Reconnaissance d'objets

La reconnaissance d'objets a pour objectif d'identifier un objet particulier dans une base de données image ou une vidéo. Cette identification est effectuée grâce à des attributs décrivant la forme, la couleur et/ou la texture. Dans ce TP, nous proposons de caractériser 4 classes d'objets. Nous évaluerons la pertinence de cette caractérisation en présentant au système de nouveaux objets et en vérifiant si ceux-ci sont assignés correctement à leur classe d'appartenance.

1 Base de données image

La base de données image considérée ici est constituée de 4 classes : panneaux de signalisation, personnages, voitures et camions. Pour chaque classe, 6 objets sont utilisés (cf. figure 1).

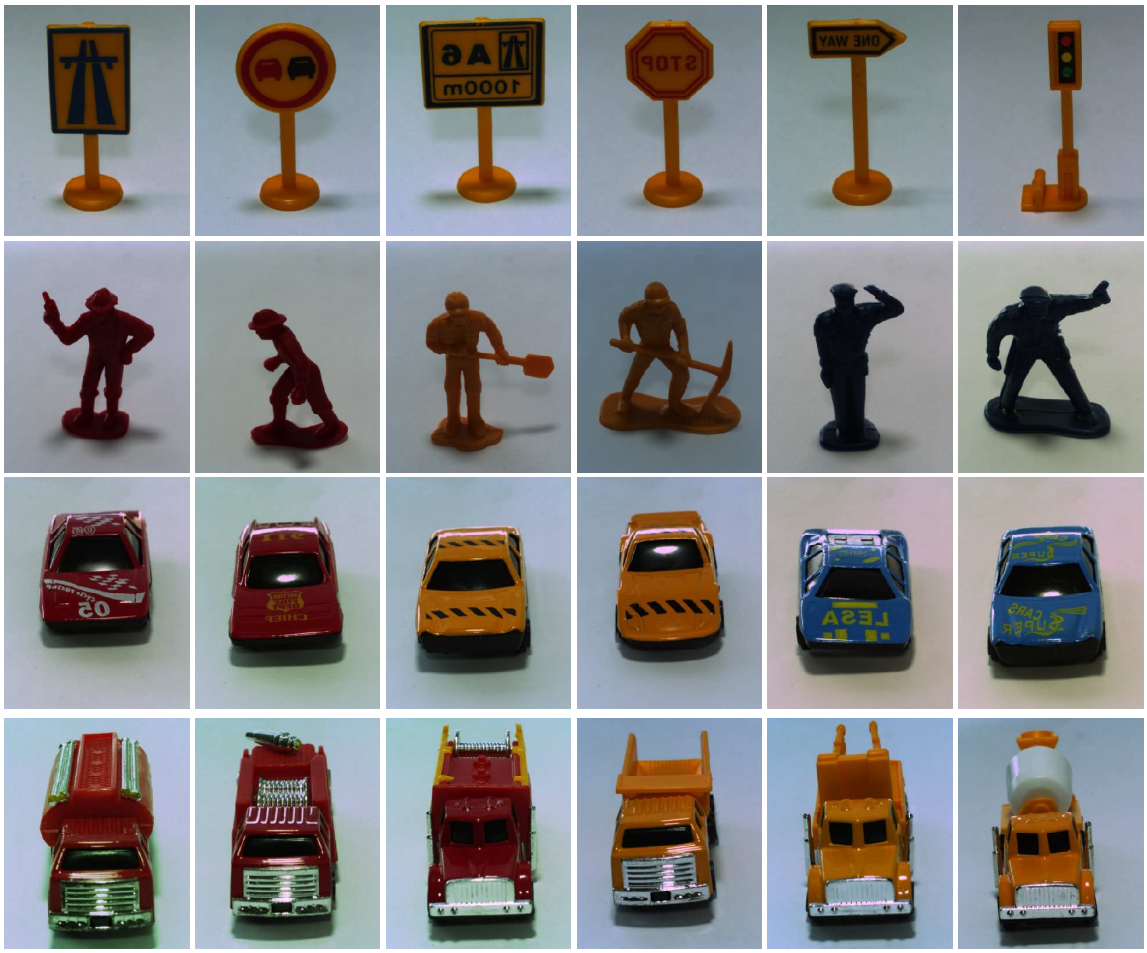

FIGURE 1 – Base de données image : chaque ligne représente une classe.

Pour chacun des 24 objets présentés ci-dessus, 5 images représentant l'objet selon 5 angles de vue différents ont été acquises (cf. figure 2).

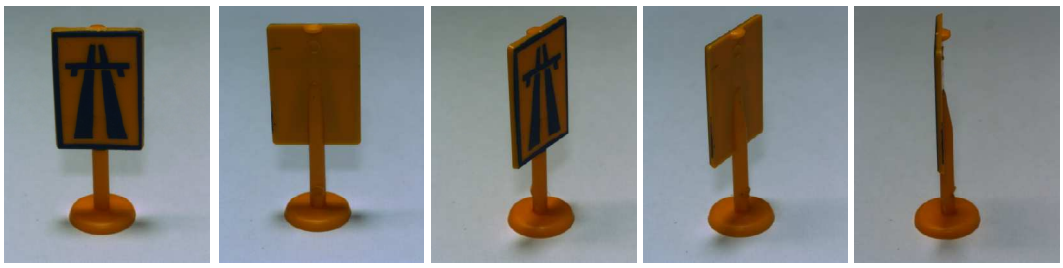

FIGURE 2 – Acquisition d'un objet selon 5 angles de vue différents.

Les images ont été enregistrées au format .png en suivant l'incrémentation suivante :
- de 000001.png à 000005.png pour le premier objet pris sous ses différents angles de vue,
- de 000006.png à 000010.png pour le deuxième objet pris sous ses différents angles de vue,
- etc

Un objet de chaque classe, pris sous ses différents angles de vue, sera utilisé pour construire la base d'apprentissage. Les 5 autres objets de la classe seront utilisés pour tester la pertinence de la caractérisation. La base d'apprentissage est donc constituée de 20 images (5 images par classe) et la base à tester de 100 images (25 images par classe).

2 Caractérisation : extraction des attributs de forme

La fonction qui va être utilisée dans ce TP pour extraire les attributs de forme nécessite une binarisation préalable des images.

1) Créer un nouveau fichier **TP.m**. Au sein de ce fichier, lire l'image 000002.png et la convertir en niveaux de gris.

2) Binariser l'image en niveau de gris obtenue de sorte à obtenir la forme en blanc et le fond en noir.

3) Le seuil utilisé pour binariser les images pouvant varier d'une image à l'autre, utiliser la fonction *graythresh* pour déterminer de manière automatique le seuil en fonction de l'image passée en paramètre.

Afin de parfaire la segmentation de l'image, plusieurs opérations morphologiques peuvent être utilisées :
- Les fonctions *imerode* et *imdilate* permettent respectivement de réaliser des opérations d'érosion et de dilatation sur des images binaires à partir d'un élément structurant. Cet élément structurant peut être défini soit par une matrice de 0 et de 1, soit en utilisant la fonction *strel* qui permet de configurer des éléments structurants élémentaires.
- La fonction *imclearborder* est une fonction morphologique qui permet de supprimer des régions qui sont au contact des bords de l'image binaire.
- La fonction *bwareaopen* est basée sur une analyse en composantes connexes. Elle permet de supprimer des régions de trop petites tailles dans une image binaire.
- La fonction *imfill* est une fonction morphologique qui permet de combler les "trous" dans les régions d'une image binaire.

4) Utiliser les fonctions morphologiques nécessaires pour améliorer votre binarisation. Vérifier en zoomant sur l'image obtenue qu'il ne reste pas de petits pixels parasites sur le fond de l'image et que les pixels relatifs à l'objet appartiennent bien à une unique région.

La fonction *bwlabel* détermine les régions de pixels connexes dans une image binaire. Elle renvoie :
- une image indexée L, où chaque index correspond à une étiquette (label), c'est à dire une région,
- et le nombre N de régions présentes dans l'image.

La fonction *label2rgb* permet quant à elle de visualiser le résultat de l'analyse en composantes connexes effectuée par la fonction *bwlabel*.

5) A l'aide de ces deux fonctions, vérifier que le nombre de régions présentes dans l'image est bien égal à 1.

6) Effectuer cette vérification pour d'autres images et réajuster vos opérations morphologiques et votre seuil de binarisation si nécessaire.

La fonction *regionprops* permet de mesurer différents paramètres de régions contenues dans une image d'étiquettes et donc obtenues après une analyse en composantes connexes. Les paramètres mesurés par cette fonction sont les suivants :
- la surface,
- le périmètre,
- le diamètre,
- la longueur de l'axe principal d'inertie (longueur de la région),
- la longueur du second axe d'inertie (largeur de la région),
- l'angle de l'axe principal d'inertie (orientation de la région),
- les coordonnées du centre de gravité,
- les coordonnées du cadre circonscrit à la région (boundingbox),
- etc

7) Calculer les coordonnées du centre de gravité de la forme, ainsi que sa surface, son périmètre, son diamètre, sa longueur, sa largeur et son orientation.

8) Écrire une fonction appelée **AttributsForme.m** permettant de retourner le vecteur des attributs de forme à partir d'une image en niveau de gris. Tester cette fonction.

3 Classification

Maintenant qu'il est possible de calculer un vecteur d'attributs de forme à partir d'une image, nous allons mettre en place la procédure de classification. Cette procédure va nous permettre d'analyser la pertinence de nos attributs en mesurant le taux d'images bien classées.

Le processus de classification est divisé en deux étapes successives :

1. L'apprentissage : l'objectif est de "construire" des classes d'objets à partir de l'ensemble d'images d'apprentissage. Pour cela, les objets présents dans les images d'apprentissage sont décrits par l'ensemble d'attributs considéré.
2. La classification : durant cette seconde phase, nous utiliserons un classifieur afin d'assigner chaque image test à une classe en fonction de sa similarité. Cette similarité entre images est mesurée en comparant les vecteurs d'attributs.

3.1 Apprentissage

L'apprentissage consiste à ouvrir chacune des images de la base d'apprentissage, calculer et enregistrer dans une variable le vecteur d'attributs de chaque image d'apprentissage ainsi que la classe correspondante.

Pour cela, nous allons utiliser une boucle répétitive de la manière suivante :

```
nb_classe = 4; % défini le nombre de classes
nb_image = 30; % défini le nombre d'images par classe
chemin = 'Base\'; % défini le chemin vers la base d'images
nb_ima = nb_classe*nb_image;
nb_ima_train = nb_ima/6;
% pour chaque classe, seul un objet sur 6 est utilisé pour l'apprentissage
Attributs = zeros(nb_ima_train, 8);

%% Apprentisage
ima_label=0;
for i_train=1:nb_ima
    if (((1<=i_train)&&(i_train<=5))||((31<=i_train)&&(i_train<=35))
    ||((61<=i_train)&&(i_train<=65))||((91<=i_train)&&(i_train<=95)))
        ima_label=ima_label+1;
        % Enregistrement du numéro de la classe dans un tableau
        num_classe_train(ima_label) = floor((i_train-1)/nb_image) + 1;
        % Concaténaion des chaînes de caractères
        % pour constituer le chemain d'accès au fichier image
        if (i_train/10 < 1)
            fichier_train = [chemin '00000' int2str(i_train) '.png'];
        else
            if (i_train/100 < 1)
                fichier_train = [chemin '0000' int2str(i_train) '.png'];
            else
                fichier_train = [chemin '000' int2str(i_train) '.png'];
            end
        end
        % Affichage du numéro de la classe
        disp([fichier_train ' Classe ' int2str(num_classe_train(ima_label))]);

        % Ouverture de l'image
        Ima_train = imread(fichier_train);

        % Conversion en niveaux de gris
        Ima_gray = rgb2gray(Ima_train);

        % Ajouter ici si nécessaire des opérations de prétraitement

        % Extraction des attributs de forme
        Attributs(ima_label,:) = AttributsForme(Ima_gray);
    end
end
```

9) Dans un nouveau script **Reconnaissance.m**, réaliser l'apprentissage du processus de classification (Note : si, pour certaines images d'apprentissage, l'ajustement des opérations morphologiques et du seuil de binarisation ne suffit pas à obtenir une unique région représentant l'objet, une solution est de ne conserver lors de l'extraction des attributs que la région qui possède la plus grande aire).

10) A l'aide de la commande :

```
scatter(Attributs(:,3),Attributs(:,4),2*num_classe_train,num_classe_train);
```

visualiser les nuages de points dont les coordonnées sont deux des attributs calculés (dans cet exemple, la surface et le périmètre). Déterminer ensuite visuellement les attributs qui semblent les plus pertinents pour la classification.

Les attributs calculés peuvent avoir des échelles de valeurs très différentes, ce qui peut leur donner un poids différent lors du processus de comparaison et biaiser les résultats de classification.

Afin d'obtenir des données indépendantes de l'unité ou de l'échelle choisie, il est nécessaire de centrer et réduire les variables utilisées. Pour cela, nous devons calculer la moyenne (fonction *mean*) et l'écart-type (fonction *std*) des valeurs de chaque attribut sur l'ensemble des données d'apprentissage.

Centrer une variable consiste à soustraire sa moyenne à sa valeur et réduire une variable consiste à diviser sa valeur par son écart-type.

11) Centrer et réduire les attributs de forme calculés pendant l'apprentissage.

3.2 Décision

Nous allons maintenant mettre en place la procédure permettant de classer une image de la base test, l'objectif étant de reconnaître le type d'objet correspondant à l'image analysée.

Le classifieur utilisé pour cela sera l'algorithme du plus proche voisin, associé à la distance euclidienne. L'objectif de ce classifieur est de mesurer la distance entre le vecteur d'attributs de l'image à classer avec chacun des vecteurs d'attributs des images de la base d'apprentissage. L'image à classer sera alors assignée à la classe de l'image d'apprentissage pour laquelle la distance est minimale.

12) Compléter le programme précédent afin d'ouvrir une image de la base test et classer cette image par l'algorithme du plus proche voisin.

Le taux de classification correspond au rapport entre la somme des images test bien classées et le nombre total d'images test.

13) En vous inspirant du programme permettant l'apprentissage, compléter votre script afin de calculer le taux de classification (Note : si, pour certaines images test, l'ajustement des opérations morphologiques et du seuil de binarisation ne suffit pas à obtenir une unique région représentant l'objet, une solution est de ne conserver lors de l'extraction des attributs que la région qui possède la plus grande aire).

14) Valider votre programme en classant les images de la base d'apprentissage : vous devez obtenir un taux de 100%.

15) Une fois votre programme validé, revenir à la classification des images test et relever le taux de classification, la matrice de confusion (fonction *confusionmat*), ainsi que les temps de traitement des phases d'apprentissage et de décision (instructions *tic* et *toc*).

4 Essais expérimentaux

16) Analyser et comparer votre approche lorsque la distance euclidienne est remplacée par la distance de Manhattan. On comparera les taux de classification, les temps de traitement et la matrice de confusion.

17) Même question lorsque le classifieur du plus proche voisin est remplacé par le classifieur du plus proche barycentre.

Le nombre d'attributs de reconnaissance de forme extrait ici est de 8 (coordonnées du centre de gravité de la forme (2), surface, périmètre, diamètre, longueur, largeur et orientation). Ces attributs sont plus ou moins pertinents pour discriminer les classes en présence (cf. question 10).

18) Tester votre approche en sélectionnant un nombre réduit d'attributs pertinents parmi l'ensemble d'attributs disponibles. Observer l'influence de cette sélection sur les taux de classification et les temps de traitement.

19) Dans un nouveau script **ReconnaissanceCouleur.m**, regarder si l'apport de l'information couleur permet d'améliorer les résultats de classification. On décomposera pour cela chaque image couleur en 3 images-composante afin d'extraire 3 fois plus d'attributs par image.

5 Descripteurs HOG

Dans cette partie, nous proposons d'étudier une autre approche utilisant les descripteurs HOG afin de faire de la reconnaissance d'objets. L'idée générale de cette méthode est de détecter des points remarquables permettant de caractériser l'objet.

20) Dans un nouveau script intitulé **HOG.m**, ouvrir une image de la base, la convertir en niveau de gris, et extraire les descripteurs HOG à l'aide de la fonction *extractHOGFeatures*.

21) Visualiser le résultat de cette extraction grâce à l'option "Visualization".

22) Faire varier le paramètre 'CellSize' et observer le résultat obtenu.

23) En vous inspirant de votre script **Reconnaissance.m**, créer un nouveau programme **ReconnaissanceHOG.m** qui classifie vos images à l'aide des descripteurs HOG.

24) Faites varier le paramètre 'CellSize', ainsi que la mesure de similarité, et comparer les résultats de classification obtenus.