

# Rapport d'indexation de document multimédia

Nom et prénom : Vu Hong Thuan

## Introduction

Ce TP a pour l'objectif de faire comprendre l'application de ACP dans la reconnaissance de visage. Le travail pratique est réalisé sur l'outil Octave. La base de teste pour la reconnaissance de visage est la base de visage ORL.

## Réponse des questions et résultats

### COMPRENDRE LE CODE FOURNI

1. On cherche à appliquer une ACP sur ces données, dans le but de décrire au mieux les différents visages dans un objectif d'identification.

Questions à repondre :

- Quels sont les individus ? Les images sont des individus pour ce système.
- Combien d'individus y a-t-il dans la base ? Dans la base ORL, il y a 400 individus de 40 classes.
- Quels sont les variables initiales ? Les variables initiales sont les valeurs de niveau de gris de chaque pixel des images.
- Combien de variables initiales y a-t-il dans la base ? Le nombre de pixels est  $92 \times 112 = 10304$  pixels

2. Consultez le contenu des fichiers ACP.m et createBD.m. Chacun de ces fichiers correspond à la fonction Matlab de même nom. Dans chacun de ces fichiers, les variables d'entrée sont expliquées mais pas les variables de sortie. Lisez attentivement le code.

Questions à repondre :

- A quoi correspondent les variables de sortie de la fonction ACP (vous pouvez vous aider du cours d'Annie Morin) ?

Les variables de sortie de la fonction ACP :

**function [A,C,lambda,BDbarre]=ACP(BD)**

- A : les axes principaux
  - C : la projection des éléments sur l'axe principal. Ce sont des composants principaux attachés aux axes principaux.
  - lambda : les valeurs propres
  - BDbarre : centre de gravité des images entrées
- A quoi correspondent les variables de sortie de la fonction CreateBD ?  
**function [BD,names,cl,nrow,ncol]=CreateBD(path,liste,ncl)**
    - BD : liste des valeurs de niveau de gris des images
    - names : liste de nom des éléments de BD
    - cl : liste de classe correspondant avec les éléments de BD
    - nrow : nombre de ligne (hauteur) d'une image
    - ncol : nombre de colonne (largeur) d'une image

3. Question à répondre : Dans le fichier ACP.m, pourquoi l'attribut BDbarre a-t-il été spécifié comme variable de sortie ?

- Pour calculer de nouvelles coordonnées des images de test, nous devons utiliser l'attribut Bdbarre. Pour chaque image de teste, nous devons faire la soustraction avec l'image moyenne.

4. Question à répondre : Dans le fichier CreateBD.m, à quoi correspond la ligne de code suivante : `BD(k,:)=reshape(img,1,nrow*ncol);` ?

- Pour convertir d'une matrice de nrow x ncol à d'une matrice 1 x (nrow\*ncol) (une dimension). Et nous devons assigner à un élément de liste BD.

5. Question à répondre : Dans le fichier ACP.m, à quoi correspond la ligne de code suivante : `slambda=cumsum(lambda)/sum(lambda);` ?

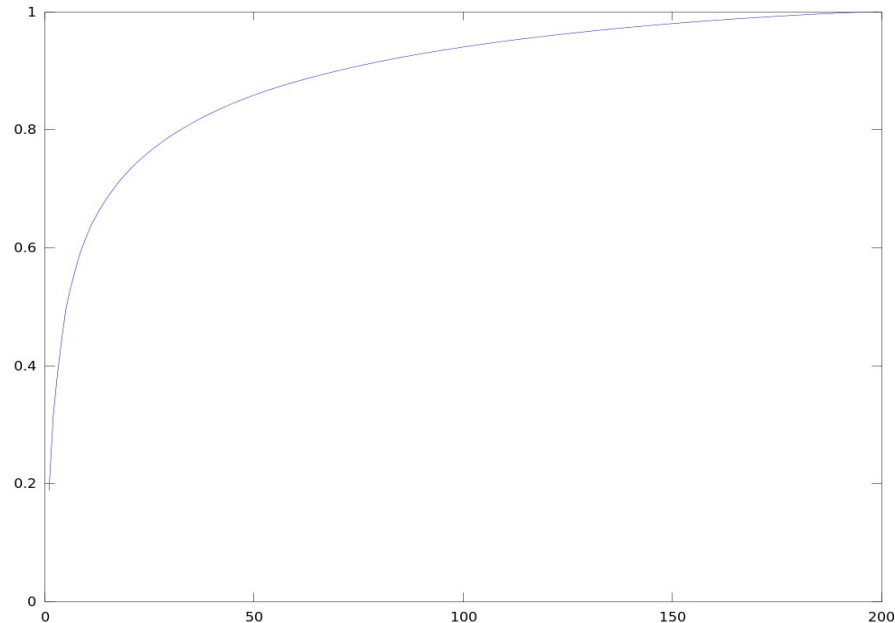
- Pour calculer la valeur accumulée de l'inertie cumulée par les vecteurs propres. Ce sont le pourcentage de description des données en utilisant les axes principaux.

6. Question à répondre : Dans le fichier ACP.m, à quoi correspond la ligne de code suivante : `C=X*A;` ?

- Calculer les nouvelles coordonnées des images sur la nouvelle coordonnées des axes principaux. Ce sont des composants principaux.

7. Questions à répondre :

- Combien d'axes principaux (eigenfaces) vous proposez-vous de conserver pour l'identification de visages (aidez-vous du graphe des taux d'inertie cumulés) ?



Je vais choisir environ 75 axes principales pour l'identification de visages. Parce que le taux d'inertie cumulé est plus grand que 90% à partir de 75 axes principales.

- Quel était le nombre d'attributs initiaux ?  
10304 attributs
- Quel est le nombre d'attributs après application de l'ACP ?

- 199 attributs, et je choisis 75 attributs parmi eux
- Quel est par conséquent le pourcentage de réduction de la dimensionnalité du problème ?  
 $75/10304=0.007$

## MANIPULER LES EIGENFACES

Image moyenne de base de l'apprentissage



Image moyenne de base d'apprentissage.




Question à répondre : regardez les 5 premières eigenfaces et expliquez à quoi elles correspondent (c'est-à-dire l'information qu'elles portent).



Eigenface 1: Cette image implique des valeurs de cheveux, un peu des chemises, un peu des yeux, des bouches, et le fond. Ce sont des points en noir et blanc de cette image. Dans cette image, les valeurs sont très différents : la partie blanche et noire.



Eigenface 2 : Cette image implique les cheveux, les yeux, le chemiser, un part de front, un peu de bouche. Ce sont des éléments importants aussi pour reconnaître des visage.

	<p>Eigenface 3 : Cette image implique des yeux, le peau de visage. Ce sont des parties noires.</p>
	<p>Eigenface 4 : Cette image implique la lumière à gauche et la sombre à droit. Ce ne sont pas des caractères de visage.</p>
	<p>Eigenface 5 : Cette image implique des barbes, des lunettes. Ce ne sont pas plus des caractères de visage. Et la contraste diminue.</p>

## **CREATION ET CLASSIFICATION DE LA BASE DE TEST**

1. Question à répondre : Quelle préparation devez-vous faire subir aux données de test avant de les projeter ?

Pour chaque image, nous devons :

- Lire des valeurs de pixels
- Faire la soustraction avec l'image moyenne.

2. Résultats demandés : dans votre rapport, donnez les 5 premières valeurs de coefficients pour les 10 premiers exemples de test.

Voici les 5 premières valeurs de coefficients pour les 10 premiers exemples de test.

2398.520    1319.827    -335.138    -537.118    -1822.050

2228.177	1030.675	-1227.850	120.448	-836.174
2658.670	848.279	-1417.360	-1785.031	760.231
2518.395	1451.937	-926.578	-2146.460	-14.401
2480.970	1396.923	-1233.333	-1350.658	66.231
469.890	-353.558	-1401.712	1047.248	-457.578
1416.991	-579.623	-748.805	-79.298	35.129
1135.515	-534.848	-1175.712	224.338	79.232
875.945	-1001.060	-843.097	1064.617	-301.820
1345.080	-399.081	-907.625	316.917	-299.957

3. Dans le fichier `appel.m`, comparez les coefficients de projection de la base de test à ceux de la base d'apprentissage en utilisant l'algorithme du 1 plus proche voisin et la distance Euclidienne.

Résultats demandés : dans votre rapport, donnez les plus proches voisins des exemples numéro 6 à 10 de la base de test

s2\_6 a pour plus proche voisin : s2\_2  
s2\_7 a pour plus proche voisin : s2\_3  
s2\_8 a pour plus proche voisin : s2\_3  
s2\_9 a pour plus proche voisin : s2\_4  
s2\_10 a pour plus proche voisin : s2\_3

## EVALUATION DES PERFORMANCES

1. Dans le fichier `appel.m`, calculez la matrice de confusion associée à vos données et appelez-la `matConf`. dans votre rapport, donnez la matrice de confusion pour les exemples numéro 11 à 15 de la base de test

4	0	0	0	1
0	5	0	0	0
0	0	5	0	0
0	0	0	4	0
0	0	0	0	5

2. Dans le fichier `appel.m`, calculez, pour chaque classe, la précision et le rappel correspondant.

Résultats demandés : Dans votre rapport, donnez pour chaque classe, la précision et le rappel correspondants.

	Precision	Rapel
s1 :	0.833333	1.000000
s2 :	0.714286	1.000000
s3 :	1.000000	1.000000
s4 :	1.000000	1.000000
s5 :	0.666667	0.800000
s6 :	1.000000	1.000000
s7 :	1.000000	1.000000
s8 :	1.000000	1.000000
s9 :	1.000000	0.800000
s10 :	1.000000	0.800000
s11 :	1.000000	0.800000
s12 :	1.000000	1.000000
s13 :	1.000000	1.000000
s14 :	1.000000	0.800000
s15 :	0.714286	1.000000
s16 :	1.000000	0.800000
s17 :	0.333333	0.400000
s18 :	1.000000	1.000000
s19 :	1.000000	0.800000
s20 :	1.000000	0.800000

s21 :	0.833333	1.000000
s22 :	0.833333	1.000000
s23 :	1.000000	0.800000
s24 :	0.625000	1.000000
s25 :	0.833333	1.000000
s26 :	1.000000	1.000000
s27 :	1.000000	0.400000
s28 :	1.000000	0.800000
s29 :	0.833333	1.000000
s30 :	1.000000	1.000000
s31 :	1.000000	0.800000
s32 :	1.000000	0.600000
s33 :	1.000000	1.000000
s34 :	1.000000	1.000000
s35 :	1.000000	0.800000
s36 :	0.750000	0.600000
s37 :	0.833333	1.000000
s38 :	0.555556	1.000000
s39 :	1.000000	0.800000
s40 :	0.750000	0.600000

3. Dans le fichier appel.m, calculez, le taux de reconnaissance global (non pondéré). Résultat demandé : Dans votre rapport, donnez le taux de reconnaissance global sur votre base de test.  
Le taux de reconnaissance global sur votre base de test : 0.87500 (87%)

4. Refaites les expérimentations précédentes en inversant votre base d'apprentissage et votre base de test. Question à répondre : le taux de reconnaissance obtenu est-il le même qu'à la question précédente ? Expliquez votre résultat.

Le taux de reconnaissance global sur votre base de test : 91%. Après avoir inversé la base d'apprentissage et la base de test, nous obtenons un meilleur résultat. Le changement de base de test et base d'apprentissage nous donne des résultats différents. Cela n'implique pas quelle base est meilleure.

5. Question à répondre : Donnez les avantages et les inconvénients de la méthode des eigenfaces, utilisée dans ce TP. Selon vous, comment peut-on améliorer la qualité des résultats de ce système (et non pas le temps d'exécution) ? Donnez quelques (2-3) idées d'améliorations possibles.

Les avantages :

- Réduire le nombre des attributs. Donc, réduire les calculs.
- Simple, et facile à implémenter

Les inconvénients







- Ne marchera pas avec des changements forts des visages tels que l'angle de tourne. Avec la variance des exemples, ACP nous donne des résultats comme au hasard.
- Ne considère pas des attributs typiques de visages.

Amélioration

- Il faut éviter des éléments moins importants des visages tels que le fond, les cheveux, etc.
- Il faut concentre des éléments importants des visages : yeux, nez, bouche. Donc, nous pouvons comparer des éléments séparément.

## QUESTIONS OPTIONNELLES

1. Dans le fichier appel.m, rajoutez quelques lignes de code pour afficher sous forme d'image la reconstruction du visage TDTP2/s1/1.pgm avec q eigenfaces, où q varie de 1 à k.

		
Image originale	Image générée 1	Image générée 20
		
Image générée 40	Image générée 60	Image générée 75

Au début, nous avons une image 1 qui est très ressemblant avec l'image moyenne présentée dans la partie précédente. De pas en pas, nous avons des images proches avec l'image originale.

2. Dans le fichier appel.m, calculez la courbe CMC associée à vos données, pour les 20 premiers rangs.

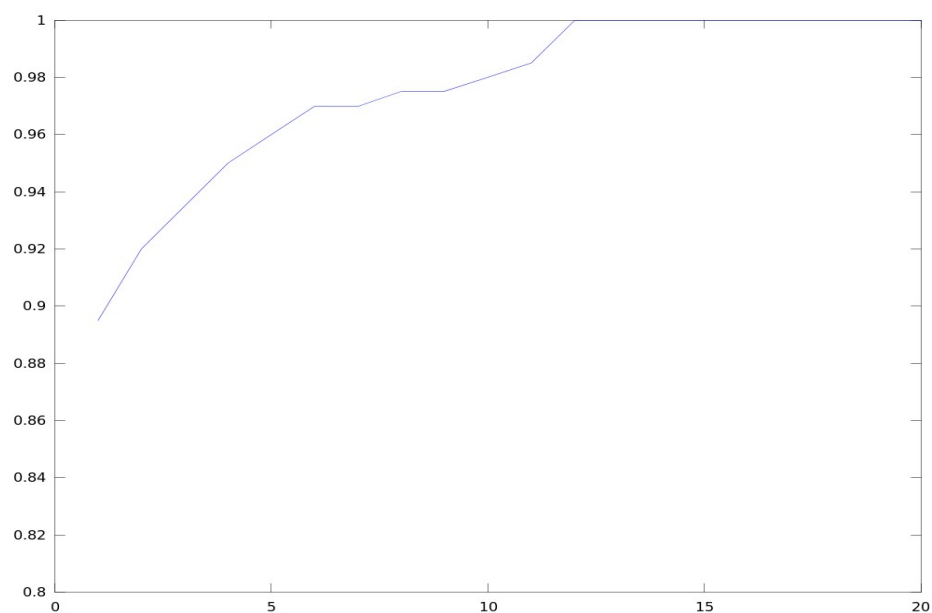
Le nombre de corrects exemples à chaque range de 1 à 20 est :

```
179 5 3 3 2 2 0 1 0 1 1 3 0 0 0 0 0
0 0 0
```

Le pourcentage accumulé de taux de reconnaissance de chaque range est :

```
0.89500 0.92000 0.93500 0.95000 0.96000 0.97000 0.97000 0.97500
0.97500 0.98000 0.98500 1.00000 1.00000 1.00000 1.00000 1.00000
1.00000 1.00000 1.00000 1.00000
```

Le pilot de CMC



a) Question à répondre : Quel est le taux de reconnaissance au rang 1 ? **89.5%**

b) Question à répondre : Si l'on suppose que, dans notre application, un opérateur humain est chargé de détecter parmi les 10 premiers visages renvoyés par l'algorithme lequel est le bon, quel taux de reconnaissance peut-on atteindre ?

Le taux de reconnaissance sera peut-être **98%**.

## Conclusion

Dans ce TP, nous avons implémenter ACP pour la reconnaissance de forme. Cela nous donne des résultats acceptables.

## Annexe

Fichier eigsort.m

```
function [Vsort,Dsort] = eigsort(M)

%%
%% Obtenir et trier des vecteurs propres et les valeurs propres
%%
%% Entree:
%% M: matrice entree des elements
%% Sortie
%% Vsort : Les vecteurs propres trie
%% Dsort : Les valeurs propres trie
[VectP,ValP]=eig(M);
eigvals = diag(ValP);
% Tri des valeurs propres descendant.
% Stockage de ces vp dans le vecteur colonne lambda.
[lambda,index] = sort(eigvals,'descend');
Dsort = lambda;
% Tri des vecteurs propres selon l'index
n = length(lambda);
Vsort = zeros(n,n);
for i=1:n
Vsort(:,i) = VectP(:,index(i));
end;
```

Fichier CreateBD.m

```
function [BD,names,cl,nrow,ncol]=CreateBD(path,liste,ncl)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Explication des variables d'entre :
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% path : chemin du rpertoire att_faces
%% liste : vecteur des images utilises pour construire la base. Si, par
%% exemple, liste = [1;2;3;4;5;6;7] les images 1.pgm 7.pgm de chacun des
%% rpertoires s1 s40 seront mises dans cette Base
%% Nombre de classes. Par dfaut, sur la base ORL, ncl=40.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Explication des variables de sortie :
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BD : liste des valeurs des images
%% names : liste de nom des éléments de BD
%% cl : liste de classe correspondant avec les éléments de BD
%% nrow : nombre de ligne (hauteur) d'un élément de BD
%% ncol : nombre de colonne (largeur) d'un élément de BD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k=0;
for (i=1:ncl)
dirname=strcat(path,'/s',num2str(i,2));
for (j=1:size(liste))
k=k+1;
filename=strcat(dirname,'/',num2str(liste(j),2),'.pgm');
img=imread(filename);
[nrow ncol]=size(img);
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% rearrange des images a le matrice une dimension,
%% attacher des ID de classe
%% attacher des noms des images et des classes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% assigner la valeur de matrice d'image à un élément de la base de données
BD(k,:)=reshape(img,1,nrow*ncol);
% attache le ID de classe correspondante à cet élément
cl(k)=i;
% attache le nom de classe correspondante et le nom de cette image à cet élément
names{k}=strcat('s',num2str(i,2),'_',num2str(liste(j),2));
end
end

```

Fichier ACP.m

```

function [A,C,lambda,BDbarre]=ACP(BD)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Explication des variables d'entre :
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BD : tableau de donnees
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Explication des variables de sortie :
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% A : les axes principaux
%% C : la projection des éléments sur les axes principaux, composants principaux
%% lambda : les valeurs propres
%% BDbarre : centre de gravite des images entree
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% n nombre d'individus, p nombre de variables
[n p]=size(BD); %--
%Calcul de X (BD centre)
BDbarre=mean(BD);
X=double(BD)-ones(n,1)*BDbarre; %--
% Calcul de Stilde : matrice des variances-covariances
% On ne peut pas calculer directement la matrice des
% variance-covariance S=cov(X) car le calcul est trop instable et
% demande trop de mmoire (car le nombre d'attributs est trop
% important compar au nombre d'individus) !
% On va donc utiliser une astuce : la double projection (ou
% projection de Karhunen-Loeve).
% On calcule les axes principaux de la matrice des
% variance-covariance de la transpose de X : X', puis on retourne
% dans l'espace initial des visages par la X'*A
% Stilde=cov(X'); %Adaptation Image
Stilde=X*X'/n; %Adaptation Image
% %Calcul des axes principaux A et des lambdak
[A,lambda]=eigsort(Stilde);
A=X'*A;
% %on norme les axes principaux
A=A*(diag(diag(A'*A)))^(-1/2);%Adaptation Image
%On a au plus min(n,p)-1 axes principaux
ptilde=min(n,p);
A=A(:,1:ptilde-1);
lambda=lambda(1:ptilde-1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculer l'inertie cummule des vecteurs propres. C'est le pourcentage de
%% description des axes principaux
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
slambda=cumsum(lambda)/sum(lambda);
figure;
plot([1:ptilde-1],slambda)
%pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% La projection des éléments sur les axes principaux
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C=X*A;

```

Fichier Appel.m

```

%pathOfImage="/home/thuanvh/sl/index-mutilmedia-doc/visagerecog/att_faces"

pathOfImage="../att_faces";

```

```

numberOfClass=40;
listOfLearn=[1:5]';
listOfTest=[6:10]';
#listOfLearn=[6:10]';
#listOfTest=[1:5]';
#
[BD,names,cl,nrow,ncol] = CreateBD(pathOfImage,listOfLearn,numberOfClass);
[A,C,lambda,BDbarre] = ACP(BD);
print("imgplot.png");
imgBDbarre=reshape(BDbarre,nrow,ncol);
imshow(imgBDbarre,[]);
print("imgBDbarre.png");
%imshow(imgBDbarre,[0,255])
%imagesc(imgBDbarre)
kfacetoview=5;
for (i=1:kfacetoview)
eigenface=reshape(A(:,i),nrow,ncol);
% imagesc(eigenface)
imshow(eigenface,[]);
% sauver des images
print(strcat('eigenface',num2str(i,2),'.png')) ;
end
# choisir le nombre des axes principaux à utiliser
k=75;
A=A( :,1 :k);
C=C( :,1 :k);
#télécharger des images de test
[BDtest,namestest,cltest,nrowtest,ncoltest] = CreateBD(pathOfImage,listOfTest,numberOfClass);
[nctest p]=size(BDtest);
[nlearn p]=size(C);
#calculer les coordonnées sur les axes principaux
Xtest=double(BDtest)-ones(nctest,1)*BDbarre;
Ctest=Xtest*A;
#lister des 5 premiers coefficients des 10 premiers éléments
Ctest(1:10,1:5)
#nombre des tests corrects
right=0;
#liste de correspondance
matchStr=cellstr("");
#matrice de confusion
matConf=zeros(numberOfClass,numberOfClass);
#CMC values
cmcRange=20;
cmcValues=zeros(1,cmcRange);
#test
for(i=1:nctest)
testitem=Ctest(i,:); #item de test
distanceVector=zeros(nlearn,1) ;#la vecteur
for(j=1:nlearn)
learnitem=C(j,:);
#calculer la distance euclidienne
distanceVector(j)=sqrt(sum((testitem-learnitem).^2));
end
#chercher l'élément de distance minimale
[minDistance minIndex]=min(distanceVector);
#afficher la correspondance
matchStr{i}=sprintf("%s a pour plus proche voisin : %s",namestest{i},names{minIndex});
#mis à jour la matrice de confusion
matConf(cltest(i),cl(minIndex))++;
if(cltest(i)==cl(minIndex))
right++;
endif
#calculer cmcValues
[sortedDistance indexOrigin]=sort(distanceVector);
for(j=1:cmcRange)
if(cltest(i)==cl(indexOrigin(j)))
cmcValues(1,j)++;
break;
end
end
end
#afficher la correspondance de test
matchStr
right #le nombre de test correct

```

```

right/nctest #le pourcentage de test correct
#afficher la matrice de confusion
matConf
matConf(5:10,5:10)
matConf(11:15, 11:15)
#calculer le rappel et la précision
matRappel=zeros(numberOfClass,1);
matPrecision=zeros(numberOfClass,1);
cellRappelPrecisionStr=cellstr("");
for(i=1:numberOfClass)
matRappel(i)=matConf(i,i)/sum(matConf(i,:));
matPrecision(i)=matConf(i,i)/sum(matConf(:,i));
cellRappelPrecisionStr{i}=sprintf("s%d : %f %f",i,matPrecision(i),matRappel(i));
end
cellRappelPrecisionStr
#
right #le nombre de test correct
right/nctest #le pourcentage de test correct
#calculer images 1
imagegen=reshape(BDbarre,nrow,ncol);
imageindex=1;
imageC=C(imageindex,:);
for(q=1:k)
cq=imageC(1,q);
aq=A(:,q);
imagegen=imagegen+reshape(cq*aq,nrow,ncol);
imshow(imagegen,[]);
% sauver des images
print(strcat('imagegen',num2str(q,2),'.png')) ;
end
#Afficher CMC
cmcValues
#accumuler des valeurs
scmcValues=cumsum(cmcValues)/sum(cmcValues);
figure;
#plot([1:cmcRange],scmcValues)
plot(scmcValues)
print("cmcplot.png");
sumCmcValues=sum(cmcValues);
tempvalue=0;
for(i=1:cmcRange)
cmcValues(i)=tempvalue+cmcValues(i);
tempvalue=cmcValues(i);
end
#calculer pourcentage
cmcValues=cmcValues/tempvalue;
cmcValues
plot(cmcValues)
print("cmcplot.2.png");

```