

## Rapport de TP2 – lecture automatique de chiffre par analyse d'image

### I. Introduction

L'analyse automatique d'image (OCR pour Optical Character Recognition) est employée dans de nombreux domaines. L'objectif de ce TP est d'illustrer une technique d'OCR relativement simple : la reconnaissance par corrélation avec des modèles.

### II. Travail Préparatoire

#### 1. Question (1).

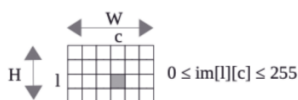


Figure 1 : image en niveau de gris

Analyser attentivement la classe `image` et remarquer que les attributs **H**, **W** qui indiquent la taille de l'image et l'attribut **pixels** qui contient un tableau 2D numpy contenant les valeurs de l'image en pratique.

→ Les attributs **H**, **W** représentent respectivement la hauteur (Height) et la largeur (Width). L'attribut **pixels** est une liste en 2D qui représente la teinte des pixels.

#### 2. Question (2).

Écrire une méthode **binarisation(self, S)** à la classe `**image**` qui permet de passer d'une image codée sur 256 valeurs à une image avec seulement deux valeurs (0 ou 255). Cela se fait en comparant pour chaque pixel de l'image la valeur du pixel à une valeur choisie par l'utilisateur (entrée **S** de la méthode). Ainsi en pratique, il faut itérer sur tous les pixels du tableau numpy et comparer la valeur au seuil. On veut également que le résultat soit donné sous la forme d'une nouvelle image que l'on crée dans la fonction afin de ne pas modifier l'image de base.

- On crée une nouvelle image dans laquelle tous les pixels sont de valeur 0
- Dans deux boucles for imbriquées, pour chaque pixel de l'image originale, on compare sa valeur à un seuil
- Selon si la valeur est inférieure ou supérieure ou égale au seuil, on donne à chaque pixel de la nouvelle image la valeur 0 ou 255.

Seuil = 10, il manque des morceaux de l'image	Seuil = 70	Seuil = 200, apparition de bruits

#### 3. Question (3)

Écrire une méthode **localisation(self)** à la classe `Image` calculant et retournant l'image recadrée sur le chiffre à identifier. Le principe de ce recadrage consiste, sur une image binaire **im\_bin** de dimension  $H \times W$ , à déterminer les coordonnées  $l_{min}$ ,  $l_{max}$  et  $c_{min}$  et  $c_{max}$  du rectangle englobant la forme noire (valeurs 0) et à construire l'image limitée à ce rectangle englobant.

- Pour chaque pixel de l'image, on vérifie que chaque valeur en X et Y soit supérieure ou inférieure au maximum ou maximum précédent, afin de trouver les valeurs limites de l'image pour la recadrer.

### III. Reconnaissance automatique de chiffre

#### 1. Question (1)

Essayer de lancer le fichier **main.py** et de voir le résultat de la méthode de binarisation. Essayer avec différentes valeurs de seuil et observez le résultat.

- On observe que le « bruit » (pixels gris) n'est plus présent sur l'image passée en niveau de gris, et que les bordures du chiffre, plus claires, apparaissent toujours. Le seuil est correctement calibré.

Lancer également le fichier de tests **test\_Image.py** pour voir si votre méthode répond bien aux spécifications demandées (regarder seulement le résultat des tests en rapport avec la binarisation). Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

#### 2. Question (2)

Essayer de lancer le fichier **main.py** et de voir le résultat de la méthode de localisation. Essayer avec différentes valeurs de seuil et observez le résultat.

- Selon la valeur du seuil, les bordures du chiffre disparaissent ou sont conservées. On observe donc une variation de la taille du chiffre en fonction du seuil.

Lancer également le fichier de tests **test\_Image.py** pour voir si votre méthode répond bien aux spécifications demandées. (regarder seulement le résultat des tests en rapport avec la localisation). Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

#### 3. Question (3)

Ajouter à la classe **Image** la méthode `**resize(self, new_H, new_W)**` qui redimensionne l'image à la taille voulue et renvoie un autre objet de type **Image** en sortie. Tester cette méthode dans le fichier **main.py** sur l'image obtenue par l'étape de localisation. Pour ce test, on choisira des dimensions quelconques, (60,100) par exemple.

- On crée une nouvelle image vide
- On appelle la fonction `resize(60, 100)` qui renvoie un nouveau tableau de pixels correspondant à l'image redimensionnée, et on copie ces valeurs dans la nouvelle image.

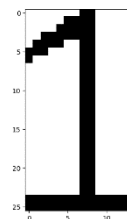


Figure 2 : Image redimensionnée

Lancer également le fichier de tests **test\_Image.py** pour voir si votre méthode répond bien aux spécifications demandées. (regarder seulement le résultat des tests en rapport avec la localisation). Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

#### 4. Question (4)

Ajouter à la classe **Image**, la méthode **similitude(self, image)** qui mesure la similitude par corrélation d'images entre l'image représenté par l'objet courant (**self**) et un objet de type **Image** entrée en paramètre. La procédure pour le calcul de similitude est décrite dans la présentation du TP.

- Pour chaque pixel de l'image redimensionnée, on vérifie si sa valeur est égale à celle de l'image argument de la méthode **similitude**. On compte le nombre de similitudes.
- La méthode renvoie la proportion de similitudes par rapport au nombre total de pixels.

#### 5. Question (5)

Dans le fichier **reconnaissance.py**, écrire la fonction **reconnaissance\_chiffre(image, liste\_modeles, S)** qui va effectuer la reconnaissance de chiffre sur l'image **image** donnée en entrée de la fonction. Pour cela il faudra dans la fonction tout à tour, binariser l'image et la localiser. Ensuite, il faut calculer sa similitude à tous les modèles (en redimensionnant l'image à la taille du modèle) et trouver le modèle pour lequel il y a la plus grande similitude. Cela

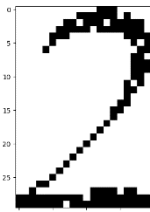
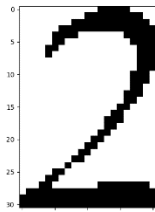
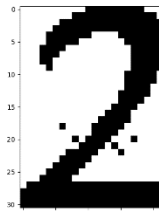
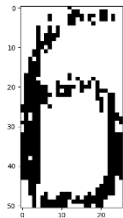
peut être fait en parcourant une liste d'images modèles et en sauvegardant la similitude maximale ainsi que l'indice de l'image avec la similitude maximale. La fonction doit renvoyer un entier compris entre 0 et 9.

- ➔ On crée un objet *best* dont les propriétés sont la similitude et l'index de l'image, avec comme conditions initiales -1 et « none »
- ➔ On crée une boucle qui itère sur tous les éléments de la liste l'ensemble des opérations précédentes afin de renvoyer le taux de similitudes entre les images.
- ➔ On vérifie si le taux de similitude avec chaque référence est supérieure à celle en propriété de l'objet *best*. Dans ce cas on associe aux propriétés similitude et index le taux de similitude et le numéro d'itération.
- ➔ Une fois que l'image d'entrée est comparée à toutes les images de la liste, on retourne le taux de similitude et l'index de l'image *best*.

Lancer également le fichier de tests **test\_reconnaissance.py** pour voir si votre méthode répond bien aux spécifications demandées. Corriger au besoin en analysant les tests qui ne réussissent pas ou pour lequel il y a des erreurs.

## 6. Question (6)

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 (pas besoin d'en faire 100...). Présenter les résultats dans le rapport sous forme de tableau (différentes images en ligne et différents seuil en colonne). Proposer une valeur de seuil qui marche le mieux selon vos expérimentations.

Seuil Valeur d'entrée	10	70	240
« test5.jpg » : 2	 2	 2	 2
« test10.jpg » : 6	Image blanche : crash du programme	 6	Image noire : 8

#### IV. Conclusion

Ce programme permet de reconnaître les chiffres dans les images :

- Il transforme les images en niveau de gris en images en noir et blanc
- Il recadre l'image autour du chiffre
- Il redimensionne l'image à la même taille que celle des images références
- Il vérifie quelle image parmi les modèles à le taux de pixels similaires le plus élevé

Le seuil, qui est un paramètre choisi arbitrairement, utilisé dans la méthode **binarisation** a un impact important sur le résultat du programme. De plus, ce programme a des limites : il ne peut pas reconnaître des images tournées ou contenant plusieurs chiffres.

On note que si l'image est blanche, le programme plante, mais que si l'image est noire il l'assimile à un 8 : en effet, si le programme ne trouve pas de pixel noir il ne peut pas recadrer correctement l'image, créant une image de dimensions 0, 0 que le programme ne sait pas correctement interpréter, et l'image référence du 8 est celle contenant le plus de pixels noirs.