**TP sur le traitement d’images :**

**Reconnaissance de chiffres (uniquement de 0 à 9)**



Données fournies : une base de données composé d’images de chiffre (stockée dans ‘images’), une base de données de chiffres inconnues (stocké dans ‘images\_inconnues’), les dossiers ‘images’, ‘images\_inconnues’, ‘images\_traitees’, ‘images\_inconnues\_traitees’

Les bases de données sont faites maison !!!

**Partie 1 : Traiter les images**

Etape 1:

*Explication :*

Faire la matrice de l’image.

Il va falloir faire des calculs sur des matrices d’image. Chaque membre d’une matrice correspond à un seul nombre, or chaque pixel possède trois valeurs (LED). Ainsi, on va créer une matrice de même taille que l’image dont les membres auront la valeur des niveaux de gris pour chaque pixel (on additionne les valeurs du rouge, vert et bleu et on divise par trois). Puis on divise cette valeur par 255(ainsi un 1 dans la matrice correspond à 255 dans l’image, donc du blanc, et un 0 correspond à du noir). Ensuite, on applique une convolution sur cette matrice pour en délimiter les contours et ainsi repérer le chiffre. Pour cela, on va utiliser la matrice de Sobel. Une fois cela fait, on fait un seuillage de l’image et si la valeur est inférieure à ce seuil, on la met à 1.0, et si elle est supérieure, on la met à 0 (les contours sont plus foncés suite à la convolution de Sobel). Enfin, on stocke ces valeurs dans une nouvelle matrice et on a enfin la matrice prête pour en extraire les données de reconnaissance de chiffre.

*Instructions :*

Utiliser les modules ‘numpy’, ‘PIL.Image’, ‘os’ (pour gérer les dossiers), ‘csv’ (pour les fichiers CSV, oui il y en aura !) et ‘math’

1. Codez une fonction qui parcourt l’image et associe pour chaque pixel son niveau de gris. Elle doit diviser cette valeur par 255 puis la stocker dans une matrice. (Aide : créez d’abord une matrice de 0 de mêmes dimensions que l’image puis remplacer les 0 par la valeur obtenue). La fonction prend en paramètre l’image et doit renvoyer la matrice obtenue.
2. Codez une fonction qui applique la convolution de Sobel. Elle prend en paramètre une matrice (celle obtenue avec la fonction dans le 1) ). La convolution doit s’appliquer pour chaque membre de la matrice. Pour ceux qui ne connaissent pas la convolution : <https://youtu.be/5CaotCBDs6c> (le motif est la matrice de Sobel, la grande matrice est la matrice de l’image). Il faut en réalité appliquer deux convolutions, une en axe des abscisses et une en axe des ordonnées. Voici les matrices de Sobel : [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]] pour les abscisses et [[-1, -2, -1], [0, 0, 0], [1, 2, 1]] pour les ordonnées. Une fois les deux valeurs calculées, on calcule la distance entre les deux (théorème de Pythagore) => √(val1 \*\* 2 + val2 \*\*2). Enfin, on multiplie cette valeur par un coefficient : 255 / (√2\*1020) pour la ramener dans une valeur comprise entre 0 et 1. Enfin, on stocke cette valeur dans une nouvelle matrice (ou dans la même) à la même place et la fonction doit renvoyer cette matrice convolée.
3. Codez une fonction qui va faire le seuillage. On prendra comme valeur 0.16 (valeur arbitraire). La fonction doit parcourir la matrice. Si la valeur qu’elle rencontre est inférieure à 0.16, cette valeur passe à 1.0. Si elle est supérieure, elle passe à 0.0. La fonction prend en paramètre une matrice (convolée dans le 2) ) et doit renvoyer la matrice seuillée.

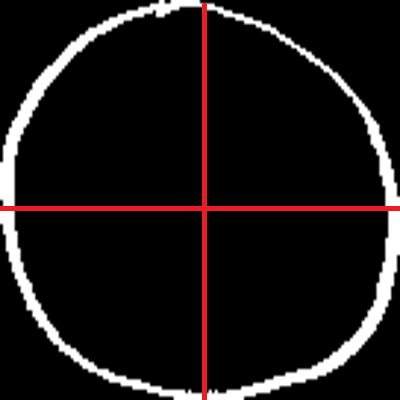
On a enfin la matrice de l’image prête pour les calculs.

Etape 2 :

*Explications :*

Il faut ensuite centrer l’image, puis on va définir une ligne verticale qui coupe l’image en deux dans le sens de la hauteur, puis une autre ligne horizontale qui coupe l’image en deux dans le sens de la largeur. On va ensuite compter le nombre de fois que le chiffre coupe la ligne verticale puis la ligne horizontale (l’histoire des lignes c’est pour la compréhension, on va juste parcourir la matrice). Intérêt : le chiffre est coupé une fois en hauteur et largeur : c’est un 1 …, deux fois en hauteur et largeur : c’est un 0…

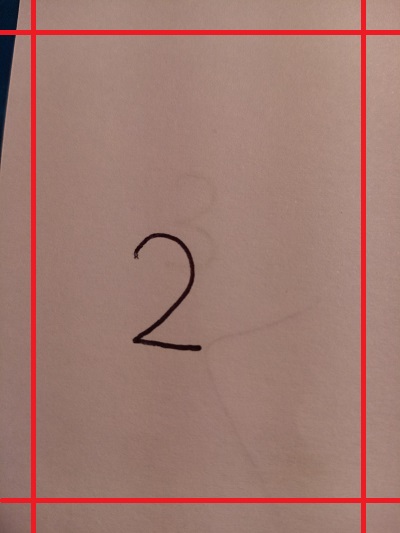
Cela fait comme un repère où l’on place le chiffre.

 Le 0 coupe deux fois la ligne verticale et deux fois la ligne horizontale.

Rappel : suite à la convolution, le chiffre dans l’image est constitué de 1, et le reste de l’image est constitué de 0.

*Instructions :*

1. Codez une fonction qui prend en paramètre une matrice et un paramètre de coupe. Tout d’abord, la fonction doit couper une partie des bords, en fonction du paramètre de coupe.



Partie à rogner, par exemple on rogne à 40 pixels à partir des bords.

Ensuite, elle doit renvoyer une matrice centrée sur le chiffre, c'est à dire que la fonction va couper les bords de l'image qui ne contiennent pas le chiffre. Aide : Les bords sont constitués uniquement de 0, donc on coupe les parties de la matrice composées uniquement de 0. Pour cela on parcourt la matrice en longueur (colonne) et en largeur (ligne), on repère le premier 1 rencontré, puis on repère le dernier 1 et on coupe après. On fait cela sur la longueur et la largeur (il faudra parcourir la matrice dans plusieurs sens).

1. Codez une fonction qui compte combien de fois le chiffre (constitué de 1) coupe la ligne verticale du milieu, puis une fonction similaire qui compte combien de fois le chiffre coupe la ligne horizontale du milieu. Aide : il faut compter le nombre de 1 dans la colonne du milieu et dans la ligne du milieu. Ces fonctions prennent en paramètre une matrice et renvoie un nombre.

Les données peuvent être extraites des images !

Etape 3 :

*Explications et instructions :*

Appliquer ces fonctions sur les images dans le dossier ‘images’, puis ‘images\_inconnues’. ATTENTION : redimensionner les images en 400\*400 (sinon les calculs mettent trop de temps). Ensuite, pour chaque image, après avoir calculé la matrice, l’avoir convolée et centrée (ATTENTION les images de ‘images’ seront rognées à 40 pixels et les images de ‘images\_inconnues’ seront rognées à 20 pixels), la convertir en image, la redimensionner encore en 400\*400, l’enregistrer dans le répertoire ‘images\_traitees’ pour les images de ‘images’ et ‘images\_traitees\_inconnues’ pour les images de ‘images\_inconnues’. En déduire une matrice (encore une fois !) de cette image traitée (pour avoir des matrices de même taille, ici, simplement prendre la première valeur de la LED pour chaque pixel (puisqu’ils ont les même), la diviser par 255, la placer dans une matrice de même taille que l’image). Appliquer les fonctions de lignes verticale et horizontale et stocker les données obtenues dans un fichier CSV ainsi que le type pour les images de ‘images’ (bref dire que la 1ère image est un 0, la deuxième est un 0, …) et appliquer aussi les fonctions de lignes verticale et horizontale et stocker uniquement ces données dans un autre fichier CSV pour les images de ‘images\_inconnues’.

Infos : Il y a 5 ‘0’, 8 ‘1’, 8 ‘2’, 5 ‘3’, 5 ‘4’, 9 ‘5’, 4 ‘6’, 4 ‘7’, 4 ‘8’, 6 ‘9’ dans ‘images’.

**Partie 2 : Analyser les paramètres pour reconnaitre le chiffre**

*Explications :*

Dans une nouvelle page de code, on va comparer les paramètres des fichiers CSV et on va comparer les images traitées pour reconnaitre la nature des chiffres de ‘images\_inconnues’. Pour cela, après avoir récupérer les images de ‘images\_traitees’ et ‘images\_inconnues\_traitees’ et les paramètres des deux fichiers CSV (l’un des chiffres connus et l’autre des chiffres inconnus), on recrée des matrices à partir des images (simplement prendre la première valeur de la LED pour chaque pixel (puisqu’ils ont les même), la diviser par 255, la placer dans une matrice de même taille que l’image). On regarde combien de fois les images inconnues coupent les lignes verticales et horizontales et on regarde quelles images ont les mêmes paramètres : cela orientera la deuxième partie de la reconnaissance : la superposition. S’il n’y a qu’une seule nature de chiffre qui présente les même paramètres (par exemple le 0 qui coupe deux fois les deux lignes), évidemment que le chiffre qu’on cherche est de même nature (si le chiffre inconnue coupe lui aussi deux fois les deux lignes, alors c’est forcément un 0). Sinon on utilise la superposition, qui sera orienté avec le nombre de fois que le chiffre coupe les lignes.

La superposition consiste à vérifier si à un même emplacement dans deux matrices (donc qui font la même taille), il y a la même valeur. Comme on ne s’intéresse qu’au chiffre (et pas aux espaces entre), on vérifie si pour tous les 1 d’une matrice, il y a au même emplacement un 1 aussi dans l’autre matrice. On superpose ainsi la matrice d’image inconnue avec toutes les autres matrices d’images connues après orientation.

  Par exemple on essaie de superposer ces deux 8.

*Instructions :*

1. Récupérer les images dans ‘images\_traitees’ et ‘images\_inconnues\_traitees’, en extraire leur matrice, les mettre dans un objet conteneur (listes ou autre). Récupérer les données du fichier CSV connu, associer chaque donnée à sa matrice, idem avec les données du CSV inconnu. Ensuite regrouper, dans les données connues, les matrices et coupe de lignes par nature de chiffre (ex : toutes les matrices et coupe de ligne pour les images de 0, de 1) dans des listes ou des sous-listes (ou autre).
2. Repérer à quelle(s) sous-liste(s) appartient le chiffre inconnu en fonction du nombre de fois qu’il coupe les lignes. S’il n’appartient qu’à une seule sous-liste, alors il a forcément la nature du chiffre de la sous-liste. Sinon appliquer la superposition orienté (on sait à quelles sous-listes appartient le chiffre, reste à savoir lequel parmi eux).
3. Codez une fonction qui fait la superposition entre deux matrices : On compare chaque membre d'une matrice avec le membre d'une autre matrice à la même position dans la matrice. S’ils ont la même valeur, il faut que ce soit un 1 (on ne s'intéresse qu'au chiffre, composé de 1). La matrice inconnue qui ressemble le plus à la matrice connue est donc de même nature que celui-ci. Faire la superposition avec les matrices connues qui ont les mêmes paramètres de coupe de ligne (par exemple coupe 1 fois la ligne horizontale et 3 fois la ligne verticale), c’est-à-dire dans les sous-listes qui ont les mêmes paramètres de coupe et donc la même nature de chiffre (ATTENTION : c’est-à-dire qu’il y a des images inconnues qui ont des paramètres de coupe qui sont les même dans plusieurs sous-listes ; pour savoir à quelle sous-liste le chiffre inconnue appartient, on superpose la matrice du chiffre inconnue avec toutes les matrices des sous-listes. La matrice connue avec qui elle se ressemble le plus a donc un chiffre de même nature que la sous-liste de la matrice connue).

Pour conclure, si vous avez la chance d’avoir la même écriture que moi, le même stylo que moi et le même téléphone que moi, vous aurez l’honneur d’avoir ce super algorithme pour reconnaître vos propres chiffres. Autant vous dire qu’il vous sera vachement utile.

Victor Gasse

1°5