**Université Sidi Mohammed Ben Abdallah de Fès**

**Faculté des Sciences Dhar El Mahraz**

# Image mining

# Atelier N 1 : CBIR

Prof : My Abdelouahed Sabri

abdelouahed.sabri@gmail.com

# 1 Introduction

L'objectif de cet atelier est l'initier le lecteur aux systèmes de recherches basées sur le contenu (CBIR, Content Based Image retrieval). Le CBIR permet entre autre de rechercher les images similaires à une image requête dans une base d'images. Cette similarité est calculée non pas sur les images elles-mêmes mais par le biais d'une description (descripteur) utilisée par celui qui a conçu le système. La figure ci-dessous présente une vue globale d'un système CBIR.

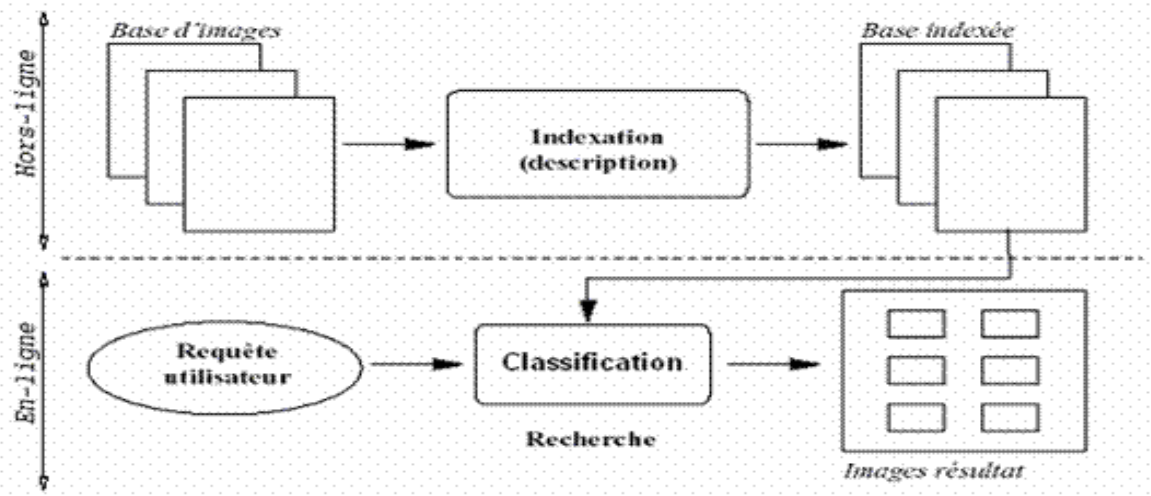



*Figure .1 Schéma fonctionnel d'un système de recherche d'images*

Un système CBIR est organisé en deux parties :

1-Phase hors ligne appelé souvent indexation où pour chaque image un vecteur descripteur sera extrait et sauvegarder sous forme de base d'indexes

2-Phase en ligne : c'est la recherche d'images similaires pour une image requête. Le même type de descripteur sera extrait de l'image requête et comparé avec la base d'indexes.

Deux grandes étapes sont à prendre en considération lors de l'élaboration de l'élaboration d'un système CBIR. Le premier est la nature des descripteurs (caractéristiques) à utiliser pour former la base d'indexes. Et le deuxième c'est la mesure de similarité à utiliser pour chercher les images les plus similaires pour une image requête. En effet la recherche ce fait par mesure de distance entre descripteurs au lieu des images.

Les descripteurs utilisés pour former la base d'indexes sont des descripteurs bas niveau, à savoir ; la forme, la couleur et la texture. Et la mesure de similarité utilisée peut être une simple distance euclidienne ou une projection multidimensionnelle.

## 2 Base d'images

Pour cet atelier nous allons utiliser la base d'images COREL qui contient 10800 images et qui est classée en 80 groupes d'images. Cette base est une parfaite pour comprendre les fondamentaux du CBIR.

## 3 Implémentation

Afin de développer un système CBIR robuste vis-à-vis des transformations que peut subirent les images, nous avons opté pour Matlab comme environnement de travail.

La première étape est l'indexation et le premier code à écrire est pour boucler sur les images de la base de données pour extraire de chacune un vecteur descripteur convenable.

Mon premier conseil est de copier la base d'images dans le dossier par défaut de Matlab afin d'éviter les erreurs d'accès. Mon deuxième est de ne pas écrire le code directement sur l'invite de commande mais plutôt créer un fichier *.m, le modifier et l'exécuter à fur et à mesure de l'avancement de cet atelier.

### 3.1 Lecture des images

L'objectif du premier code Matlab est de lire les images et afficher les 6 premières images de la base d'images

```
%% Affichage des 6 premières images
% lister les images contenues dans le dossier
list= dir('obj_decoys'); % retourne une structure
% pour vérifier le nombre de fichier, exécuter "length(list)"
% la taille est +2 ".." et "."
length(list)
% afficher le nom du fichier numéro 10
disp (list(12).name)
% afficher l'image
imshow(list(12).name)
% boucler et afficher les 6 premières images
for i=1:6
    subplot (3,2,i); imshow(list(i+2).name), title(list(i+2).name);
end
```

### 3.2 Recherche par image entière

Pour une première expérimentation d'un système CBIR, l'objectif est d'afficher les 5 images les plus similaires à l'image requête « Imrequest.jpg » en se basant sur un calcul de différence entre images entières.

Exécuter le fichier Matlab « CBIR_Basic.m ».

```
%%
% Sabri My Abdelouahed, 29-09-2018
% FSDM - USMBA, Fez, Morocco

%%
clc
%% CBIR brut
% lister les images contenues dans le dossier
% Nous allons travailler sur un dossier de la base choisi au hasard et qui
% contient les images de deux catégories différentes (juste pour tester)
list= dir('obj_decoys'); % retourne une structure
% list =dir('obj_decoys\*.jpg'); pour ne récuper que les jpg
% Lecture de l'image requête
n=length(list)-2; % -2 par ce list retourne aussi les .. et .
% boucler sur les toutes les images de la base et les comparer avec l'image
% requéte
% v est un vecteur qui va contenir le nombre de points similaires entre
% image requéte et chacune des images de la base
v=[];
for i=3:n
   IDB=imread(list(i).name);
   Ires=abs(diff(rgb2gray(IDB-Ireq)));
   [m,~]=find(Ires==0);
   v(i-2)=9480-size(m,1);
end
% charger l'image requéte dans Ireq
Ireq=imread('Imrequest.jpg');
% Trier le vecteur v pour ne tenir en compte que les 5 premiers éléments
vsort=sort(v,'ascend');
% afficher l'image requête et les 5 images similaires par ordre de similarité
figure;
subplot (3,2,1); imshow(Ireq), title('Image requéte');
for i=1:5
    [aa,bb,~]=find(v==vsort(i));
    subplot (3,2,i+1); imshow(list(bb(1)+2).name), title(list(bb(1)+2).name);
end
```

Le résultat de recherche doit être comme sur la figure ci-dessous


*Figure .2 Résultat de recherche utilisant une similarité sur l'image complète*

La première image trouvée est identique à l'image requête ce qui montre que le système développer a pu répondre à la requête correctement alors que les autres images ne sont pas visuellement très similaires. L'autre inconvénient et qui est majeur et qu'il est long en termes de temps de recherche et très gourment en mémoire. Aussi, il n'est pas du tout robuste vis-à-vis des transformations sur l'image. Pour exemple vous pouvez roter l'image requête et vous allez voir que le résultat est complétement erroné.

L'idéal est de ne pas chercher sur la totalité des images mais travailler avec des caractéristiques (descripteurs) de chaque image. 3 types de caractéristiques sont les plus utilisés dans un système CBIR : la Couleur, la Forme et la Texture.

Par la suite, nous allons présenter et utiliser chacune des 3 caractéristiques pour créer un système CBIR performant et puissant.

## 3.3 Recherche par Couleur

Dans cette partie, la caractéristique couleur sera utilisée pour tester notre système. Contrairement à la recherche par similarité sur toute l'image qui consiste à calculer la différence entre l'image requête et toutes les images de la base à chaque recherche, le CBIR consiste dans sa partie hors-ligne de créer une base d'index à base des caractéristiques des images de la base et la recherche est

effectuée par similarité entre caractéristiques de l'image requête et caractéristiques de la base d'index. Tout cela nous ramène à dire que la première étape est d'extraire les caractéristiques couleurs des images de la base et l'enregistrer afin de l'utiliser dans chaque recherche.

Le vecteur descripteur couleur à utilise est constitué des valeurs des moments statistiques ; moyenne et variance de chaque composante couleur R, G et B. La taille du vecteur sera donc 6.

Le code dans le fichier « **CBIR_Color.m** » permet en un premier temps d'indexer la base d'image et après de lancer la recherche pour une image requête et d'afficher les 5 images les plus similaires.

Le code Matlab contient X parties :

1-La partie principale

2-La partie d'indexation. La fonction « CBIR_Indexation » est appelée avec comme paramètre la taille du vecteur descripteur. Cette partie ne doit être exécutée qu'une seule fois

3-La partie de recherche. La fonction « CBIR_ Recherche » est appelée avec comme paramètres l'image requête, la matrice des descripteurs et la liste des images.

Le vecteur descripteur est créé par la fonction « getFeatures » qui a comme arguments d'entrée l'image et la taille du vecteur. C'est cette fonction qu'on va modifier pour ajouter d'autres descripteurs par la suite. Cette fonction appelle la fonction « color_Moments » qui va retourner le les 6 moments de couleurs.

```
%%
% Sabri My Abdelouahed, 29-09-2018
% Master WISD-FSDM-USMBA, Fez, Morocco
%%
% Fichier CBIR_Color.m
clc
%% CBIR Color
fsize=7; % 6  features pour colorMoments + 1 pour sauvegarder l'indice de
l'image
%% Indexation de la base de données: Exécutée une seule fois
index = input('Voulez vous lancer l''indexation? taper Y si oui:','s');
if (index=='Y')
    [features, Image_names]=CBIR_Indexation(fsize) ;
end
%% Processus en ligne: Recherche
% Lecture de l'image requête
Ireq=imread('Imrequest1.jpg');

%% Recherche de 5 images les plus similaires
CBIR_Recherche(Ireq,features, Image_names);

%% Fonctions de recherche
```

```
function CBIR_Recherche(Ireq,features, Image_names)
% Ireq: Image requéte
% features: Matrice des indexes
% Image_names: liste des noms des images
disp('Recherche...');
% extraire le vecteur descripteur
[~, fsize]=size(features);
feature_req=getFeatures(Ireq, fsize);
% calculer la distance euclidienne à la matrice de caractéristiques
Distance(:,1) = pdist2(features(:,1:fsize-1),feature_req,'euclidean');
Sorted_Distance=sort(Distance,'ascend');
% Affichage des images similaires par ordre de similarité
figure;
subplot (3,2,1); imshow(Ireq), title('Image requéte');
for i=1:5
    [aa,bb,~]=find(Distance==Sorted_Distance(i));
    subplot (3,2,i+1); imshow(imread(char(Image_names(aa(1))))),
title(char(Image_names(aa(1))));
end
end
%% Fonction d'Indexation
function [features, Image_names]=CBIR_Indexation(fsize)
%
% features: matrice des caractéristiques
% Image_names: Nom des images de la base

% lister les images contenues dans le dossier
list= dir('obj_decoys'); % retourne une structure
% Lecture de l'image requéte
n=length(list)-2;
% v est un vecteur qui va contenir le nombre de points simlaires entre
% image requéte et chacune des images de la base
features=zeros(n-2,fsize);
% boucler sur les toutes les images de la base et les comparer avec l'image
% requéte
% Indexation
disp('Debut d''Indexation')
for i=3:n
    IDB=imread(strcat(list(i).folder,'\',list(i).name));
    features(i-2,:)=[getFeatures(IDB, fsize), i-2];
    filename(i-2)=string(list(i).name);
end
Image_names=filename';
disp('Fin d''Indexation');
end

function features = getFeatures(img, fsize)
% fonction pour créer le vecteur descripteur
features=color_Moments(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
```

Le résultat de recherche de deux images est donnée sur par les figures suivantes :

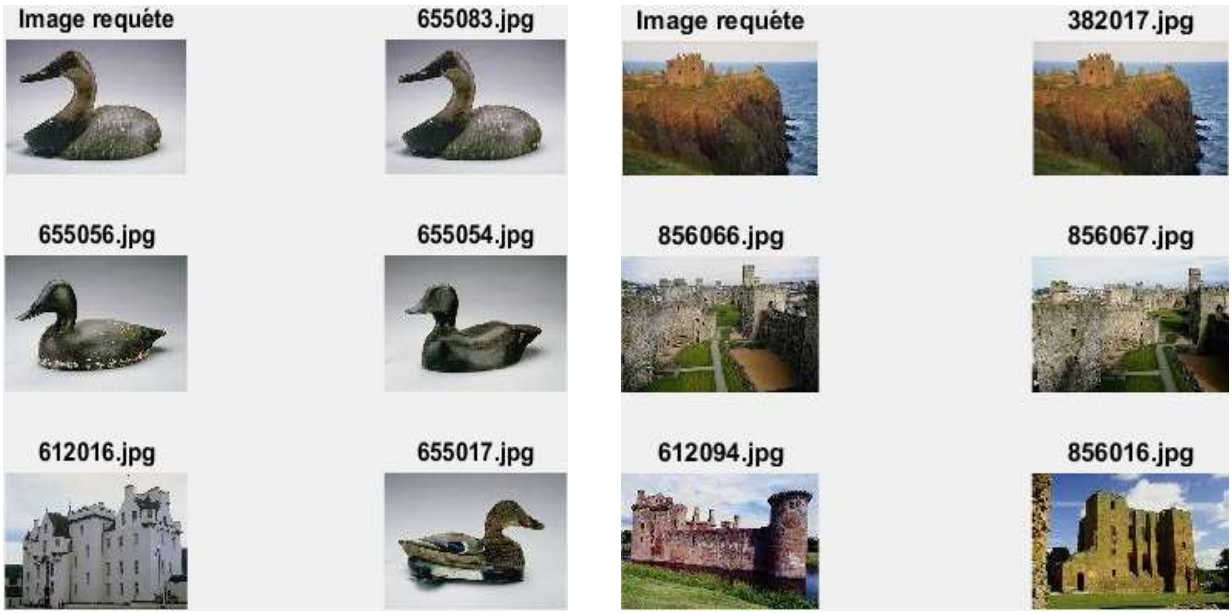



*Figure .3 Résultat de recherche de deux images requêtes utilisant la caractéristique couleur*

L'utilisation des moments statistiques semblent donner de bons résultats que la recherche par image entière. Reste à tester la robustesse vis-à-vis des transformations comme la rotation, la translation et le changement d'échelle.

## 3.4 Recherche par Histogramme

L'histogramme est souvent utilisé pour regrouper le nombre de pixels par intensité. L'histogramme peut être élaboré sur n'importe quel espace couleur ; RGB, HSV, … et même à niveau de gris. Des recherches scientifiques ont montré que l'espace de couleur HSV est le plus adapté à ce type de système de recherche. L'histogramme est un vecteur dont la taille est égale au nombre de valeur

d'intensité dans l'image. L'idée est de quantifier l'histogramme en un nombre limité. La taille du vecteur descripteur a été largement discutée et la taille adopté est 32 caractéristiques. Une quantification par niveau pour chaque composante H, S et V est élaborée. Pour la composante H (hue) le niveau choisi est 8, pour la composante S (saturation) le niveau est 2 et pour la composante V (value) le niveau est 2.

Modifier le code Matlab pour permettre d'avoir un système CBIR utilisant un vecteur descripteur de taille 1x38 composé de couleur et d'histogramme. Il faut modifier la valeur de « fsize » à 39 pour avoir un vecteur de 38 descripteurs (6 pour les moments et 32 pour l'histogramme). Il faut aussi ajouter la fonction « hsvHistogramFeatures » qui admet comme argument d'entrer une image et qui retourne le vecteur descripteur créé à partir de l'histogramme. La fonction « getFeatures » doit être modifiée pour concaténer les deux vecteurs descripteurs de moments et d'histogramme.

```
%%
% Sabri My Abdelouahed, 29-09-2018
% FSDM - USMBA, Fez, Morocco
%%
function features = getFeatures(img, fsize)
% fonction pour créer le vecteur descripteur
features=zeros(fsize-1,1);
if (fsize>=7)
    features=color_Moments(img);
end
if (fsize>=39)
    features = [features, hsvHistogramFeatures(img)];
end
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
```

Les résultats d'indexation recherche par le nouveau descripteur est présenté sur la figure ci-dessous.


*Figure .4 Résultat de recherche de deux images requêtes utilisant les caractéristiques : couleur et histogramme*

Si on compare les résultats de recherche utilisant seulement la caractéristique couleur avec les résultats de recherche utilisant en plus la caractéristique de l'histogramme HSV on remarque une grande amélioration. Reste que dans le cas du château, les résultats présentés sont comparable en terme de couleur mais pas en terme de représentation ni de forme. C'est pour cette raison, qu'il faut introduire d'autres caractéristiques pour affiner la recherche.

## 3.5 Recherche par texture

La texture est une information très importante et toujours présente dans les images. Elle est très utilisée dans les systèmes CBIR. Dans cette partie nous allons créer un nouveau descripteur de texture de taille 4 et qui va contenir le contraste, la corrélation, l'énergie et l'homogénéité. Ces 4 informations sont très utilisées dans ce type de système et qui sont des mesures statistiques extraites de la matrice de cooccurrence de niveau de gris (GLCM, Gray Level Co-occurrence Matrix) également appelée matrice de dépendance spatiale de niveau de gris. Les fonctions GLCM caractérisent la texture d'une image en calculant la fréquence à laquelle des paires de pixels avec des valeurs spécifiques et dans une relation spatiale spécifiée se produisent dans une image, en créant un GLCM, puis en extrayant des mesures statistiques de cette matrice.

En ajoutant la caractéristique texture aux deux autres caractéristiques, nous aurons un vecteur descripteur de taille 1x42.

Modifier le code Matlab pour permettre d'avoir un système CBIR utilisant un vecteur descripteur de taille 42 composé de couleur et d'histogramme. Il faut modifier la valeur de « fsize » à 43 pour avoir un vecteur de 38 descripteurs (6 pour les moments, 32 pour l'histogramme et 4 pour la texture). Il faut aussi ajouter la fonction « textureFeatures » qui admet comme argument d'entrer une image et qui retourne le vecteur descripteur créé à partir de la texture. La fonction « getFeatures » doit être modifiée pour concaténer les deux vecteurs descripteurs de moments et d'histogramme.

```
%%
% Sabri My Abdelouahed, 29-09-2018
% FSDM - USMBA, Fez, Morocco
function features = getFeatures(img, fsize)
% fonction pour créer le vecteur descripteur
features=zeros(fsize-1,1);
if (fsize>=7)
features=color_Moments(img);
end
if (fsize>=39)
features = [features, hsvHistogramFeatures(img)];
end
if (fsize>=43)
features = [features, textureFeatures(img)];
end
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
```

Les résultats d'indexation recherche par le nouveau descripteur est présenté sur la figure ci-dessous.


*Figure .5 Résultat de recherche de deux images requêtes utilisant les caractéristiques : couleur, histogramme et texture*

En peut remarquer de la figure ci-dessus que l'ajout de l'information texture a amélioré le résultat de recherche pour le cas de l'image du château qui est une image très texturée.

## 3.1 Recherche par Forme

Le troisième type de caractéristiques utilisé dans les systèmes CBIR est la forme. Ainsi, notre 5éme implémentation aura comme objectif de créer un vecteur descripteur de forme et le concaténer avec les 3 caractéristiques utilisées auparavant pour voir son impact sur le résultat de recherche. Le vecteur descripteur de forme de taille 1x7 utilisé sera créé à partir des moments invariants de Hu.

Modifier le code Matlab pour permettre d'avoir un système CBIR utilisant un vecteur descripteur de taille 49 composé de couleur et d'histogramme. Il faut modifier la valeur de « fsize » à 50 pour avoir un vecteur de 38 descripteurs (6 pour les moments, 32 pour l'histogramme, 4 pour la texture et 7 pour la forme). Il faut aussi ajouter la fonction « shapeFeatures » qui admet comme argument d'entrer une image et qui retourne le vecteur descripteur créé à partir de la forme. La fonction « getFeatures » doit être modifiée pour concaténer les deux vecteurs descripteurs de moments et d'histogramme.

```
function features = getFeatures(img, fsize)
% fonction pour créer le vecteur descripteur
features=zeros(fsize-1,1);
features = shapeFeatures(img);
return;
```

```
if (fsize>=7)
features=color_Moments(img);
end
if (fsize>=39)
features = [features, hsvHistogramFeatures(img)];
end
if (fsize>=43)
features = [features, textureFeatures(img)];
end
if (fsize>=50)
  features = [features, shapeFeatures(img)];
end
end
function shapeFeat= shapeFeatures(img)
% Basée sur les 7 momens 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 invariants de Hu
shapeFeat=shapeFeat/mean(shapeFeat);
end
```

Les résultats d'indexation recherche par le nouveau descripteur est présenté sur la figure ci-dessous.


*Figure .6 Résultat de recherche de deux images requêtes utilisant les caractéristiques : couleur, histogramme, texture et forme*

## 3.2 Etude comparative

Pour bien comprendre l'effet de chacun des descripteurs utilisé dans le système CBIR développé, la figure ci-dessous présente les résultats du CBIR utilisant chaque descripteur séparément de chaque descripteur. On peut constater que le choix du descripteur dépend du type d'image à chercher.

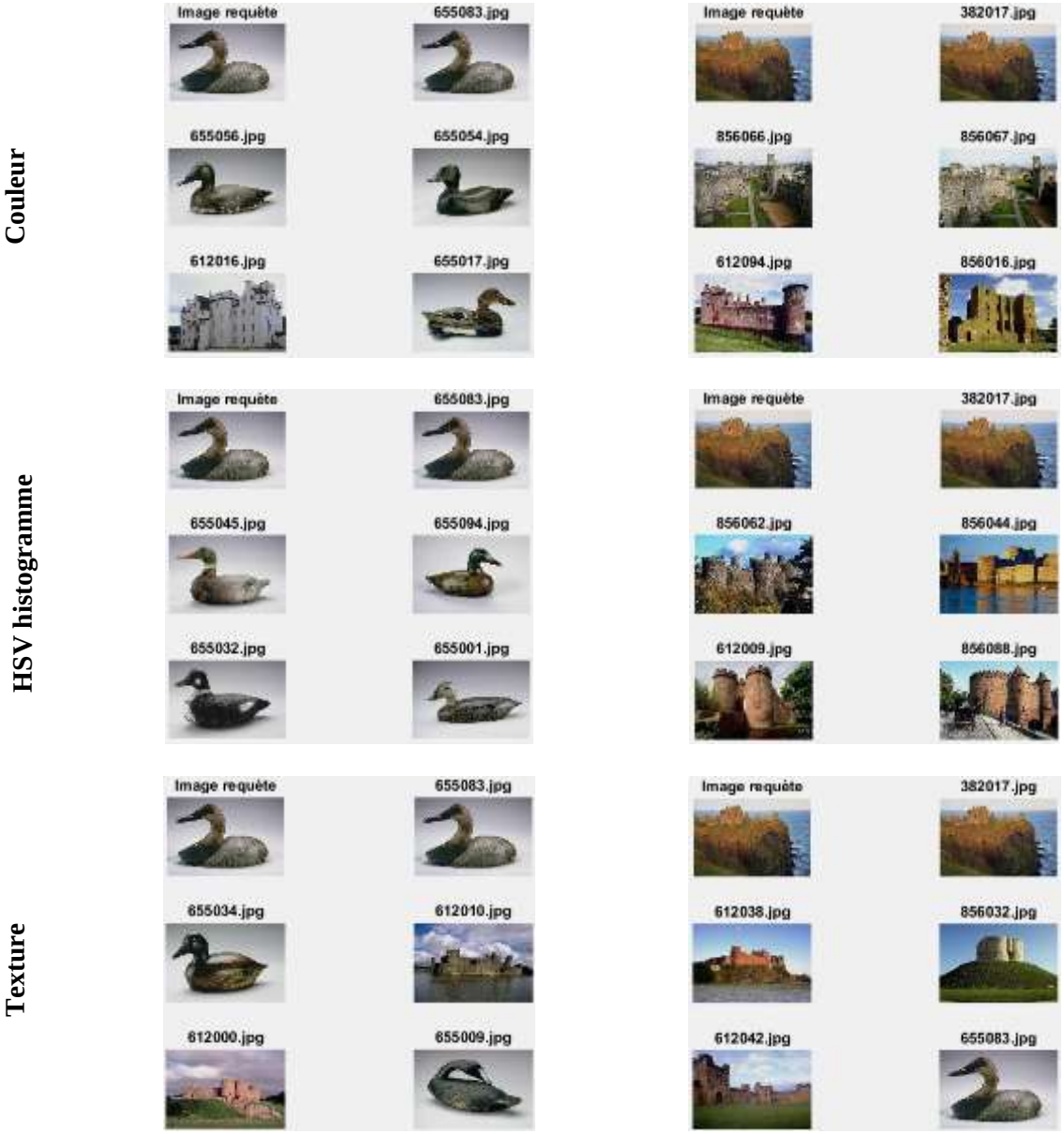



*Figure .7 Comparatif des résultats du ssystème CBIR avec différentes caractéristiques pour deux images choisies aléatoirement de la base : Couleur, Histogramme, Texture ou/et Forme.*

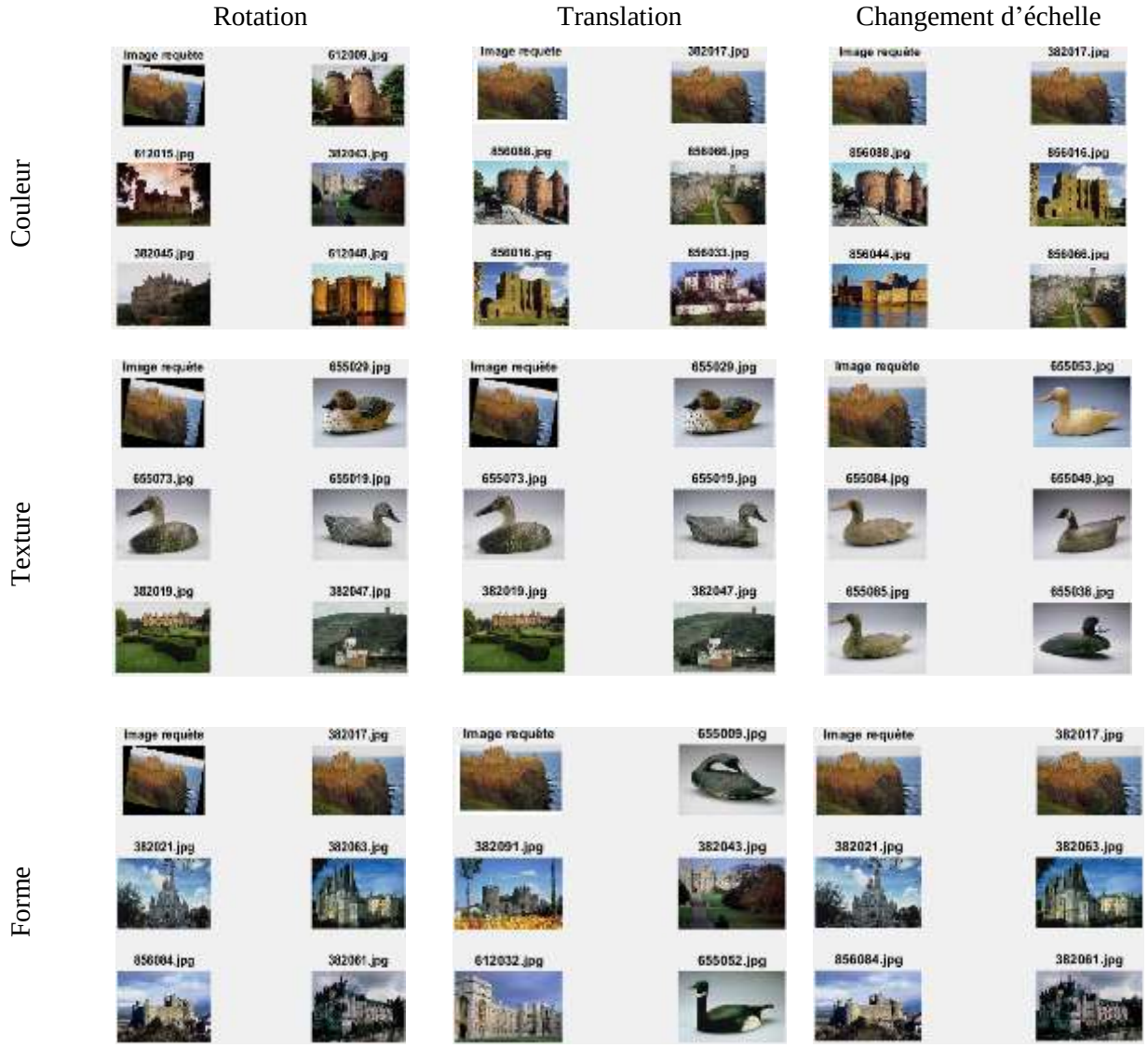
On remarque que la performance du système CBIR développé ici dépend de la nature de l'image. Ainsi, pour le cas de bases d'image couleurs les descripteurs de couleurs sont très performants et pour des bases d'image texturées les descripteurs de textures sont à utiliser.

En ce qui concerne la forme, dans les deux exemples présentés on peut remarquer que l'effet n'est pas très important et ceci est dû aux types d'images de la base. Les descripteurs de forme sont très

utiles dans le cas où les images ont subi des transformations géométriques comme sera montré dans l'étude de la section suivante.

### 3.3 Robustesse vis-à-vis des transformations géométriques

Le robustesse des systèmes CBIR est testée en présence des transformations géométriques ; à savoir la translation, la rotation et le changement d'échelle. La figure ci-dessous présente le résultat de recherche utilisant tous les descripteurs en présence de transformations géométriques.



Couleur + Texture + Forme


*Figure .8 Résultat de recherche en présence de rotation, changement d'échelle et translation*

Les résultats présents sur la figure ci-dessus montrent que chaque descripteur présente une faiblesse et une robustesse vis-à-vis d'une transformation géométrique. Ainsi, la couleur est robuste en présence de changement d'échelle et de translation mais pas robuste en présence de rotation. La texture seule n'est pas robuste à aucune des transformations géométriques. Par contre, la forme est robuste à la rotation et au changement d'échelle. L'utilisation des 3 descripteurs permet une forte robustesse en présence de translation et de changement d'échelle et un peu moins pour la rotation.