LEON Alice

DUNIAU Timée

Date du TP (03/12/2021)

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

## Introduction

Durant ce TP nous réaliseront une étude sur des images de nombres. Premièrement nous allons les mettre sous forme binaire puis nous réaliseront des modifications sur elles dans le but de pouvoir ensuite les comparer. La comparaison va nous servir à réaliser une fonction pour pouvoir reconnaitre la valeur des nombres sur différentes images. Ensuite nous mettrons cette fonction en marche afin de vérifier sa fiabilité.

## Section 3 du TP

### Question (1).

Dans la classe image les trois variable H, w et pixels définisse la représentation de l’image.

* H est responsable de la hauteur
* W est responsable de la largeur

En initialisation notre tableau est vide, il sera rempli par la suite

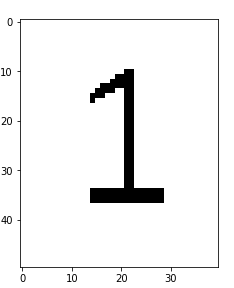
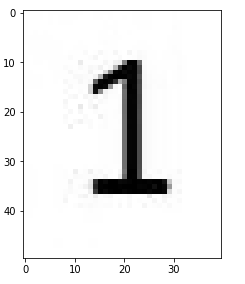
### Question (2).

On crée une méthode binarisation(self, S) qui nous permettra de passer d’une image codée sur 256 valeurs à une image avec seulement deux valeurs (0 ou 255). Pour cela on réalise une boucle **for** qui passera par toutes les valeurs du tableau. Pour chaque valeur on vérifiera à l’aide d’un **if** si la valeur est supérieure ou égale à S, si supérieur on remplace la valeur par 255, si inférieur, par 0.

Une image contenant texte

Description générée automatiquement

On test alors le code de la fonction a l’aide du fichier main.py. L’image obtenue est alors bien nette, la binarisation a fonctionné :



=> Binarisation =>

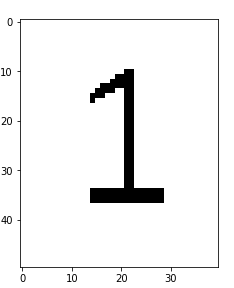
### Question (3).

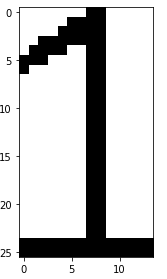
On réalise maintenant la fonction localisation(self) qui va sélectionner dans l’image uniquement le rectangle dans lequel se trouve le nombre. Pour cela on fait comme précédemment on réalise une boucle qui va nous permettre de tester toutes les valeurs. On crée 4 constantes l\_min, l\_max, c\_min et c\_max. Puis on test avec un **if** chaque valeur afin de trouver les extremums de la hauteur et de la largeur. Ensuite on utilise la fonction set\_pixel afin de réduire le tableau au format du rectangle. Le code est le suivant :

Une image contenant texte

Description générée automatiquement

On test alors le code de la fonction à l’aide du fichier main.py. L’image obtenue est la suivante, une réduction de la table à bien été réalisé :



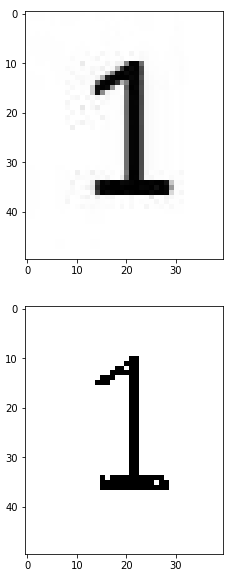
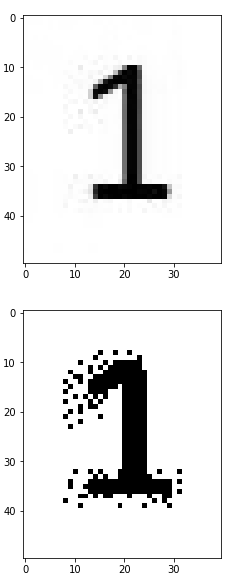


=> Localisation =>

## Section 4 du TP

### Question (1).

Dans cette question on fait varier la valeur de S dans la fonction binarisation.

Pour S = 10 Pour S =250

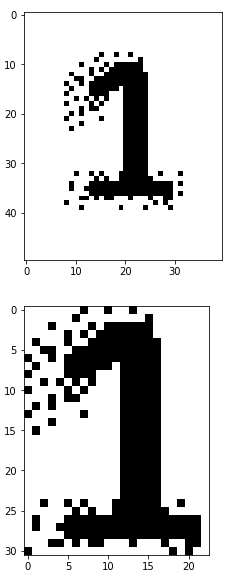
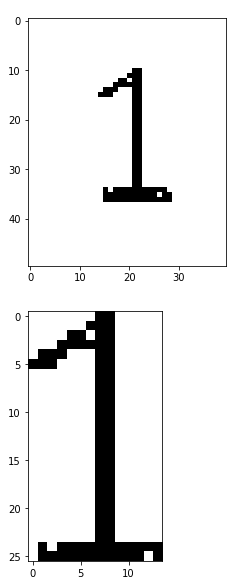
On peut constater qu’il existe une valeur idéale de S afin de conserver uniquement ce qui nous intéresse de l’image. Si le seuil est trop élevé alors on se retrouve avec trop d’informations (S=250), a l’inverse si trop bas il manque des informations (S=10).

Pour réaliser le reste de la question il y a une erreur, il faudrait que l’on importe le module nose 2.

### Question (2).

Dans cette question on utilise la fonction localisation avec différents seuils. On prendra comme tout précédemment :

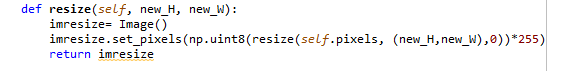
S=10 S=250



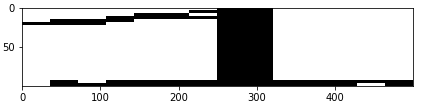
Le fait d’avoir plus d’informations agrandi le rectangle concerné par l’affichage du nombre avec un S important, et le reduit avec une petite valeur de S, car moins d’information sont prise en compte.

### Question (3).

Dans cette question nous allons utiliser la fonction resize de la librairie skimage afin de modifier les constantes H et w. Ainsi nous pourrons modifier la hauteur et la largeur de l’image. Le code est le suivant :



On peut constater que en modifiant la valeur de H en 100 et W en 500 on se retrouve avec un ensemble de définition de largeur 500 unités et de hauteur 100 unités.



### Question (4).

Dans cette question nous allons crée une fonction similitude(self, image) dans la classe Image. Cette fonction va comparer différentes images entre elles et donner un pourcentage de ressemblance. On utilise des boucles for pour comparer chaque valeur du tableau. Dans cette boucle on comptabilisera le nombre d’élément identique et différent à l’aide d’un if. A la fin on réalise le calcul suivant : (nb identique/ (nb identique + nb différent)). Le code est le suivant :

Une image contenant texte

Description générée automatiquement

### Question (5).

Dans cette question nous allons réaliser une fonction reconnaissance\_chiffre(image, liste\_modeles, S) qui permettra de reconnaitre le nombre présent sur l’image. Pour cela on utilise une banque d’images de nombres et la fonction similitude(self, im). Ainsi on peut creer une boucle for qui va comparer notre image a toutes les autres afin de garder celle ayant le plus grand pourcentage de similitudes. Avant cela on réalisera un travail sur l’image afin qu’elle soit comparable entre elles, en les mettant de même dimension, localisation et en binaire. Le code est le suivant :

Une image contenant texte

Description générée automatiquement

### Question (6).

On va maintenant exploiter notre fonction afin de voir ces capacités et ces limites. Pour cela on va comparer différents et voir les résultats.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **comparaison** | **Seuil** | **nombre réel** | **indice** | **fonctionnement** |
| Test 2 | 10 | 1 | 1 | oui |
| Test 2 | 100 | 1 | 1 | oui |
| Test 2 | 250 | 1 | 1 | oui |
| Test 4 | 50 | 2 | 2 | oui |
| Test 4 | 250 | 2 | 2 | oui |
| Test 7 | 50 | 5 | 7 | non |
| Test 7 | 250 | 5 | 7 | non |
| Test 1 | 70 | 4 | 4 | oui |
| Test 8 | 70 | 1352 | 7 | non |

## Conclusion

Après avoir binarisé les images nous avons réalisé un traitement afin de pouvoir les comparer à l’aide de notre fonction similitude. Ensuite cette fonction nous a servi à réaliser une méthode permettant de reconnaitre la valeur d’un nombre avec une image de celui-ci. Notre programme reste limité. Il commet des erreurs sur des chiffres simples, et il est incapable de comparer des nombres a plusieurs caractères entre eux.