MONNIER Marine

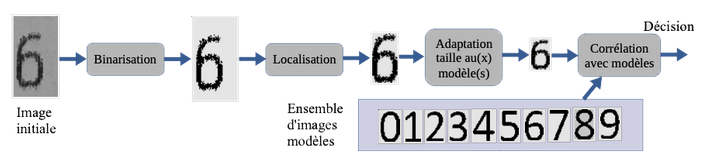
SANGOUARD Marine

30/11/2021

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

## Introduction

Lors de ce TP, nous allons programmer un système de reconnaissance automatique de caractères, en particulier de chiffres. La reconnaissance se fera par corrélation avec des modèles selon le principe ci-dessous :



## Travail préparatoire

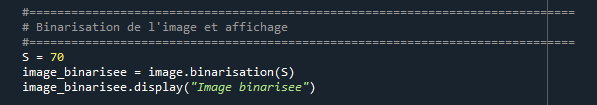
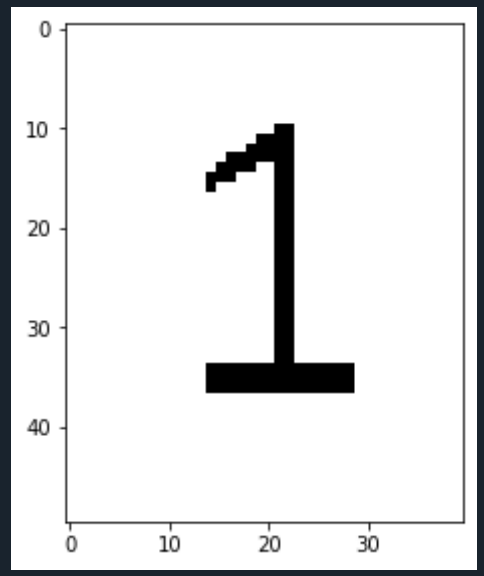
### Analyser attentivement la classe image :

H (pour Height) et W (pour Width) représentent le nombre de pixels de l’image en hauteur et en largeur. Ces valeurs définissent aussi la taille du tableau 2D “pixels”.

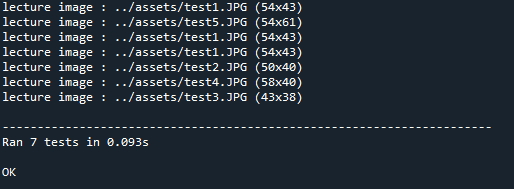
### Écrire une méthode ****binarisation(self, S) :****

On écrit deux boucles imbriquées pour accéder à chaque pixel 1 à 1 dans le tableau 2D. Pour chaque pixel, si sa valeur est supérieure ou égale au seuil, on lui attribue (dans une nouvelle image “im\_bin”) la couleur blanche (255), si non la couleur noire (0).

Résultat des 7 tests concernant « binarisation » :



### Écrire une méthode ****localisation(self) :****

Dans la fonction de localisation, on cherche les bornes de l’image les plus petites possibles : c’est à dire les lignes et colonne min et max contenant au moins 1 pixel noir.

On crée une nouvelle image im\_loc.

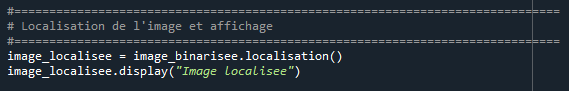
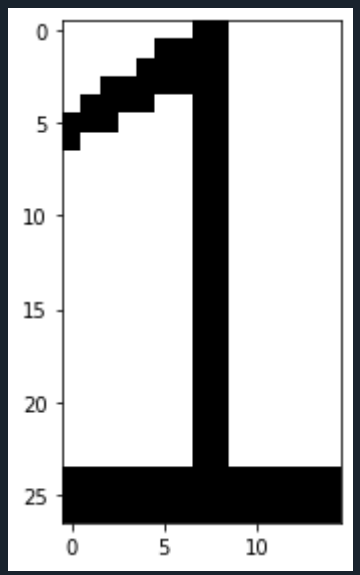
On initialise les lignes et colonnes max au minimum (i.e. à 0), et les lignes et colonnes min aux maximum (i.e. à la taille de l’image : self.H-1 et self.W-1).

On fait une double boucle pour parcourir tous les pixels.

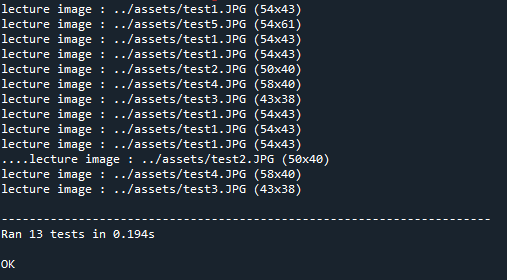
Pour chaque pixel noir trouvé, on regarde ses coordonnées.

On met à jour les bornes : I\_min devient l si l’on trouve un l si l'indice de la ligne est inférieur au l\_min. On procède de la même façon avec c\_min; puis avec l\_max et c\_max (si l’indice est supérieur à la borne actuelle).

On associe à la nouvelle image les pixels de l’image de départ, limités aux nouvelles bornes.

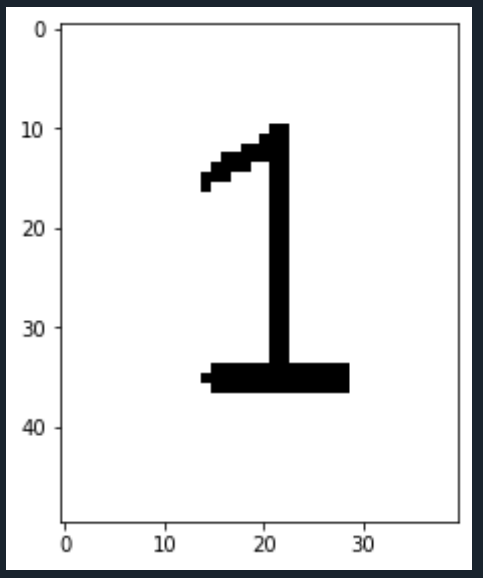
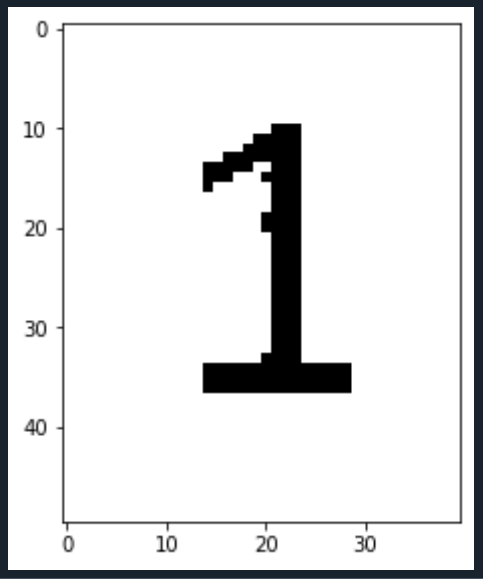
Résultat des 13 tests concernant « binarisation » et « localisation » :



## Reconnaissance automatique de chiffre

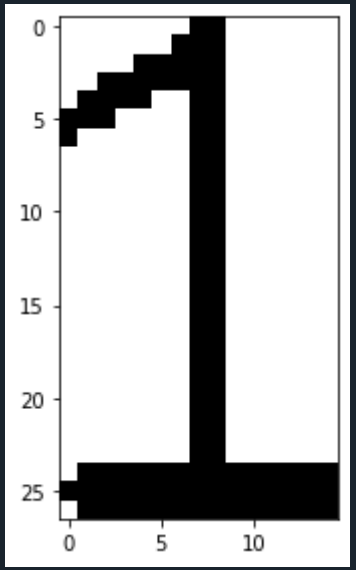
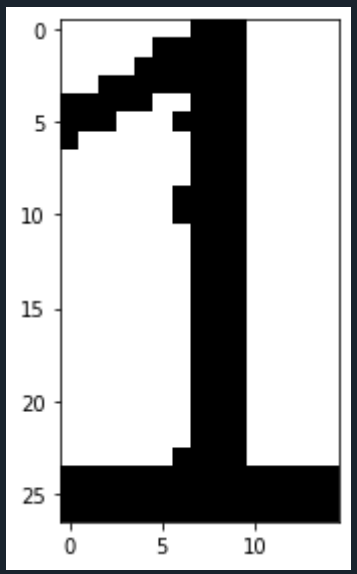
### Essayer avec différentes valeurs de seuil pour la méthode de binarisation et observez le résultat :

Avec un seuil à 50, on obtient l’image suivante : Avec un seuil à 110, on obtient l’image suivante :

### Essayer avec différentes valeurs de seuil pour la méthode de localisation et observez le résultat ****:****

Avec un seuil à 50, on obtient l’image suivante : Avec un seuil à 110, on obtient l’image suivante :

### Ajouter à la classe ****Image**** la méthode resize(self,new\_H,new\_W)****:****

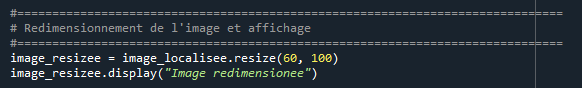
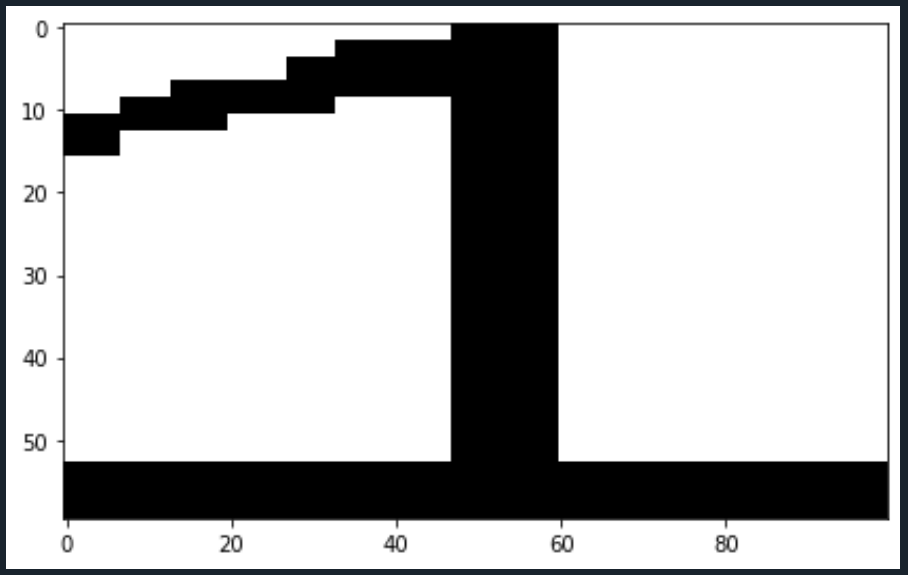
 

On crée une nouvelle image im\_resized à laquelle on veut imposer des dimensions (plus grandes ou plus petites que l’image d’origine).

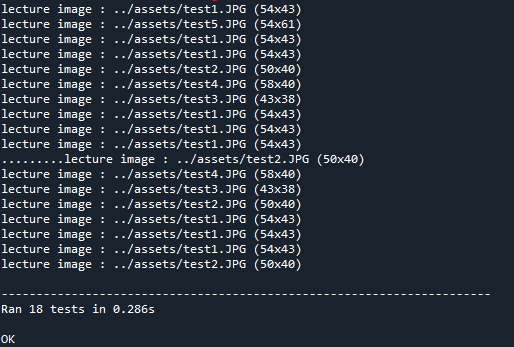
les pixels de im\_resized prennent les valeurs retournées par la fonction resize de la bibliothèque skimage, prenant pour paramètres une image binaire et les dimensions souhaitées.

L'échelle de l'image pouvant prendre des valeurs de type float, on convertit les valeurs en int.

Remarque : On remarque que si l’on resize l’image à une taille vraiment très petite (10,10), des pixels gris apparaissent, donc l’image n’est pas binarisée.

Résultat des 18 tests :



### Ajouter à la classe ****Image****, la méthode ****similitude(self, image)**** :

On cherche la proportion de similitude : (nombre de pixels identiques) / (nombre de pixels total)

Pour cela, les deux images doivent avoir la même taille. C’est pourquoi on a fait la méthode « resize » précédente : on donne à l’image la même taille que celle du modèle à comparer.

On commence par créer un compteur pour compter les pixels identiques, et on les initialise à 0.

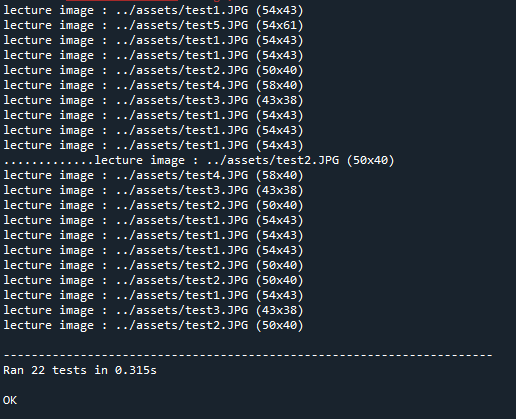
On parcourt tous les pixels de l’image dans une double boucle, et on les compare à ceux du modèle.

Lorsque 2 pixels de mêmes indices sont identiques, on incrémente le compteur.

On calcule la proportion en divisant la valeur du compteur par la taille de l’image (hauteur\*largeur)

Plus la proportion est proche de 1, plus les 2 images sont similaires.

Résultat des 22 tests :



### Écrire la fonction ****reconnaissance\_chiffre(image, liste\_modeles, S) :****

On prend une image. On la binarise, on la localise.

On crée une liste dans laquelle on memorisera la ressemblance à chaque modele.

Pour chaque modèle,

* on « resize » l’image localisée à la taille du modele
* on calcule la similitude avec le modèle
* on memorise cette similitude dans la liste

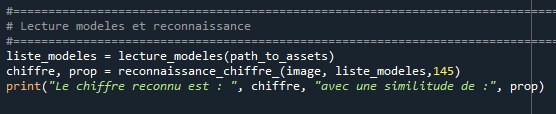
Ensuite, on veut trouver la plus grande similitude dans la liste.

On intialise deux variables max\_similitude et indice\_max\_similitude à 0.

On parcourt la liste, si la valeur de similitude est plus grande que le maximum courant, on attribue cette valeur au maximum et on mémorise son indice i.

On retourne l’indice du maximum.

Remarque : par la suite, nous avons choisi de modifier cette fonction afin de retourner également la valeur de la similitude maximale.





Résultat des 3 tests de « reconnaissance » :



### Essayer la fonction de reconnaissance en modifiant l’image de test dans ****main.py**** avec différentes images disponibles dans ****assets/**** et en modifiant également le seuil avec quelques valeurs et proposer une valeur de seuil qui marche le mieux selon vos expérimentations :

N.B. Nous réalisons l’affichage des images à une taille fixe de 40x40, mais le calcul de similitude se fait bien en adaptant la taille à celle du modèle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Seuil | Image test 2 | Image test 5 | Image test 10 | Image test 7 |
| 70 | Le chiffre reconnu est : 1 avec une similitude de : 0.9184027777777778 | Le chiffre reconnu est : 2 avec une similitude de : 0.76875 | Le chiffre reconnu est : 0 avec une similitude de : 0.6306818181818182 | Le chiffre reconnu est : 9 avec une similitude de : 0.08096590909090909 |
| 145 | Le chiffre reconnu est : 1 avec une similitude de : 0.921875 | Le chiffre reconnu est : 2 avec une similitude de : 0.79375 | Le chiffre reconnu est : 6 avec une similitude de : 0.5684523809523809 | Le chiffre reconnu est : 5 avec une similitude de : 0.0953125 |
| 240 | Le chiffre reconnu est : 1 avec une similitude de : 0.8506944444444444 | Le chiffre reconnu est : 2 avec une similitude de : 0.8125 | Le chiffre reconnu est : 8 avec une similitude de : 0.4659090909090909 | Le chiffre reconnu est : 7 avec une similitude de : 0.5372023809523809 |

Nous avons modifié légèrement notre programme et particulièrement la fonction de reconnaissance afin de retourner la proportion de similitude en plus du chiffre reconnu.

D’après nos tests, nous pouvons considérer que le seuil idéal se situe autour de 145. Un seuil trop faible ou trop grand peut engendrer des erreurs de reconnaissance.

## Conclusion

Nous avons réussi à comparer deux images et calculer leur niveau ressemblance. Cela nous permet de comparer une image à plusieurs modèles pour identifier le chiffre inscrit sur l’image. Cela fonctionne dans des cas simples (chiffre écrit « correctement » et noir sur blanc).

En effet, il y a des limites au programme, par exemple on ne pourrait pas traiter une image avec plusieurs chiffres ou identifier un chiffre blanc écrit sur fond noir (ou foncé).

On a pu observer dans le cas du numéro 6 écrit sur fond gris qu’un seuil élevé empêche complètement l’identification d’un chiffre.

La reconnaissance dépend aussi des modèles disponibles, notamment de leur police d’écriture (c’est ce qu’on a pu voir avec le numéro 5 écrit en « biais » qui est reconnu comme un 7 ou un 9).

Notre programme a passé tous les tests ☺ !

