SUCCHE Valentin

JENNY Camille

Date du TP (30/11/2021)

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

## Introduction

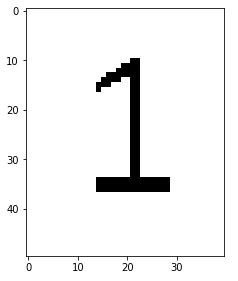
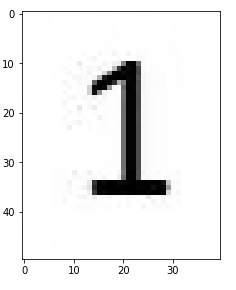
Durant ce TP, nous allons programmer nous familiariser avec l’analyse d’image. Pour cela, nous allons analyser des images de chiffres et les binariser, avant de les recadrer et de vérifier leur ressemblance au modèle par corrélation.

## Travail préparatoire

### Question (1).

La fonction binarisation permet de passer une image en niveau de gris à sa représentation en pixels noirs ou blanc. Pour cela, un utilise un seuil. Si la valeur d’un pixel dépasse le seuil, elle prendra la valeur 255 (blanc), sinon, elle prend la valeur 0 (noir). Pour vérifier chaque pixels on fait une boucle pour chaque ligne et une boucle pour chaque colonne.

On crée un tableau numpy vide dans lequel on va importer ces valeurs, pour ne pas modifier l’image originale.



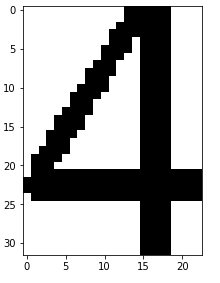
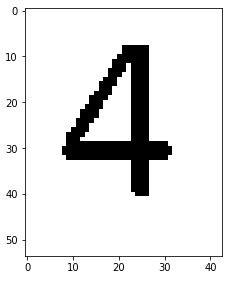
Ici, on peut voir à gauche l’image à binariser et à droite le résultat obtenu. En effectuant un test avec test\_Image.py, on obtient seulement des messages d’erreur reliés à des fonctions qui ne seront réalisées que dans les questions suivantes.

### Question (2).

La fonction localisation a pour but à pour but de délimiter les contours du chiffre binarisé pour ensuite recadrer l’image selon ces contours.

Pour cela, on réalise une boucle qui fixe une valeur L maximum et minimum pour la première ligne où l’on trouve un pixel noir et une valeur C maximum et minimum pour les premières colonnes où l’on trouve un pixel noir.

On fixe la valeur de Lmin et Cmin au maximum et les valeurs de Lmax et Cmax au minimum pour ensuite les réajuster en vérifiant la valeur de chaque pixel. Les resultats sont stockés dans un tableau numpy dont la taille est fixée par Lmin, Lmax, Cmin et Cmax, et qui sera ensuite affiché.

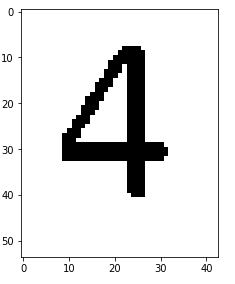
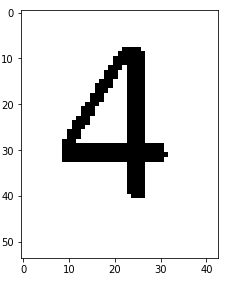
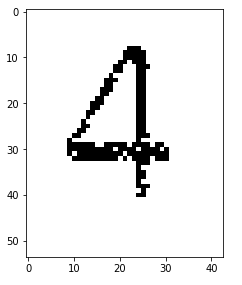


A gauche nous pouvons voir l’image binarisée, et à droite cette même image une fois localisée.

## Section 2 du TP

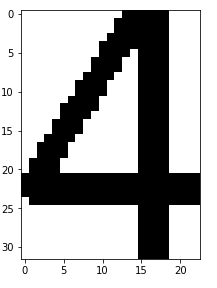
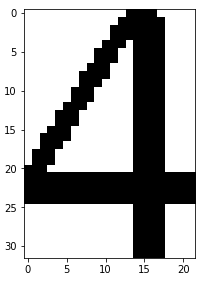
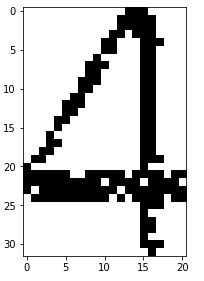
### Question (1).

On teste le résultat de la binarisation avec trois seuils différents, pour chercher à l’optimiser.



Ici, on a le résultat de la binarisation pour un seuil respectif de 10, 80 et 170. On peut voir qu’avec un seuil trop faible, l’image possède de moins en moins de pixels noir, et il peut être compliqué d’ensuite étudier la similitude avec une autre image. Quand la valeur du seuil augmente, on peut voir les traits du chiffre s’épaissir pour devenir de moins en moins précis. La valeur de seuil la plus optimale serait aux alentours de 100.

### Question (2).

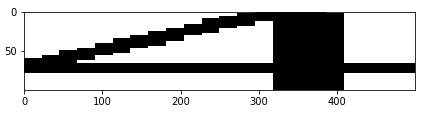


Ensuite, en effectuant le test avec les mêmes valeurs de seuil mais cette fois avec la localisation, on obtient les images ci-dessus. On peut voir que la localisation coupe quelques pixels quand leur quantité est trop faible.

### Question (3).

Ensuite, on a programmé la fonction resize permettant de choisir les dimensions d’une image. Cette fonction est utile pour pouvoir analyser la similitude entre deux images qui auront la même taille. Les nouvelles valeurs H et W correspondant aux nombre de ligne et de colonnes seront fixées en entrée.

Pour cela on crée une nouvelle image qui prend comme valeur les pixels de l’image précédentes pour le nombre de colonne et de ligne définit.



Ici, on peut voir le résultat si on prend H = 100 et W = 500 avec l’image de 4 précédente.

### Question (4)

Une fois la fonction resize définie, on peut définir la fonction similitude, qui analyse la correspondance entre chaque pixel de deux images pour retourner la proportion de similitude entre les deux.

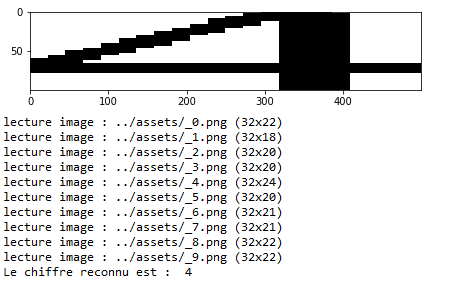
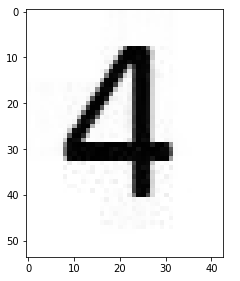
Pour cela on réalise une boucle pour chaque ligne et colonne correspondant à un pixel. On initialise un compteur qu’on incrémente lorsqu’un des pixels a exactement la même valeur pour les deux images. Pour obtenir la proportion, on divise la valeur du compteur par le nombre de pixels.

### Question (5)

Une fois toutes ces fonctions définie, on réalise la fonction reconnaissance\_chiffre. Cette fonction prend en entrée une image pour évaluer sa similitude avec les modèles de chiffres de 0 à 9. Ensuite, elle retournera l’indice du chiffre correspondant le plus à celui présent sur l’image.

Pour cela, il faut binariser, localiser, puis redimensionner l’image de base à la taille de l’image de chaque modèle. On réalise une boucle, qui, pour chaque modèle, testera la similitude avec l’image et retournera la valeur de l’indice du chiffre si celui-ci a une similitude plus élevée que le chiffre précédent.

Ainsi, on obtient le résultat suivant par exemple à partir de l’image de gauche ci dessous :



Le nombre reconnu est bien 4.

### Question (6)

|  |  |  |  |
| --- | --- | --- | --- |
| Valeur de seuil\Image |  |  |  |
| 30 | Le chiffre reconnu est : 1 | Le chiffre reconnu est : 4 | Le chiffre reconnu est : 6 |
| 100 | Le chiffre reconnu est : 1 | Le chiffre reconnu est : 4 | Le chiffre reconnu est : 6 |
| 200 | Le chiffre reconnu est : 1 | Le chiffre reconnu est : 4 | Le chiffre reconnu est : 6 |

Pour chacun des seuils ci-dessus, le résultat obtenu est toujours correct, mais on remarque qu’un seuil aux alentours de 100 est le plus optimal. En effet, pour des images comme celle du 6, où le contraste est faible, mettre un seuil trop élevé rend l’image complètement noire et un seuil trop faible la rendra trop blanche.

Dans le cas du 4, un seuil inadapté pourra diminuer la précision sur les contours du chiffre, ce qui pourrai fausser le résultat.

## Conclusion

En conclusion, ce TP nous a permis de réaliser un programme de reconnaissance de chiffre à partir d’une image en passant par différentes étapes telles que la binarisation, la localisation, le redimensionnement et l’analyse de similitudes.

En lançant les tests sur l’ensemble du programme, on obtient aucune erreur.