Université Sidi Mohammed Ben Abdallah de Fès

Faculté des Sciences Dhar El Mahraz

Image mining

Atelier N 2 : Classification

Prof : My Abdelouahed Sabri

abdelouahed.sabri@gmail.com

1 Introduction

L'objectif principal de cet atelier est de créer un système de classification d'images à base des algorithmes de Machine Learning et Data Mining. La classification est le processus de prédiction de la classe d'une image donnée. Les classes sont parfois appelées étiquettes ou catégories. Comme pour le CBIR, la classification sera effectuée en se basant sur le vecteur descripteur de l'image au lieu d'utiliser l'image. La classification peut être supervisée ou non supervisée ; pour la classification supervisée on dispose d'éléments déjà classés avec leurs étiquètes en avance alors que pour le cas de classification non supervisée les éléments ne sont pas classées et on cherche à les regrouper en classes.

Dans cet atelier, notre objectif est de développer un système de classification supervisé. Dans ce système, deux parties (ou 3) seront élaborées :

1-Apprentissage : Créer un modèle en se basant sur les descripteurs de la base d'images. Cette phase consiste en l'approximation d'une fonction de mappage entre la matrice de caractéristiques (variables d'entrées) et les étiquettes (variables de sorties)

2-Classification (prédiction) : prédire l'étiquette d'une nouvelle entrée (image) par le modèle élaboré.

La figure ci-dessous présente le schéma général du processus de classification

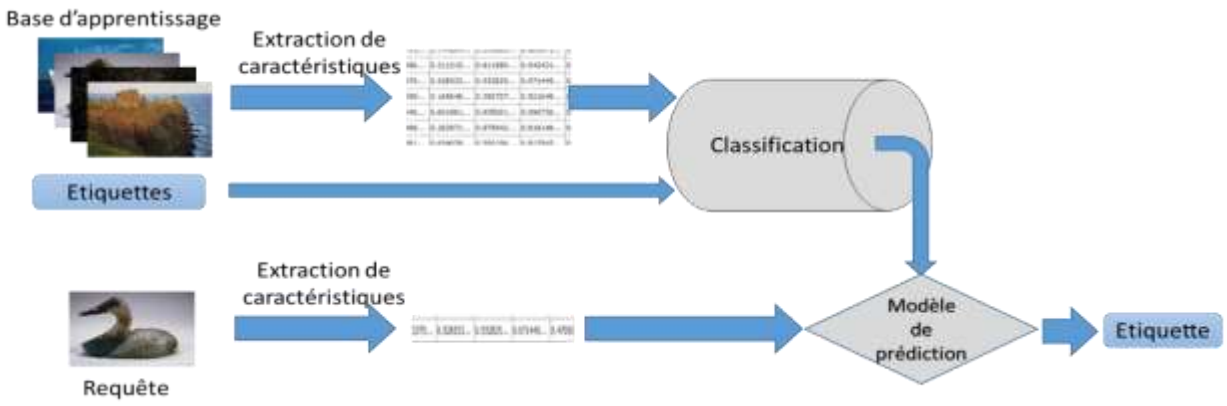

Figure .1 Schéma fonctionnel d'un système de classification

La validation et l'évaluation d'un système de classification est effectuée en divisant la base d'images d'apprentissage en deux parties : partie d'apprentissage ou d'entrainement (training) et base de tests ou de validation. Sachant que les images sont étiquetées au préalable, dans la base

d'apprentissage les images et les étiquettes seront utilisées pour élaborer le modèle de classification alors que les images de la base de tests seront utilisées pour prédire les étiquettes de chaque image. Ces étiquettes prédites seront comparées avec les étiquettes originelles pour mesurer la performance du modèle élaboré. La figure ci-dessous présente le schéma général du processus de classification/validation du modèle.

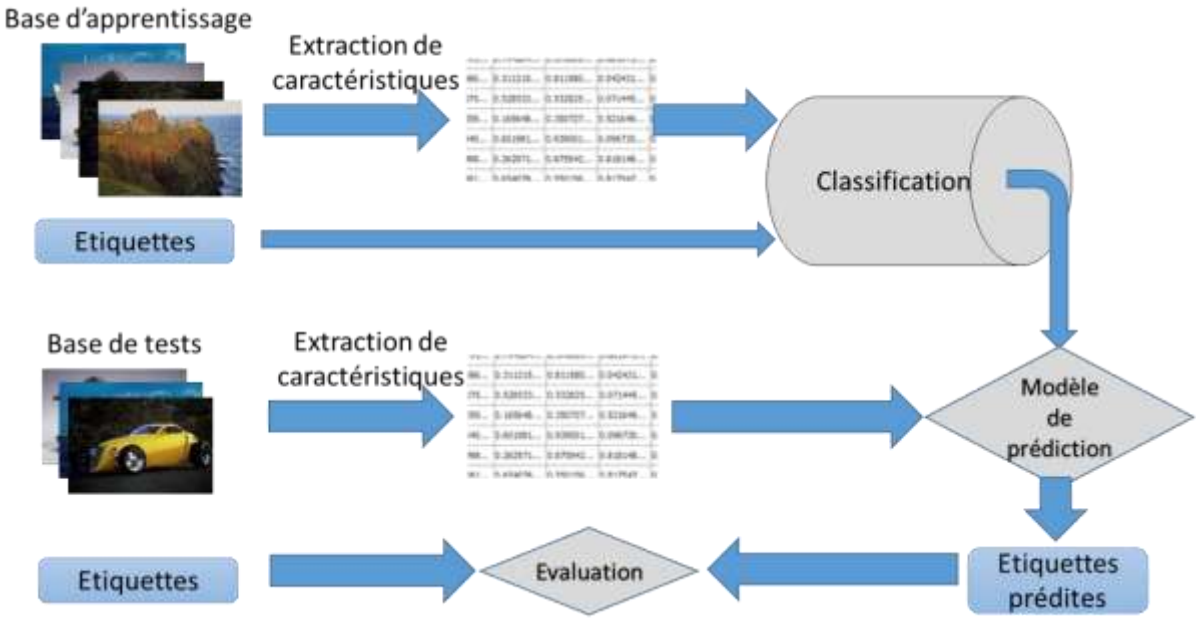

Figure .2 Schéma général d'un système de classification/validation

Une multitude d'algorithmes de classification supervisée ont été présentés dans la littérature. Ci-dessous une liste non exhaustive des méthodes les plus utilisées :

- Machine à vecteurs de support (SVM ; Support Vector Machine)
- Classification naïve bayésienne
- Méthode des k plus proches voisins (KNN ; K
- Arbre de décision
- Réseau de neurones
- …

2 Implémentation

L'implémentation sera réalisée sous Matlab en utilisant une partie de la base de données COREL. Cette base d'images contient 10800 images et qui est classée en 80 groupes d'images.

2.1 Préparation de la base d'images

Dans cet atelier de classification, nous allons utiliser seulement deux groupes d'images (deux classes) ; la première classe va contenir les images de type (classe) « voiture » et la deuxième classe va contenir les images de types « bateau ». Chaque classe sera mise dans un dossier portant successivement les noms « obj_car » pour les voitures et « obj_ship » pour les bateaux. La classe « obj_car » contient 400 images et la classe « obj_ship » contient 90 images. La figure ci-dessous présente quelques images des deux classes

Figure .3 Quelques images des classes « obj_car » et « obj_ship »

2.2 Objectif

L'objectif de la classification supervisée est de pouvoir prédire la classe d'une (nouvelle) image à partir des connaissances (classification déjà établies en se basant sur les classes des images de la base. Ainsi, il faut établir un modèle de classification utilisant les deux classes « car » et « ship » de la base. Le classifieur choisi pour cette tâche est le SVM.

Comme pour le CBIR, nous allons premièrement construire une matrice d'indexes à base des caractéristiques des images. Les caractéristiques choisies sont : Les moments statistiques de couleurs dans l'espace RGB, l'histogramme de couleurs dans l'espace HSV, les mesures statistiques de la matrice GLCM pour la texture et les moments de Hu pour la forme (voir atelier CBIR). Cette matrice de caractéristiques (features) sera passée avec les étiquettes (classes) au classifieur pour élaborer un modèle de classification qui sera utilisé pour la prédiction (voir Figure. 1)

2.3 Extraction de caractéristiques (features extraction)

La première étape du processus de classification et l'extraction de caractéristiques. Contrairement au CBIR, il faut identifier pour chaque image le vecteur descripteur et sa classe.

```
[features, etatFull]=createFeatures('DB2C');
function [features, etat]=createFeatures(fold)
% fold: nom du dossier contenant les images par dossier (classe)
% features: Matrice de caractéristiques de taille (nombre_d_image x
nombre_features)
% etat: Matrice d'état de taille (nombre_d_image x 2). La première
% colonne contient le nom de la classe et le deuxième contient un id de
% classe 1 pour la première classe, 2 pour la deuxième ...
lst=dir(fold);
k=1;
for i=3:length(lst)
    if (lst(i).isdir)
        sub=dir(strcat(lst(i).folder,'\',lst(i).name,'\*.jpg'));
        for j=1:length(sub)
            etat(k,1)=string(lst(i).name);
            %etat(k,2)=i-2;
            IDB=imread(strcat(lst(i).folder,'\',lst(i).name,'\',sub(j).name));
            features(k,:)=getFeatures(IDB, 50);
            k=k+1;
        end
    end
end
end
```

2.4 Apprentissage

Une fois la matrice de caractéristiques est créer et pour chaque vecteur sa classe, on passe à l'apprentissage. Le classifieur choisi est le SVM

```
% Train with SVM one-vs-one encoding scheme
svmStruct =
fitcsvm(features,etatFull,'Standardize',true,'KernelFunction','rbf','KernelSca
le','auto');
```

2.1 Prédiction de la classe d'une nouvelle image

Ici nous allons prédire la classe d'une nouvelle image (supposée non utilisée dans l'apprentissage). Pour cette image il faut premièrement extraire le vecteur descripteur et l'utiliser pour identifier sa classe

```
% Prédire la classe d'une nouvelle image
% Logiquement, l'image ne doit pas être dans la base d'apprentissage
Ireq=imread('29023.jpg');
features_req=getFeatures(Ireq, 50);
```

```
% Prédire la classe de l'image requête
class_pred=predict(svmStruct,features_req);
disp(strcat('La classe est: ',class_pred))
figure, imshow(Ireq), title(strcat('La classe est: ',class_pred))
```

Pour le cas du code de prédiction, on remarque que effectivement le classifier SVM a pu en se basant sur la matrice de caractéristiques et les classes équivalente de prédire correctement la classe d'une nouvelle image. L'évaluation ici de résultat et effectuée visuellement en affichant l'image et sa classe. Mais si on veut évaluer réellement notre modèle de classification il faut tester avec plusieurs images et voir le nombre de fois le modèle a pu classifier correctement et inexactement des images données.

2.2 Evaluation du modèle de classification

Dans ce type de système, il est commode pour évaluer la performance de prédiction du modèle de classification établi de diviser la base d'images en deux parties ; une partie pour l'apprentissage (training) et une autre partie pour le test et la validation du modèle (voir Figure .2).

Les mesures de validation des résultats de la classification utilisées sont : Sensibilité (taux de TP), Spécificité (taux de TN) et de précision (accuracy).

$$\text{Sensitivity}=TP/(TP+FN)$$

$$\text{Specificity} =TN/(FP+TN)$$

$$\text{Accuracy}= (TP+TN)/(TP+TN+FP+FN)$$

Avec :

- TP: nombre de « True Positives ». C'est le nombre d'images classées par le modèle dans la classe « obj_car » qui appartiennent effectivement à cette classe
- TN: nombre de « True Negatives. C'est le nombre d'images classées par le modèle dans la classe « obj_ship » qui appartiennent effectivement à cette classe
- FP: nombre de « False Positives » : C'est le nombre d'images qui sont classées par le modèle dans la classe « obj_car » alors et qu'elles appartiennent à la classe « obj_ship »
- FN: nombre de « False Negatives ». C'est le nombre d'images qui sont classées par le modèle dans la classe « obj_ship » alors et qu'elles appartiennent à la classe « obj_car »

Aussi, on peut utiliser la matrice de confusion pour évaluer facilement un modèle de classification. Cette matrice permet de visualiser facilement le nombre de TP, TN, FP et FN

	Classe « obj_car »	Classe « obj_ship »
Classe « obj_car »	TP	FN
Classe « obj_ship »	FP	TN

Après extraction des caractéristiques de la base avec leurs étiquettes, nous allons répartir la base d'indexes en deux parties ; base d'apprentissage (80%) et base de test (20%). Le modèle de classification sera élaboré utilisant la base d'apprentissage alors que la prédiction sera avec la base de test. Après, il faut comparer les étiquètes prédites de la base de test avec celle réelles pour évaluer la classification.

```
P = cvpartition(etatFull,'Holdout',0.20);
svmStruct =
fitcsvm(features(P.training,:),etatFull(P.training),'Standardize',true,'Kernel
Function','rbf','KernelScale','auto');
[C,score] = predict(svmStruct,features(P.test,:));

errRate = sum(etatFull(P.test)~= C)/P.TestSize  %erreur de classification
conMat = confusionmat(etatFull(P.test),string(C)) % La matrice de confusion
TP=conMat(1,1);
TN=conMat(2,2);
FP=conMat(2,1);
FN=conMat(1,2);
Accuracy= 100*(TP+TN)/ (TP+TN+FP+FN) % le taux de classification
```

Les résultats obtenus pour notre base d'images est comme suit :

	Classe « obj_car »	Classe « obj_ship »
Classe « obj_car »	80	0
Classe « obj_ship »	1	17

Accuracy =98.9796%

De la matrice de confusion on peut conclure que 80 images de classe « obj_car » ont été correctement classées, 17 images de classe « obj_ship » ont été correctement classées et 1 image de classe « obj_car » a été classée en « obj_ship ». Et, avec un taux de classification de presque 99% on peut dire que le modèle de classification est performant.

Vous allez remarquer que si vous lancez une autre fois la classification avec une nouvelle répartition des images de la base en base de apprentissage et base de test que les valeurs de la matrice de confusion changent et ainsi que l'accuracy, ceci est dû à ce que la répartition est faite d'une manière aléatoire et qui peut changer à chaque appel.

2.3 Etude comparative avec d'autres algorithmes de classification

La classification repose essentiellement sur deux paramètres ; le choix des caractéristiques et l'algorithme de classification. Dans la partie précédente, le SVM a été choisi comme classifieur alors qu'il existe d'autres algorithmes de classification que pouvez utiliser. Le choix du classifieur est fortement lié au type d'images et à la taille de la base. Dans ce qui suit nous allons élaborer une étude comparative en termes d'accuracy pour différent algorithmes de classification à savoir : SVM, KNN, l'arbre de décision (Decision Trees).

Nous allons précéder successivement pour chaque algorithme de classification par créer un modèle de classification et après faire la prédiction et à la fin calculer et afficher l'accuracy de classification

```
% SVM
svmStruct =
fitcsvm(features(P.training,:),etatFull(P.training),'Standardize',true,'Kernel
Function','rbf','KernelScale','auto');
[C,score] = predict(svmStruct,features(P.test,:));
errRate = sum(etatFull(P.test)~= C)/P.TestSize;  %erreur de classification
Accuracy_SVM=(1-errRate)*100
% KNN
knnModel=fitcknn(features(P.training,:),etatFull(P.training),'NumNeighbors',5,
'Standardize',1);
[C,score] = predict(knnModel,features(P.test,:));
errRate = sum(etatFull(P.test)~= C)/P.TestSize;  %erreur de classification
Accuracy_KNN=(1-errRate)*100
% Decision Tree
DTModel=fitctree(features(P.training,:),etatFull(P.training));
[C,score] = predict(DTModel,features(P.test,:));
errRate = sum(etatFull(P.test)~= C)/P.TestSize;  %erreur de classification
Accuracy_DT=(1-errRate)*100
```

Le tableau ci-dessus présente le taux classification pour chaque algorithme de classification

	SVM	KNN	Decision Trees
Accuracy	97.9592	96.9388	94.8980

Les résultats présentés dans le tableau montrent que l'algorithme SVM est nettement meilleur que les algorithmes KNN et Decision Trees. Sachant bien-sûr que ce taux de classification change en modifiant les images dans la base de training et de test.

3 Conclusion

L'objectif de cet atelier est de former le lecteur dans la classification supervisée des images sous Matlab. Ainsi, nous avons présenté comment préparer les images et comment préparer la matrice de caractéristique. Après, nous avons utilisé premièrement l'algorithme SVM pour créer le modèle de classification et par la suite nous avons prédit avec succès la classe de deux nouvelles images. Après, pour évaluer notre système, nous avons utilisé la notion de training/test et nous avons fait appel à la mesure d'accuracy (taux de classification) pour savoir à quel degré notre système de classification est puissant. Par la suite, nous avons présenté une étude comparative en utilisant deux autres algorithmes de classification à savoir le KNN et l'arbre de décision.

Comme conclusion, la nature de la base, les caractéristiques utilisées et l'algorithme de classification sont les clés de la réussite d'un système de classification.

4 Todo

Jusqu'ici, les résultats de classification par les algorithmes SVM, KNN et Arbre de Décision sont réalisé avec des paramètres basiques alors qu'il y a d'autres paramètres et d'autres algorithmes de classification à modifier et à utiliser pour augmenter le taux de classification.

Votre tâche dans cette partie est de :

1-Faire d'autres tests pour accroitre le taux de classification en utilisant la notion de training/test comme dans les exemples présentées précédemment afin d'identifier le meilleur algorithme de classification avec les bons paramètres.

2-Prédire les étiquettes des images dans le dossier « DataToPredict ». À signaler que dans cette partie le training doit être réalisé sur la base entière. Les étiquettes prédites ainsi que les noms des images doivent être enregistrés dans un fichier « Prediction_corelDB_**NotreNom**.csv ». Le fichier est à envoyer à l'adresse email « abdelouahed.sabri@gmail.com » au plus-tard le dimanche prochain.

   a. Le contenu du fichier doit être comme suit :

Name,classe
29011.jpg,obj_car
29013.jpg,obj_car
29055.jpg,obj_car

b. Le code Matlab pour l'exportation des noms et étiquettes vers le fichier *.csv est le suivant :

```
% C contient les étiquettes et
% imageNames les noms des images
Prediction=[imageNames,C];
T = cell2table(Prediction,'VariableNames',{'Name','classe'});
writetable(T,'Prediction_corelDB_VotreNom.csv');
```

5 Annexe

5.1 Programme Matlab de classification/prédiction

```
%% Training à exécuter une seule fois
% Le dossier "BD2C" contient deux sous dossier "obj_car" et "obj_ship".
% "obj_car" contient 400 images de voitures
% "obj_ship" contient 90 images de bateaux
% La première étape consiste à créer une matrice de caractéristiques
% (features) avec leurs classes respectivement
isFEx = input('Voulez-vous extraire les caractéristiques? taper Y si
oui:','s');
if (isFEx=='Y')
    [features, etatFull]=createFeatures('DB2C');
end
% Train with SVM one-vs-one encoding scheme
svmStruct =
fitcsvm(features,etatFull,'Standardize',true,'KernelFunction','rbf','KernelSca
le','auto');

% Prédire la classe d'une nouvelle image
% Logiquement, l'image ne doit pas être dans la base d'apprentissage
Ireq=imread('29023.jpg');
features_req=getFeatures(Ireq, 50);
% Prédire la classe de l'image requéte
class_pred=predict(svmStruct,features_req);
disp(strcat('La classe est: ',class_pred))
if (strcmp(class_pred,'obj_car'))
    etiquette='Voiture';
else
    etiquette='Bateau';
end
figure, imshow(Ireq), title(strcat('La classe est: ',string(etiquette)))
% 2éme exemple
Ireq=imread('535008.jpg');
features_req=getFeatures(Ireq, 67);
% Prédire la classe de l'image requéte
class_pred=predict(svmStruct,features_req);
```

```
disp(strcat('La classe est: ',class_pred))
if (strcmp(class_pred,'obj_car'))
    etiquette='Voiture';
else
    etiquette='Bateau';
end
figure, imshow(Ireq), title(strcat('La classe est: ',etiquette))
%% Extraction de caractéritiques
function [features, etat]=createFeatures(fold)
% fold: nom du dossier contenant les images par dossier (classe)
% features: Matrice de caractéristiques de taille (nombre_d_image x
nombre_features)
% etat: Matrice d'état de taille (nombre_d_image x 2). la premiére
% colonne contient le nom de la classe et le deuxiéme contient un id de
% classe 1 pour la première classe,2 pour la deuxième ...
lst=dir(fold);
k=1;
for i=3:length(lst)
    if (lst(i).isdir)
        sub=dir(strcat(lst(i).folder,'\',lst(i).name,'\*.jpg'));
        for j=1:length(sub)
            etat(k,1)=string(lst(i).name);
            %etat(k,2)=i-2;
            IDB=imread(strcat(lst(i).folder,'\',lst(i).name,'\',sub(j).name));
            features(k,:)=getFeatures(IDB, 50);
            k=k+1;
        end
    end
end
end
%% fonction pour créer le vecteur descripteur
function features = getFeatures(img, fsize)
features=zeros(fsize-1,1);
features=color_Moments(img);
features = [features, hsvHistogramFeatures(img)];
features = [features, textureFeatures(img)];
features = [features, shapeFeatures(img)];
end

function colorFeature = color_Moments(img)
% img: image à extraire les 2 primeirs moments (mean et std) de chaque
composante R,G,B
% sorite: vecteur de dimenssion 1x6 vector contenant les caractéristiques

% Activer cette ligne si vous voulez travailler dans l'espace de couleur
% HSV qui est meilleur que RGB
%img=rgb2hsv(img);
% extract color channels
R = double(img(:, :, 1));
G = double(img(:, :, 2));
B = double(img(:, :, 3));
% compute 2 first color moments from each channel
colorFeature=[mean(R(:)), std(R(:)), mean(G(:)), std(G(:)), mean(B(:)),
std(B(:))];
colorFeature=colorFeature/mean(colorFeature);
end
```

```
function hsvColor_Histogram = hsvHistogramFeatures(img)
% img: image à quantifier dans un espace couleur hsv en 8x2x2 cases identiques
% sortie: vecteur 1x32 indiquant les entités extraites de l'histogramme dans
l'espace hsv
% L'Histogramme dans l'espace de couleur HSV est obtenu utilisant une
% quantification par niveau:
% 8 pour H(hue), 2 pour S(saturation), et 2 pour V(Value).
% Le vecteur descripteur de taille 1x32 est calculé et normalisé

[rows, cols, numOfBands] = size(img);
% convertir l'image RGB en HSV.
img = rgb2hsv(img);

% Extraire les 3 composantes (espaces) h, s  v
h = img(:, :, 1);
s = img(:, :, 2);
v = img(:, :, 3);

% Chaque composante h,s,v sera quantifiée équitablement en 8x2x2
% le nombre de niveau de quantification est:
numberOfLevelsForH = 8; % 8 niveau pour h
numberOfLevelsForS = 2; % 2 niveau pour s
numberOfLevelsForV = 2; % 2 niveau pour v

% Il est possible de faire la quantification par seuillage. Les seuils sont
% extraits pour chaque composante comme suit:

% X seuils ==> X+1 niveaux
% thresholdForH = multithresh(h, numberOfLevelsForH-1);
% thresholdForS = multithresh(s, numberOfLevelsForS -1);
% thresholdForV = multithresh(v, numberOfLevelsForV -1);

% Quantification
% seg_h = imquantize(h, thresholdForH); % appliquer les seuils pour obtenir
une image segmentée...
% seg_s = imquantize(s, thresholdForS); % appliquer les seuils pour obtenir
une image segmentée...
% seg_v = imquantize(v, thresholdForV); % appliquer les seuils pour obtenir
une image segmentée...

% Trouver le maximum.
maxValueForH = max(h(:));
maxValueForS = max(s(:));
maxValueForV = max(v(:));

% Initialiser l'histogramme à des zéro de dimension 8x2x2
hsvColor_Histogram = zeros(8, 2, 2);


% Quantification de chaque composante en nombre niveaux étlablis
quantizedValueForH=ceil((numberOfLevelsForH .* h)./maxValueForH);
quantizedValueForS= ceil((numberOfLevelsForS .* s)./maxValueForS);
quantizedValueForV= ceil((numberOfLevelsForV .* v)./maxValueForV);
```

```
% Créer un vecteur d'indexes
index = zeros(rows*cols, 3);
index(:, 1) = reshape(quantizedValueForH',1,[]);
index(:, 2) = reshape(quantizedValueForS',1,[]);
index(:, 3) = reshape(quantizedValueForV',1,[]);

% Remplir l'histogramme pour chaque composante h,s,v
% (ex. si h=7,s=2,v=1 Alors incrémenter de 1 la matrice d'histogramme à la
position 7,2,1)
for row = 1:size(index, 1)
    if (index(row, 1) == 0 || index(row, 2) == 0 || index(row, 3) == 0)
        continue;
    end
    hsvColor_Histogram(index(row, 1), index(row, 2), index(row, 3)) = ...
        hsvColor_Histogram(index(row, 1), index(row, 2), index(row, 3)) + 1;
end

% normaliser l'histogramme à la somme
hsvColor_Histogram = hsvColor_Histogram(:)';
hsvColor_Histogram = hsvColor_Histogram/sum(hsvColor_Histogram);
end

function texture_features= textureFeatures(img)
% Basée sur l'analayse de textures par la GLCM (Gray-Level Co-Occurrence
% Matrix)
% Le vecteur de taille 1x4 contiendra [Contrast, Correlation, Energy,
% Homogeneity]
glcm = graycomatrix(rgb2gray(img),'Symmetric', true);
stats = graycoprops(glcm);
texture_features=[stats.Contrast, stats.Correlation, stats.Energy,
stats.Homogeneity];
texture_features=texture_features/sum(texture_features);
end

function shapeFeat= shapeFeatures(img)
% Basée sur les 7 momens de Hu
% Telecharger le code invmoments de
% https://ba-network.blogspot.com/2017/06/hus-seven-moments-invariant-matlab-
code.html
shapeFeat = invmoments(rgb2gray(img)); % 7 moments i=invariants de Hu
shapeFeat=shapeFeat/mean(shapeFeat);
end
```

5.1 Programme Matlab de classification/évaluation

```
%% Training à exécuter une seule fois
% Le dossier "BD2C" contient deux sous dossier "obj_car" et "obj_ship".
% "obj_car" contient 400 images de voitures
% "obj_ship" contient 90 images de bateaux
% La première étape consiste à créer une matrice de caractéristiques
% (features) avec leurs classes respectivement
isFEx = input('Voulez-vous extraire les caractéristiques? taper Y si
oui:','s');
```

```
if (isFEx=='Y')
    [features, etatFull]=createFeatures('DB2C');
end
% Partitionner la base aléatoirement en deux parties avec un taux
respectivement
% 80% pour l'apprentissage (training)
% 20% pour le test
P = cvpartition(etatFull,'Holdout',0.20);
svmStruct =
fitcsvm(features(P.training,:),etatFull(P.training),'Standardize',true,'Kernel
Function','rbf','KernelScale','auto');
[C,score] = predict(svmStruct,features(P.test,:));
errRate = sum(etatFull(P.test)~= C)/P.TestSize  %erreur de classification
conMat = confusionmat(etatFull(P.test),string(C)) % La matrice de confusion
TP=conMat(1,1);
TN=conMat(2,2);
FP=conMat(2,1);
FN=conMat(1,2);
Accuracy= 100*(TP+TN)/ (TP+TN+FP+FN) % le taux de classification
%% Extraction de caractéristiques
function [features, etat]=createFeatures(fold)
% fold: nom du dossier contenant les images par dossier (classe)
% features: Matrice de caractéristiques de taille (nombre_d_image x
nombre_features)
% etat: Matrice d'état de taille (nombre_d_image x 2). La première
% colonne contient le nom de la classe et le deuxième contient un id de
% classe 1 pour la première classe,2 pour la deuxième ...
lst=dir(fold);
k=1;
for i=3:length(lst)
    if (lst(i).isdir)
        sub=dir(strcat(lst(i).folder,'\',lst(i).name,'\*.jpg'));
        for j=1:length(sub)
            etat(k,1)=string(lst(i).name);
            %etat(k,2)=i-2;
            IDB=imread(strcat(lst(i).folder,'\',lst(i).name,'\',sub(j).name));
            features(k,:)=getFeatures(IDB, 50);
            k=k+1;
        end
    end
end
end
%% fonction pour créer le vecteur descripteur
function features = getFeatures(img, fsize)
features=zeros(fsize-1,1);
features=color_Moments(img);
features = [features, hsvHistogramFeatures(img)];
features = [features, textureFeatures(img)];
features = [features, shapeFeatures(img)];
end

function colorFeature = color_Moments(img)
% img: image à extraire les 2 premiers moments (mean et std) de chaque
composante R,G,B
% sorite: vecteur de dimension 1x6 contenants les caractéristiques
% Activer cette ligne si vous voulez travailler dans l'espace de couleur
% HSV qui est meilleur que RGB
```

```
%img=rgb2hsv(img);
% extract color channels
R = double(img(:, :, 1));
G = double(img(:, :, 2));
B = double(img(:, :, 3));
% compute 2 first color moments from each channel
colorFeature=[mean(R(:)), std(R(:)), mean(G(:)), std(G(:)), mean(B(:)),
std(B(:))];
colorFeature=colorFeature/mean(colorFeature);
end

function hsvColor_Histogram = hsvHistogramFeatures(img)
% img: image à quantifier dans un espace couleur hsv en 8x2x2 cases identiques
% sortie: vecteur 1x32 indiquant les entités extraites de l'histogramme dans
l'espace hsv
% L'Histogramme dans l'espace de couleur HSV est obtenu utilisant une
% quantification par niveau:
% 8 pour H(hue), 2 pour S(saturation), et 2 pour V(Value).
% Le vecteur descripteur de taille 1x32 est calculé et normalisé

[rows, cols, numOfBands] = size(img);
% convertir l'image RGB en HSV.
img = rgb2hsv(img);

% Extraire les 3 composantes (espaces) h, s  v
h = img(:, :, 1);
s = img(:, :, 2);
v = img(:, :, 3);

% Chaque composante h,s,v sera quantifiée équitablement en 8x2x2
% le nombre de niveau de quantification est:
numberOfLevelsForH = 8; % 8 niveau pour h
numberOfLevelsForS = 2; % 2 niveau pour s
numberOfLevelsForV = 2; % 2 niveau pour v

% Il est possible de faire la quantification par seuillage. Les seuils sont
% extraits pour chaque composante comme suit:

% X seuils ==> X+1 niveaux
% thresholdForH = multithresh(h, numberOfLevelsForH-1);
% thresholdForS = multithresh(s, numberOfLevelsForS -1);
% thresholdForV = multithresh(v, numberOfLevelsForV -1);

% Quantification
% seg_h = imquantize(h, thresholdForH); % appliquer les seuils pour obtenir
une image segmentée...
% seg_s = imquantize(s, thresholdForS); % appliquer les seuils pour obtenir
une image segmentée...
% seg_v = imquantize(v, thresholdForV); % appliquer les seuils pour obtenir
une image segmentée...

% Trouver le maximum.
maxValueForH = max(h(:));
maxValueForS = max(s(:));
maxValueForV = max(v(:));
```

```
% Initialiser l'histogramme à des zéro de dimension 8x2x2
hsvColor_Histogram = zeros(8, 2, 2);


% Quantification de chaque composante en nombre niveaux étlablis
quantizedValueForH=ceil((numberOfLevelsForH .* h)./maxValueForH);
quantizedValueForS= ceil((numberOfLevelsForS .* s)./maxValueForS);
quantizedValueForV= ceil((numberOfLevelsForV .* v)./maxValueForV);

% Créer un vecteur d'indexes
index = zeros(rows*cols, 3);
index(:, 1) = reshape(quantizedValueForH',1,[]);
index(:, 2) = reshape(quantizedValueForS',1,[]);
index(:, 3) = reshape(quantizedValueForV',1,[]);

% Remplir l'histogramme pour chaque composante h,s,v
% (ex. si h=7,s=2,v=1 Alors incrémenter de 1 la matrice d'histogramme à la
position 7,2,1)
for row = 1:size(index, 1)
    if (index(row, 1) == 0 || index(row, 2) == 0 || index(row, 3) == 0)
        continue;
    end
    hsvColor_Histogram(index(row, 1), index(row, 2), index(row, 3)) = ...
        hsvColor_Histogram(index(row, 1), index(row, 2), index(row, 3)) + 1;
end

% normaliser l'histogramme à la somme
hsvColor_Histogram = hsvColor_Histogram(:)';
hsvColor_Histogram = hsvColor_Histogram/sum(hsvColor_Histogram);
end

function texture_features= textureFeatures(img)
% Basée sur l'analayse de textures par la GLCM (Gray-Level Co-Occurrence
% Matrix)
% Le vecteur de taille 1x4 contiendra [Contrast, Correlation, Energy,
% Homogeneity]
glcm = graycomatrix(rgb2gray(img),'Symmetric', true);
stats = graycoprops(glcm);
texture_features=[stats.Contrast, stats.Correlation, stats.Energy,
stats.Homogeneity];
texture_features=texture_features/sum(texture_features);
end

function shapeFeat= shapeFeatures(img)
% Basée sur les 7 moments de Hu
% Télécharger le code invmoments de
% https://ba-network.blogspot.com/2017/06/hus-seven-moments-invariant-matlab-
code.html
shapeFeat = invmoments(rgb2gray(img)); % 7 moments i=invariants de Hu
shapeFeat=shapeFeat/mean(shapeFeat);
end
```