse d’image

## Introduction

Le but de ce TP est de reconnaitre des chiffres à l’écran sur un tableau en 2 dimensions grâce à l’aide des pixels et à la reconnaissance des images numériques de Python.

## Prise en main de l’environnement

Quand nous lançons le main du programme, cela nous affiche l’image d’un « 1 ».



## Travail préparatoire

### Analyse de la classe image

Il nous est demander d’abord d’analyser le code nous nous apercevons que les lignes : self.H = 0 et self.W = 0 est le code pour initialiser. Ce sont donc les origines de l’image. Quand elles prendront d’autres valeurs, cela correspondra à la taille de l’image.

De plus, nous considérons les lignes suivantes :

self.pixels = tab\_pixels

self.H, self.W = self.pixels.shape

Ces lignes permettent de créer un tableau et d’intégrer les valeurs H et W, qui indiquent la taille, dedans.

### Méthode binarisation

Nous voulons écrire une méthode qui nous permet de n’avoir que du blanc ou du noir sur notre image, il ne doit plus y avoir de nuances de gris.

Une image contenant texte

Description générée automatiquement

Pour se faire, nous avons ajouté deux boucles for qui s’occupent chacune d’une des valeurs H et W. Dans ces boucles, les valeurs varient et l’algorithme va donc voir s’il faut attribuer la couleur noire ou la couleur blanche grâce à un « if/else ». Si le résultat respecte les conditions alors le programme renverra un pixel de couleur noire égal à 0 sinon il renverra un pixel de couleur blanche égal à 255.

### Méthode localisation

Cette fois, la consigne veut recadrer l’image aux limites du chiffre qui s’affiche. C’est-à-dire que la fenêtre contenant le chiffre aura chacun de ses contours au niveau des premiers pixels noirs.

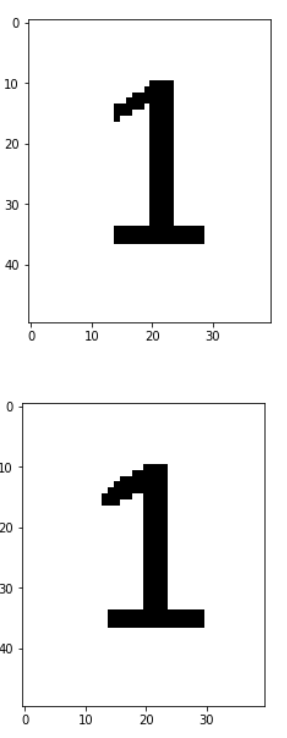
Une image contenant texte

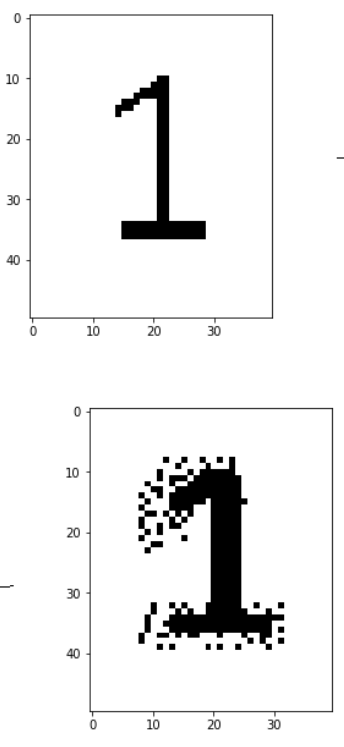
Description générée automatiquement

Dans un premier temps, à l’intérieur de cette méthode, nous initialisons les valeurs des lignes et des colonnes de pixels dans l’image. Ensuite, nous utilisons des boucles for qui tournent jusqu’à ce que les conditions soient respectées. Il y a une boucle pour les colonnes et une pour les lignes. Dans ces boucles, nous avons ajouté des fonctions « if/else » qui permettent de donner aux lignes et aux colonnes leur nouvelle valeur quand un pixel noir est détecté pour l’une d’elle. Une fois que toutes les valeurs sont remplacées avec les valeurs limites, la fonction recadre correctement l’image.

## Reconnaissance automatique de chiffre

### Observation des résultats avec les valeurs de seuils





* 138 🡪 25
* 221 🡪 250

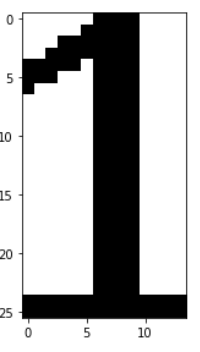
Nous pouvons remarquer qu’a partir d’une certaine valeur, certaines images ont des pixels noirs détachés du groupe principal de pixel qui forme le chiffre. La qualité de l’image dépend donc de la valeur de seuil. Plus la valeur est élevée, plus il y a de pixels.

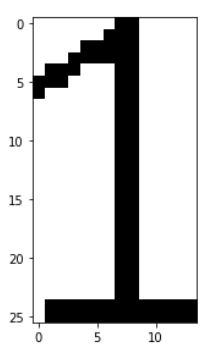
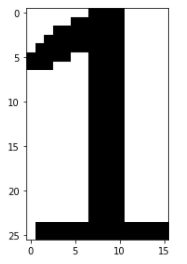
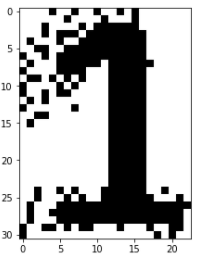
Une image contenant texte

Description générée automatiquement

Quand on lance le programme, il n’y a pas d’erreur.

### Observation de la méthode localisation





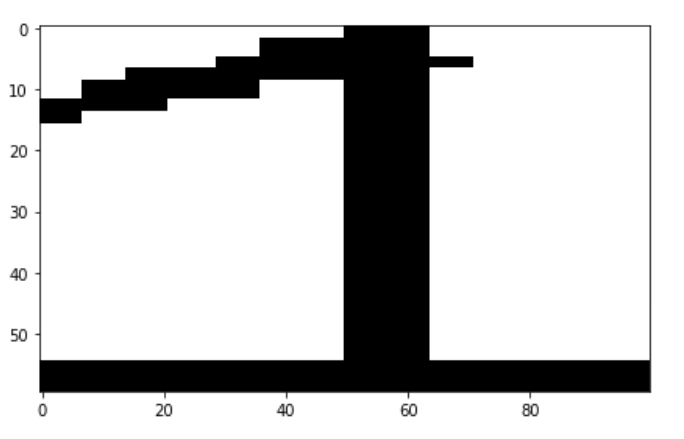
138 25 221 250

Une image contenant texte, table

Description générée automatiquement

Nous observons que la localisation fonctionne correctement mais que les valeurs de seuil trop élevées entrainent aussi un problème pour la localisation car les pixels responsables de la mauvaise qualité, qui sont détachés du groupe principal, restreignent la localisation. C’est-à-dire que la localisation n’enlève pas les pixels inutiles.

Quand on lance le programme, il n’y a pas d’erreur.



### La fonction resize

Une image contenant texte

Description générée automatiquement

La fonction resize, comme nous pouvons en voir le résultat sur cette image, permet de redimensionner le cadre dans lequel est contenu l’image du chiffre. Nous prenons d’abords la nouvelle image que nous mettons égal à l’image d’origine. Ensuite, on utilise la fonction resize dans laquelle nous rentrons les valeurs que nous voulons pour le redimensionnent. Enfin, nous combinons la nouvelle image avec le set\_pixels pour convertir l’image sous forme int au cas où elle serait passée en float lors du redimensionnement. Le test avec test\_image ne contient pas d’erreur.

### Méthode similitude

Une image contenant texte

Description générée automatiquement

Dans cette fonction, nous initialisons un compte à 0. Puis nous utilisons encore deux boucles for pour les lignes et les colonnes. Le compte augmente de 1 à chaque boucle ce qui permet de tester chaque pixel. Enfin, une fonction if permet de compter chaque fois qu’un pixel est similaire pour à la fin faire le rapport entre le compte et les similitudes.

### Fonction reconnaissance\_chiffre

Une image contenant texte

Description générée automatiquement

Nous faisons d’abord une binarisation et d’une localisation, avec une initialisation des chiffres et de la similitude comme vu précédemment. Une boucle for est alors lancée, elle va contenir une liste de modèle qui va être comparée à notre image initiale. Pour pouvoir avoir la reconnaissance, les fonctions resize et similitude sont utilisées. Enfin, la fonction if permet de confirmer le résultat avec des conditions.

### 

En modifiant les données nous obtenons les résultats ci-dessus ce qui montre que notre fonction test\_reconnaissance marche bel et bien sauf pour le chiffre 5 qu’elle détecte plus comme étant un 7. Le seuil le plus adapté reste toujours celui initial à savoir 70.

## Conclusion

Au travers de ce TP, nous avons pu comprendre et aborder la notion de reconnaissance de Python sur des images. Nous avons également pu avancer dans notre apprentissage des classes et des méthodes en langage Python.