

TP – IA - Algorithme KNN - Identification automatique de panneaux

Contexte

Nous allons développer un algorithme simple capable de reconnaître une image automatiquement à partir de l'apprentissage d'une base d'images sources. Pour cela, nous allons utiliser l'algorithme KNN, dit de recherche des « k plus proches voisins » en utilisant la norme euclidienne.

Nos images recherchées seront supposées prétraitées afin qu'elles ressemblent à cela :



Le prétraitement numérique aura donc réalisé les étapes suivantes :

- Transformation projective en couleur afin d'avoir une image « vue de face » et cadrée.



- Redimensionnement des images avec 100 lignes et 100 colonnes, soit 10 000 pixels RGB.
- Enregistrement du résultat au format BMP.

Les images sources permettant l'apprentissage auront subi un traitement supplémentaire de mise en blanc du fond en dehors du panneau, et nous nous limiterons à des panneaux circulaires.



Mise en place de l'algorithme KNN

Nous allons travailler avec des n-uplets (listes de n termes) qui seront associées à chaque image. On suppose que le nombre de termes n de chaque n-uplets est identique dans tout le sujet.

On rappelle que la distance euclidienne entre deux n-uplets $u = (u_0, u_1, \dots, u_{n-1})$ et $v = (v_0, v_1, \dots, v_{n-1})$ est le résultat du calcul suivant :

$$D = \sqrt{\sum_{i=0}^{n-1} (v_i - u_i)^2}$$

Q1 : Créer une fonction *Distance_uv(u,v)* calculant la distance euclidienne entre les deux n-uplets sous forme de listes *u* et *v*

Q2 : Créer une fonction *Distance(u,Lv)* renvoyant une liste des distances euclidiennes entre *u* et tous les n-uplets *v* de la liste *Lv* ainsi que l'indice associé sous la forme [*Distance entre u et v*, *indice de v dans Lv*].

Q3 : Créer la fonction *Proches(u,Lv,k)* qui renvoie une liste des *k* plus proches voisins de *u* dans la liste *Lv* au sens de la norme euclidienne, soit les *k* listes [*dst,ind*].

Remarques :

- On supposera que *k* est plus petit que le nombre de n-uplets de *Lv*
- On autorise l'utilisation de *L.sort()* pour trier les éléments de *L* en place par rapport à la première composante de tous ses éléments (les distances ici)

Lecture des images

On donne le code suivant qui permet d'afficher les images (au format *array*) sur une figure.

La fonction *Lecture(Chemin)* renvoie l'*array* associé à l'image de chemin contenu dans la variable *Chemin*.

```
# Affichage
import matplotlib.pyplot as plt
plt.close('all')

def Affiche(image):
    plt.figure()
    plt.imshow(image)
    plt.axis('off')
    plt.show()
    plt.pause(0.00001)

# Lecture des images

def Lecture(Chemin):
    Image = plt.imread(Chemin)
    return Image
```

Fonctions d'analyse des images

Pour la reconnaissance des images, il existe de nombreuses méthodes. Nous optons ici pour une méthode se basant sur la proportion de RGB dans les images.

Pour chaque image, on crée une liste L_RGB des valeurs de R, G et B de chacun de ses pixels. Ainsi, avec des images ayant toujours la même dimension de 100x100 pixels, on obtient une liste de 30 000 valeurs pour chacune, ce qui nous permet bien d'avoir des n-uplet de la même taille

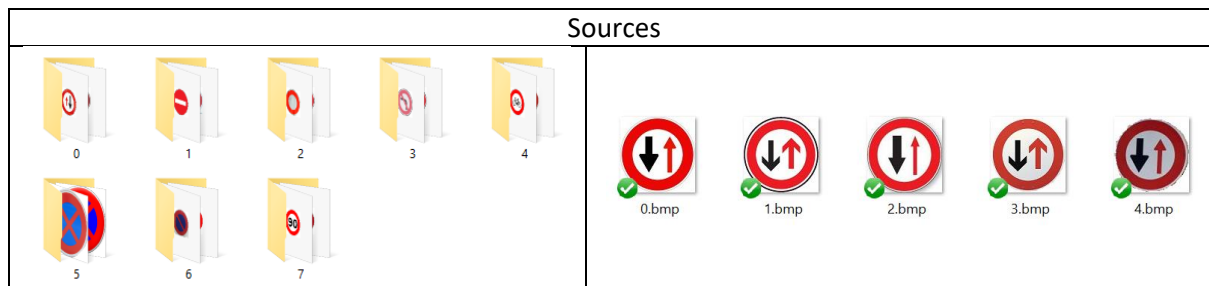
Q4 : Créer une fonction *Analyse(Image)* qui, à partir d'une image sous forme d'*array*, renvoie la liste L_RGB associée.

Attention : Il faut transformer les R, G et B en flottants, sinon il y aura *overflow* avec les *uint8* lors des calculs de distances.

Q5 : Créer une fonction *Analyse_Globale(L_Chemin)* qui, à partir d'une liste des chemins des images à analyser, renvoie la liste des listes L_RGB de chacune des images concernées.

Création de la base des données

Nous disposons d'une base de données dans laquelle apparaît 8 dossiers numérotés de 0 à 7. Chacun de ces dossiers contient 5 images sources du même panneau numérotées de 0 à 4, qui vont servir à l'apprentissage :



Cela nous donne les listes :

```
Dossiers = [0,1,2,3,4,5,6,7]
Nb_Images_Dossiers = [5,5,5,5,5,5,5,5]
```

La liste *Dossiers* liste les numéros des dossiers présents dans le répertoire *Sources*, et la liste *Nb_Images_Dossiers* répertorie le nombre d'images contenues dans chacun de ces dossiers.

Chaque image contenue dans le dossier source possède :

- Un chemin.
- Un numéro de dossier.
- Un numéro d'image.

On souhaite créer les trois listes *Liste_Chemin*, *Liste_Dossier* et *Liste_Num*, qui pour un même indice (et donc pour une même image), contiennent ces trois informations ci-dessus. Pour la suite, on appellera « **Indice d'une image** » son indice dans ces trois listes.

On donne le code utilisant *Dossiers* et *Nb_Images_Dossiers* créant les listes *Liste_Chemin*, *Liste_Dossier* et *Liste_Num*.

```
## Création de la base des données
# Ouverture des images sources

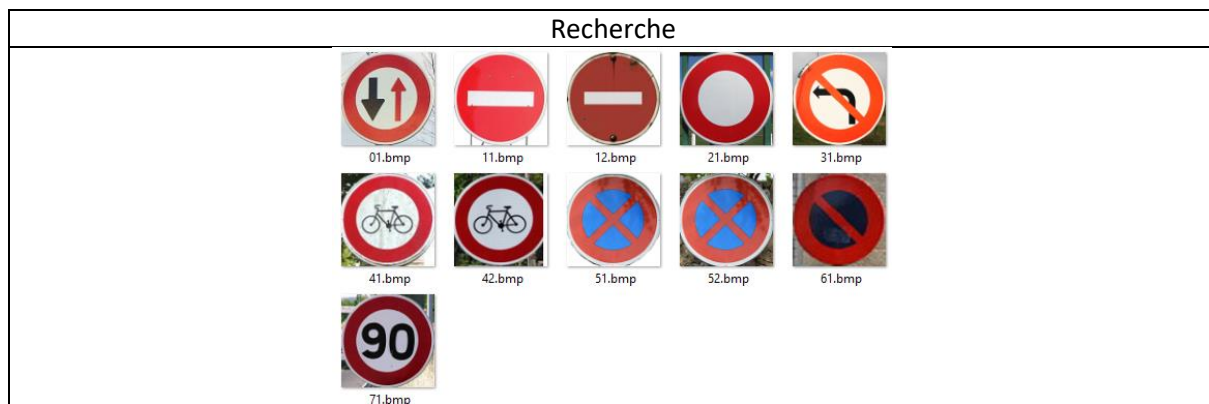
Sources = "Sources\\"
Dossiers = [0,1,2,3,4,5,6,7]
Nb_Dossiers=len(Dossiers)
Nb_Images_Dossiers = [5,5,5,5,5,5,5,5]

Liste_Chemin = []
Liste_Dossier = []
Liste_Num = []
for d in Dossiers:
    Nb_Im = Nb_Images_Dossiers[d]
    for im in range(Nb_Im):
        Chemin = Sources + str(d) + "\\" + str(im) + ".png"
        Liste_Chemin.append(Chemin)
        Liste_Dossier.append(d)
        Liste_Num.append(im)
```

Q6 : Ecrire le code nécessaire à la création de la liste de listes « *Donnees* », contenant les listes *L_RGB* de toutes les images sources en utilisant les fonctions créées précédemment.

Reconnaissance automatique

Nous disposons d'un dossier nommé « *Recherche* » contenant des images à rechercher automatiquement à l'aide de l'algorithme mis en place. Elles sont toutes issues d'une photo en situation réelle et le premier numéro de leurs noms correspond au dossier auquel elles devraient appartenir.



Les images sources ayant un fond blanc, notre algorithme s'adapte automatiquement à n'importe quel fond. En effet, une image recherchée ayant un fond quelconque sera à la même « distance » que toutes les images sources sur la partie extérieure, l'algorithme sélectionnera alors celle qui se rapproche le plus dans la comparaison du contenu intérieur du panneau.

On souhaite, avec le code ci-dessous, ouvrir, afficher et analyser (création de sa liste *L_RGB*) l'une des images du dossier *Recherche*.

```
# Ouverture et analyse de l'image recherchée
Recherche = "Recherche\\"
Im_Cherchee_Chemin = Recherche + "71.png"
Im_Cherchee = Lecture(Im_Cherchee_Chemin)
Affiche(Im_Cherchee)
Im_Cherchee_Infos = Analyse(Im_Cherchee)
```

Q7 : Créer un code qui détermine les $k=5$ plus proches voisins de l'image recherchée et crée les listes *Resultat_Ind* (indices des images résultats), *Resultat_Dossiers* (dossiers correspondants) et *Resultat_Num* (numéros des images dans les dossiers) et qui affiche dans la console dossiers et numéros des images trouvées.

Q8 : Créer une fonction *Max_Occurences(L)* qui renvoie le terme apparaissant le plus dans *L*, et le premier s'il y a des exæquos.

En rajoutant le code suivant, on peut donc déterminer le dossier résultat, l'afficher dans la console et afficher l'une des images de ce dossier.

```
Dossiers_final = Max_Occurences(Resultat_Dossiers)
print("Dossier: ",Dossiers_final)

# Affichage du résultat

Im_Trouvee_Chemin = Sources + str(Dossiers_final) + "\\0" + ".png"
Im_Trouvee = Lecture(Im_Trouvee_Chemin)
Affiche(Im_Trouvee)
```