Dubois Lucas
Laroudie Vincent
01/12/2021

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

## I/ Introduction

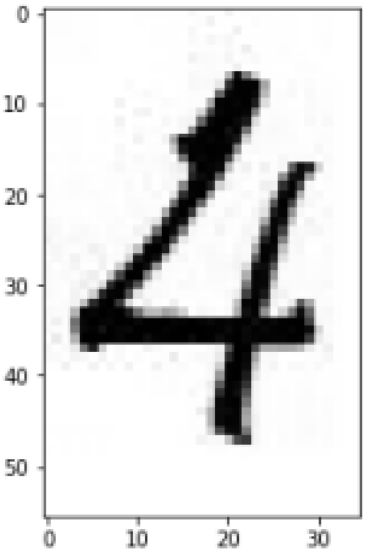
L'objectif de ce TP est d'apprendre à manipuler les images sur Python et plus particulièrement de pouvoir les comparer entre elle.

## III – Travail préparatoire

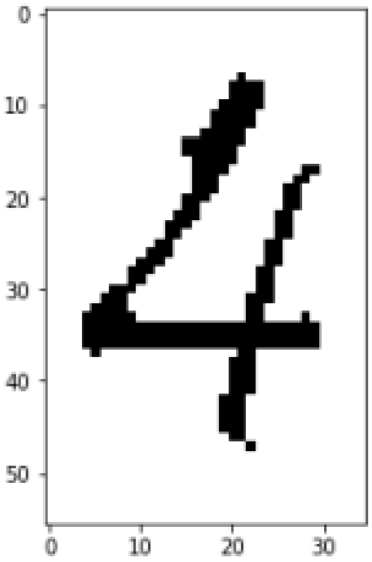
### 2)

Voici une capture d'écran des résultats obtenus :

Avant :



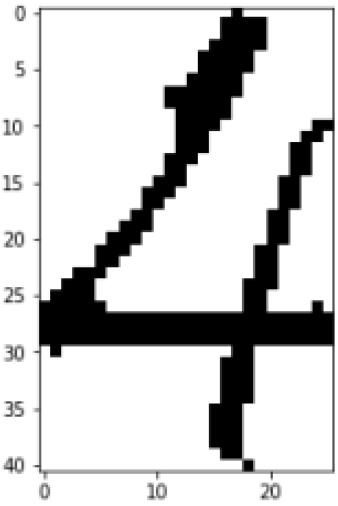

Après :




La méthode binarisation a pour attribut une image avec des pixels ayant des valeurs comprises entre 0 et 255, ainsi qu'un seuil de binarisation.
La méthode permet de renvoyer cette image modifiée avec des pixels ayant désormais pour valeur 0 ou 255. Le seuil de binarisation impose la limite où chaque pixel ayant une valeur inférieure aura comme nouvelle valeur 0, et 255 pour les pixels ayant des valeurs supérieures au seuil. Cela nous permet d'avoir une image juste en noir et blanc, comme nous pouvons le voir à travers les images ce dessus.

**3)**

Test de notre fonction :



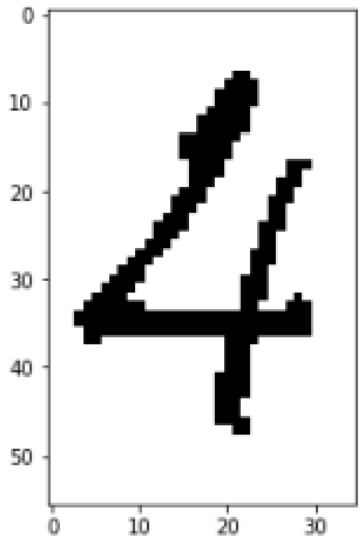

Cette méthode permet donc de « zoomer » sur l'image. En effet, en faisant cela, on évite ainsi d'avoir de grande zone blanche pas forcément intéressante. Elle permet aussi de recentrer le chiffre ce qui est intéressant pour l'analyser. Pour cela, nous avons utilisé 2 boucles for pour parcourir la matrice et ainsi trouver les extrémités du chiffre (l'extrémité est la première case noire que le code va trouver).

## IV – Reconnaissance automatique de chiffre

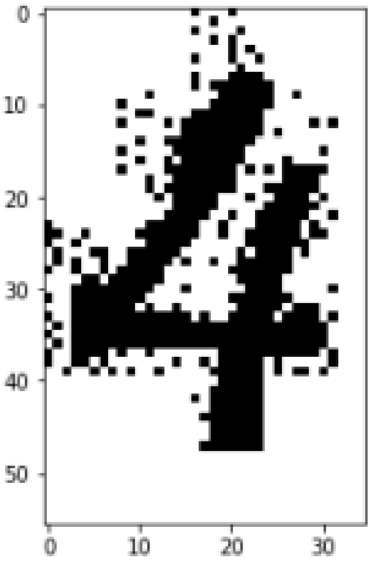
**1)**

Le seuil de base est de 70, nous avons testé plusieurs autres seuils, en voici 2 exemples :

Test seuil 127 :



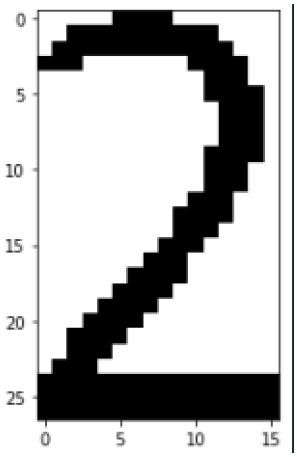
Test seuil 250 :




D'après nos tests de seuils réalisés, le seuil 70 de base est bien adapté.

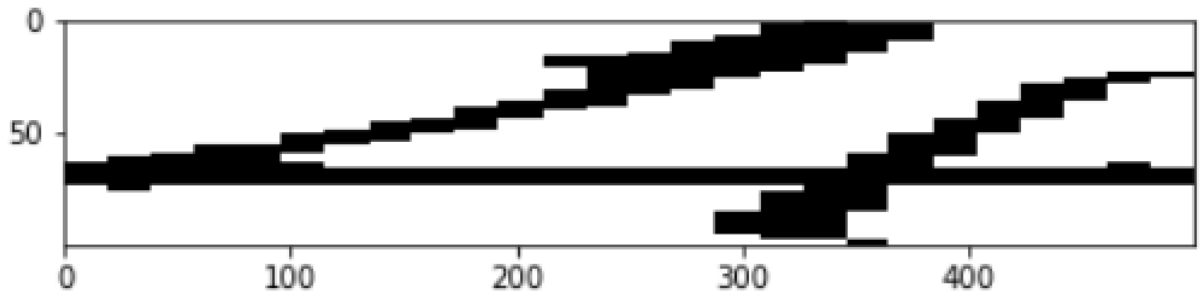
**2)**

Voici 1 autre exemple de la méthode localisé en plus de l'exemple que nous avons mis au travail préparatoire :




**3)**

Voici un exemple de la méthode resize :



Pour changer la taille d'une image on utilise simplement la fonction resize déjà présente afin de modifier la taille de l'image (hauteur, largeur) avec les données entrées par l'utilisateur.

**4)**

Cette méthode va comparer 2 images et renvoyer un chiffre compris entre 0 et 1. Plus le chiffre se rapproche de 1, plus les 2 images ont des pixels en commun (2 pixels sont dits « en commun » s'ils sont tous les 2 noires ou tous les 2 blancs).

Affichage de test_image :

```
----------------------------------------------------------------------
Ran 22 tests in 0.241s

OK

In [87]:
```

Il y a donc visiblement aucun problème.

**5)**

La fonction reconnaissance_chiffre permet comme son nom l'indique de reconnaitre le chiffre qui est sur une image donnée. Pour cela, elle va récupérer l'image test et la liste d'images modèles. Ensuite, on va localiser l'image test (pour la rendre plus nette), puis dans une boucle for (pour parcourir chacune des images exemples). On va récupérer leurs hauteurs ainsi que leurs longueurs afin de resize l'image test.
Finalement, à l'aide de la méthode similitude, on va chercher quelle image exemple est la plus similaire à notre image test.
A l'aide d'un compteur la méthode retournera le chiffre qui est sensé être afficher sur l'image test.

Affichage de test_reconnaissance :

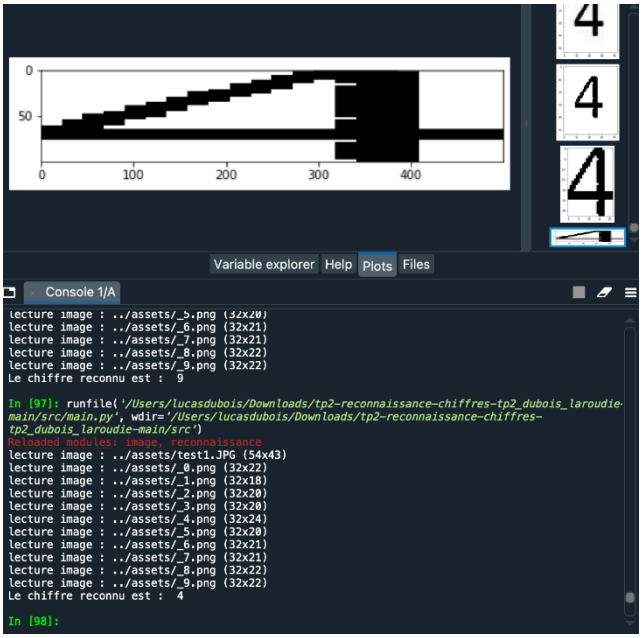
```
----------------------------------------------------------------------
Ran 3 tests in 0.144s

OK

In [88]:
```

Il n'y a donc visiblement aucun problème.

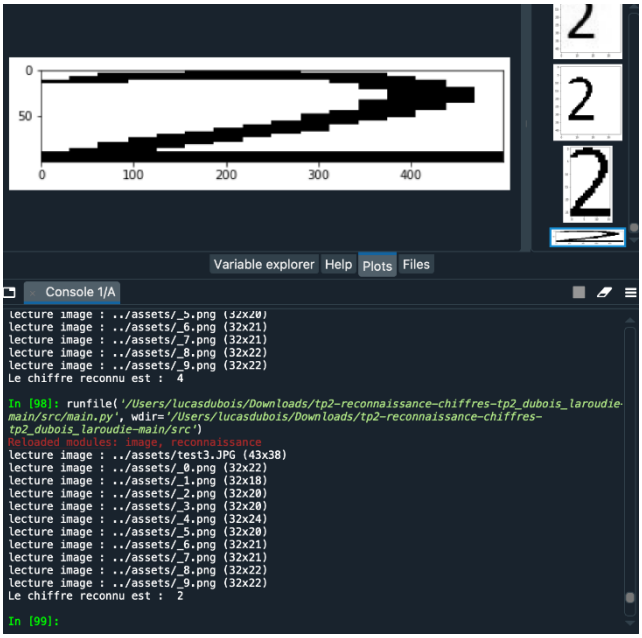
**6)**

Voici différents exemples de la fonction reconnaissance :

Test1.jpg :



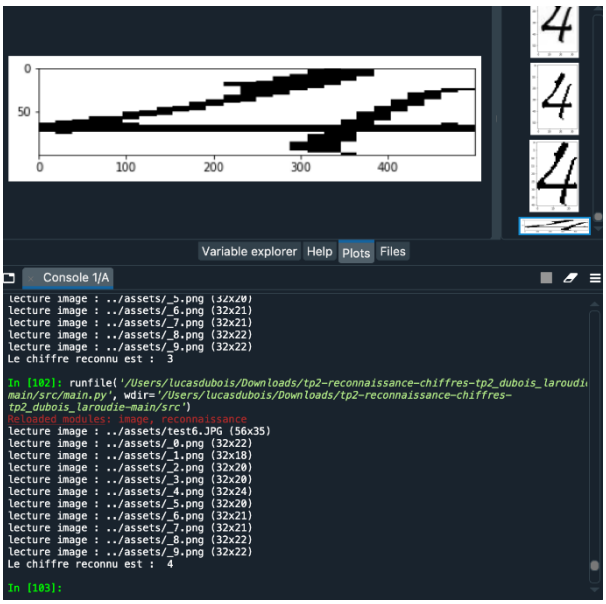

Test3.jpg :



Test7.jpg :


Nous pouvons constater que pour le test7.jpg, le programme nous donne que le chiffre reconnu est le chiffre 9. Cependant le chiffre affiché est 5. En effet, c'est logique car la construction de ce 5 est plus proche du 9 que du 5 exemple.
Cet exemple montre donc que notre programme possède des limites.

Test6.jp6 :



Test10.jpg :


On voit donc qu'en moyenne tout fonctionne bien, nous avons utilisé le seuil d'origine qui était de 70, de plus en faisant varier le seuil à 127 qui est la moitié du maximum, nous n'avons pas vu de changement remarquable (le programme nous renvoie les même chiffres).

## V – Conclusion

En conclusion, nous avons trouvé ce tp très intéressant pour plusieurs raisons. En effet, il nous a permis de découvrir les notions de modification et de comparaison d'images sur Python. Il était aussi intéressant de jouer sur les données de modification afin de voir les différents seuils avant qu'une image change entièrement de couleur.
A première vue ce TP aurait pu paraitre compliqué mais il s'est finalement avéré d'une difficulté correcte, notamment après avoir bien compris le fonctionnement des méthodes vu dans le travail préparatoire.