L'Ecole Supérieure de Technologie Essaouira

Projet de Fin d'Etudes

Pour l'obtention du

Diplôme Universitaire de Technologie / DUT filière Génie Informatique

Sujet :

Détection de visage par la méthode d'Analyse en Composantes Principales (ACP)

Réalisé par : Mohamed AIT MANSOUR, Marouane CHIHAB

L'Ecole Supérieure de Technologie Essaouira, du 31/12/2014 au 27/04/2015

Sous la direction de : M. Noureddine MOUSSAID

Année scolaire 2014/2015

Remerciements

Ce travail a été effectué dans le cadre d’obtention de DUT INFORMATIQUE à l’ESTE et encadré par Pr. Noureddine MOUSSAID, nous tiens à le remercier pour nous avoir permis de réaliser ce projet et de nous avoir orienté vers la thématique de l'étude et la mesure de le traitement d’images et l’utilisation de MATLAB. Nous lui suivons également reconnaissons pour tous les précieux conseils qu'il nous a donnés, pour la confiance qu'il nous a accordée et pour le temps qu’il nous a consacré tout au long de cette période, sachant répondre à toutes nous interrogations; sans oublier sa participation au cheminement de ce rapport.

Immanquablement, nous souhaitons remercier nos familles pour leur soutien constant. Ce projet, aboutissement de longues années d’études, nous la dois beaucoup à nos parents, à nos sœurs et nos frères exceptionnels avec qui nous avons vécu dans un climat toujours serein, à l’abri de tous soucis affectifs. Il est impossible de trouver des mots pour dire à quel point nous sommes fières d’eux, et à quel point nous les aime.

Table des matières

Table des figures

Introduction

Dans le cadre de notre deuxième année du DUT à l'école Supérieure de Technologie Essaouira, nous avons eu l'opportunité de réaliser un projet de fin d'études, consacre le traitement d'images, précisément la détection de visage par la méthode d'ACP (Analyse par Composantes Principales).

Passionnés depuis notre plus jeune âge par l'informatique et l'analyse d'information (quel que soit images son…), ce projet était pour nous l'occasion de découvrir le domaine de traitement d'images d'un point de vue professionnel et de mettre en œuvre nos connaissances théoriques et développer nos capacités personnelles.

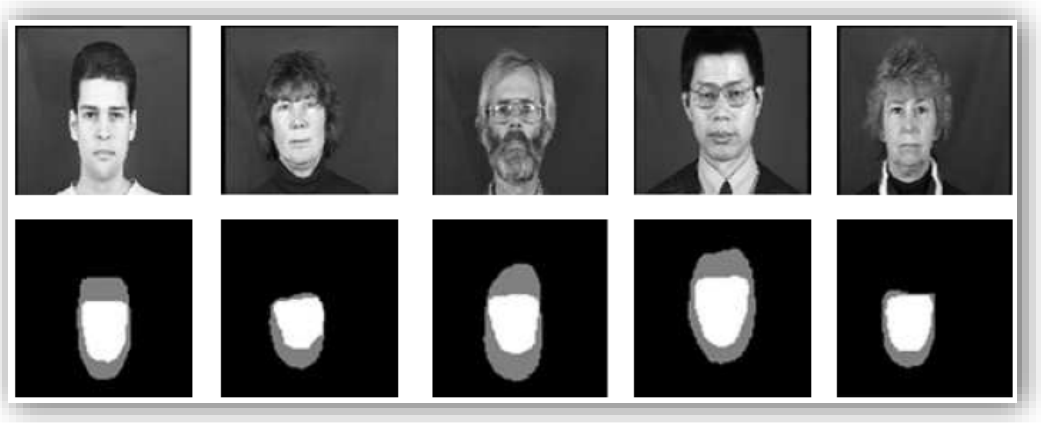


Figure 1 : Images originales et les trois zones caractéristiques associées à chacune d'elles (ACP)

Ce projet représente une expérience de réalisation d'une application de détection de visages (projet proposé par Mr MOUSSAID Noureddine). Ce projet a pour objectif la réalisation d'un logiciel de détection de visage. Il est découpé en deux parties liées mais distinctes : le codage d'un outil de détection à l'aide de l'algorithme **Analyse en Composantes Principales,** et la réalisation d'une interface graphique permettant d'utiliser n'importe quel programme de reconnaissance pourvu qu'il respecte un certain format. En particulier, cette interface servira à l'utilisation de notre premier code.

La détection de visage s'inscrit dans le domaine plus vaste de la vision par ordinateur, qui part du constat que le sens le plus utilisé par l'homme est la vue. Dès lors, il peut s'avérer très utile de donner des « yeux » à son ordinateur : ainsi, il peut devenir capable de remplacer ceux de l'homme pour des tâches répétitives telles que la reconnaissance de nombreux visages, ou demandant des calculs conséquents comme le lissage d'une image ou l'augmentation de sa netteté.

Les objectifs de ce projet étaient les suivants :

- ✓ Développement d'une application avec MATALAB
- ✓ S'initialiser avec le domaine traitement d'images
  - o Segmentation
  - o Détection de contour
  - o Filtrages
  - o …
- ✓ Acquisition d'images par WEBCAM
- ✓ Détection de visage(s)

Pourquoi ?

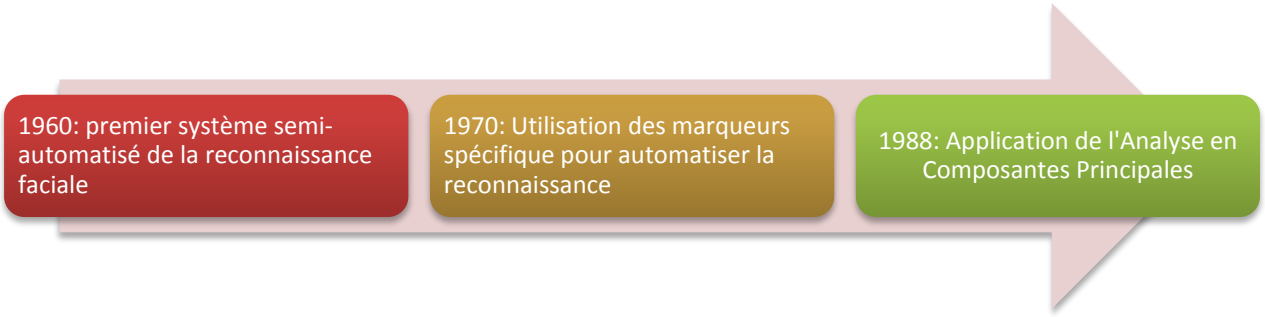
- ✓ Pour le traitement, l'analyse d'images et détection de visages.
- ✓ Programmer les déférentes formules mathématiques des étapes d'ACP.
- ✓ Etudier le principe de fonctionnement d'ACP

Historique

La reconnaissance faciale automatique est un concept relativement nouveau. Le premier système semi-automatisé de la reconnaissance faciale a été développé dans les années 1960, il nécessite à l'administrateur de localiser les yeux, les oreilles, le nez et la bouche sur la photo et de saisir les distances calculées et les ratios à un point de référence commun, qui ont ensuite été comparés aux données de référence.

Dans les années 1970, Goldstein, Harmon et Lesk ont utilisé marqueurs spécifique tels que la couleur des cheveux et l'épaisseur de la lèvre pour automatiser la reconnaissance. Le problème avec ces deux premières solutions, c'est que les mesures et les emplacements ont été calculés manuellement.

En 1988, Kirby et Sirovich ont appliqué l'**analyse en composantes principales** (ACP), une technique standard de l'algèbre linéaire. Cela a été considéré en quelque Sorte comme une étape importante car elle a montré qu'au moins une centaine de valeurs ont été nécessaires pour coder convenablement et avec précision une image alignée et normalisée.



Problématique du projet :

Identifier une personne à partir de son visage est une tâche aisée pour les humains. En est-il de même pour une machine ? Ceci définit la problématique de la reconnaissance automatique de visages, qui a engendré un grand nombre de travaux de recherche au cours des dernières années. Le visage peut être considéré comme Une donnée biométrique qui est une donnée qui permet l'identification d'une personne sur la base de ce qu'il est (caractéristiques physiologiques ou comportementales).

Pour reconnaître une personne à partir de son image, il faut passer par certaines étapes. Tout d'abord il faut localiser le visage dans l'image. Ensuite il faut éventuellement le normaliser pour ramener le visage à une taille standard. Puis il faut aborder la phase de reconnaissance proprement dite. Dans ce rapport nous proposons une synthèse de travaux permettant d'illustrer deux des étapes clés de cette chaîne d'un processus de reconnaissance faciale. Nous détaillons d'abord une des approches concurrentes pour détecter un visage, qui est basée sur les «ACP ».

Après la détection d'un visage (sujet de projet), la partie suivante sert à l'identifiée, dans un processus enchainé, cette schéma illustre les différentes étapes à suivre afin d'identifié une personne :

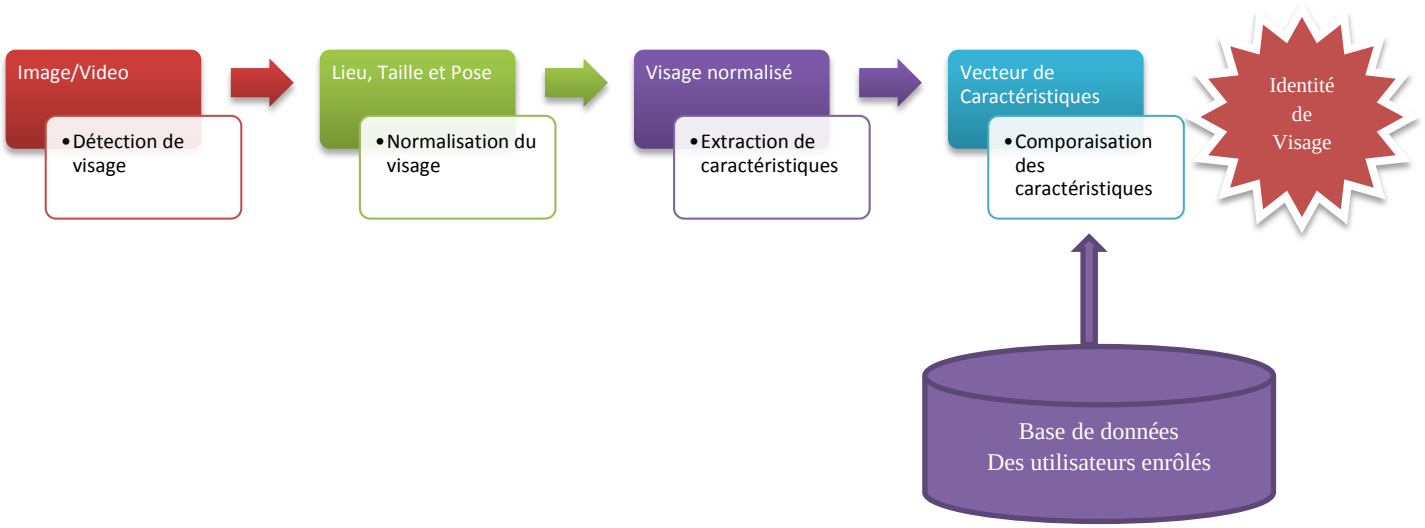


Figure 2 Schéma général d'un système de reconnaissance des visages.

Méthodes de détection et de reconnaissance des visages

I. Détection de visages

La détection de visage dans l'image est un traitement indispensable et crucial avant la Phase de reconnaissance. En effet, le processus de reconnaissance de visages ne pourra jamais devenir intégralement automatique s'il n'a pas été précédé par une étape de détection efficace.

Le traitement consiste à rechercher dans une image la position des visages et de les Extraire sous la forme d'un ensemble d'image dans le but de faciliter leur traitement ultérieur. Un visage est considéré correctement détecté si la taille d'image extraite ne dépasse pas 20% de la taille réelle de la région faciale, et qu'elle contient essentiellement les yeux, le nez et la bouche.

Les premières difficultés rencontrées par les méthodes de détections de visages sont les variations de pose (vue de profil ou de face), d'expression, de rotation du visage, d'âge et d'illumination. Ce dernier type de difficulté pouvant être surmonté par un prétraitement de normalisation et de compensation de l'illumination.

II. Reconnaissance de visages

De nombreuses méthodes de reconnaissance de visages ont été proposées au cours Des trente dernières années. La reconnaissance faciale automatique est un challenge tel qu'il a suscité de nombreuses recherches dans des disciplines différentes : psychologie, neurologie, mathématiques, physique et informatique (reconnaissance des formes, réseaux de neurones, vision par ordinateur). C'est la raison pour laquelle la littérature sur la reconnaissance de visages est vaste et diversifiée.

Les systèmes de reconnaissance de visages sont très souvent classés à partir des conclusions d'études psychologiques sur la façon dont les hommes utilisent les caractéristiques faciales pour reconnaitre les autres. De ce point de vue, on distingue les deux catégories : les méthodes globales et les méthodes locales.

1. Méthodes globales

Le principe de ces méthodes est de représenter une image faciale par un seul vecteur De grande dimension en concaténant les niveaux de gris de tous les pixels du visage. Cette représentation, appelée description basée sur l'apparence globale, a deux avantages. Premièrement, elle conserve implicitement toutes les informations de texture et de forme utiles pour différentier des visages. Deuxièmement, elle peut tenir compte des aspects d'organisation structurelle globaux du visage. Toutefois, son inconvénient majeur réside dans la dimension très grande de l'espace image qu'elle nécessite ce qui rend très difficile la classification.

Pour traiter le problème des données de grande dimension, des techniques de réduction de la dimensionnalité peuvent être utilisées. L'une des techniques les plus courantes pour la reconnaissance de visages est la description par visages propres, qui est basée sur l'**analyse en composantes principales (ACP).**

Analyse en composantes principales (ACP)

Une méthode très populaire, basée sur la technique ACP, est la méthode Eigen face. Son principe est le suivant : étant donné un ensemble d'images de visages exemples, il s'agit tout d'abord de trouver les composantes principales de ces visages. Ceci revient à déterminer les vecteurs propres de la matrice de covariance formée par l'ensemble des images exemples. Chaque visage exemple peut alors être décrit par une combinaison linéaire de ces vecteurs propres. Pour construire la matrice de covariance, chaque image de visage est transformée en vecteur. Chaque élément du vecteur correspond à l'intensité lumineuse d'un pixel.

L’ACP est une technique rapide, simple et populaire dans l’identification de modèle, C’est l’une des meilleures techniques. Les projections de l’ACP sont optimales pour la reconstruction d’une base de dimension réduite. Cependant, l’ACP n’est pas optimisé pour la séparabilité (discrimination) de classe.

2. Méthodes locales

Les méthodes locales peuvent être classées en deux catégories, les méthodes basées sur les points d’intérêt et celles basées sur l’apparence du visage.

Méthodes locales basées sur les points d’intérêts

On détecte tout d’abord les points d’intérêts et ensuite on extrait des caractéristiques localisées sur ces points d’intérêt. Les méthodes les plus anciennes en reconnaissance de visages appartiennent à cette catégorie.

Elles s’appuient toutes sur l’extraction de caractéristiques géométriques spécifiques telles que la largeur de la tête, les distances entre les yeux,... Ces données sont ensuite utilisées par des classificateurs afin de reconnaître des individus. Ces méthodes présentent les deux inconvénients suivants :

- les caractéristiques géométriques sont difficiles à extraire dans certains cas puisque la tâche de la détection précise de points caractéristiques n’est pas facile, en particulier dans les cas où des occultations ou des variations (pose, expression) de visages sont présentes.
- les caractéristiques géométriques seules ne sont pas suffisantes pour représenter réellement un visage, alors que d’autres informations utiles telles que les valeurs des
- Niveaux de gris de l’image sont complètement écartés.

Les méthodes locales basées sur l’apparence du visage

Dans les méthodes locales basées sur l’apparence du visage, on divise le visage en petites régions. Généralement cette méthode basée sur des techniques d’analyse Statistiques (pourcentage d’existence des modèles dans l’image) et d’apprentissage Automatique pour trouver des caractéristiques significatives des visages et des non visages.

Principales difficultés de la reconnaissance de visage

Pour le cerveau humain, le processus de la reconnaissance de visages est une tâche visuelle de haut niveau. Bien que les êtres humains puissent détecter et identifier des visages dans une scène sans beaucoup de peine, construire un système automatique qui accomplit de telles tâches représente un sérieux défi. Ce défi est d'autant plus grand lorsque les conditions d'acquisition des images sont très variables. Il existe deux types de variations associées aux Images de visages : inter et intra sujet. La variation inter-sujet est limitée à cause de la ressemblance physique entre les individus. Par contre la variation intra-sujet est plus vaste. Elle peut être attribuée à plusieurs facteurs que nous analysons ci-dessous.

I. Changement d'illumination

Le changement d'apparence d'un visage du à l'illumination, se révèle parfois plus critique que la différence physique entre les individus, et peut entrainer une mauvaise classification des images d'entrée.

Figure 3 Changement d'illumination

II. Variation de pose

Le taux de reconnaissance de visage baisse considérablement quand des variations de pose sont présentes dans les images. Cette difficulté a été démontrée par des tests d'évaluation élaborés sur les bases FERET et FR VT 126). La variation de pose est considérée comme un problème majeur pour les systèmes de reconnaissance faciale. Quand le visage est de profil dans le plan image (orientation 300), il peut être normalisé en détectant au moins deux traits faciaux (passant par les yeux). Cependant, lorsque la rotation est supérieure à 300, la normalisation géométrique n'est plus possible

Figure 4 Variation de pose

1. Initiation MATLAB

1.1. Définition

MAT (rix) **LAB** (oratory) est un logiciel puissant doté à la fois d'un langage de programmation haut niveau et d'outils dédiés au calcul numérique et à la visualisation numérique. Développé en C par la société Mathworks (http://www.mathworks.com/). Matlab était initialement destiné à faire du calcul matriciel simplement.

Actuellement, Matlab recouvre d'autres domaines d'applications de l'informatique scientifique :

- visualisation graphique 2D et 3D
- résolution d'équations aux dérivées partielles
- optimisation
- traitement du signal
- traitement de l'image
- logique floue
- réseaux de neurones
- …

Les systèmes Matlab se divise en deux parties :

- Le noyau Il comprend :
  - ✓ l'environnement de travail offrant plusieurs facilités pour la manipulation des données.
  - ✓ son interpréteur permet de tester rapidement ses propres programmes Matlab.
  - ✓ le système graphique Matlab (interfaces homme-machine, graphiques, images, animations).
  - ✓ le langage de programmation Matlab.
  - ✓ une librairie de fonctions mathématiques Matlab.
  - ✓ un système d'interfaçage facilitant l'exécution de programmes C ou Fortran ou sous Matlab.

- 2) Des Toolboxes (boîtes à outils)
  - ✓ Ils regroupent un ensemble de fonctions spécifiques à un thème.

1.2. Environnement de travail MATLAB

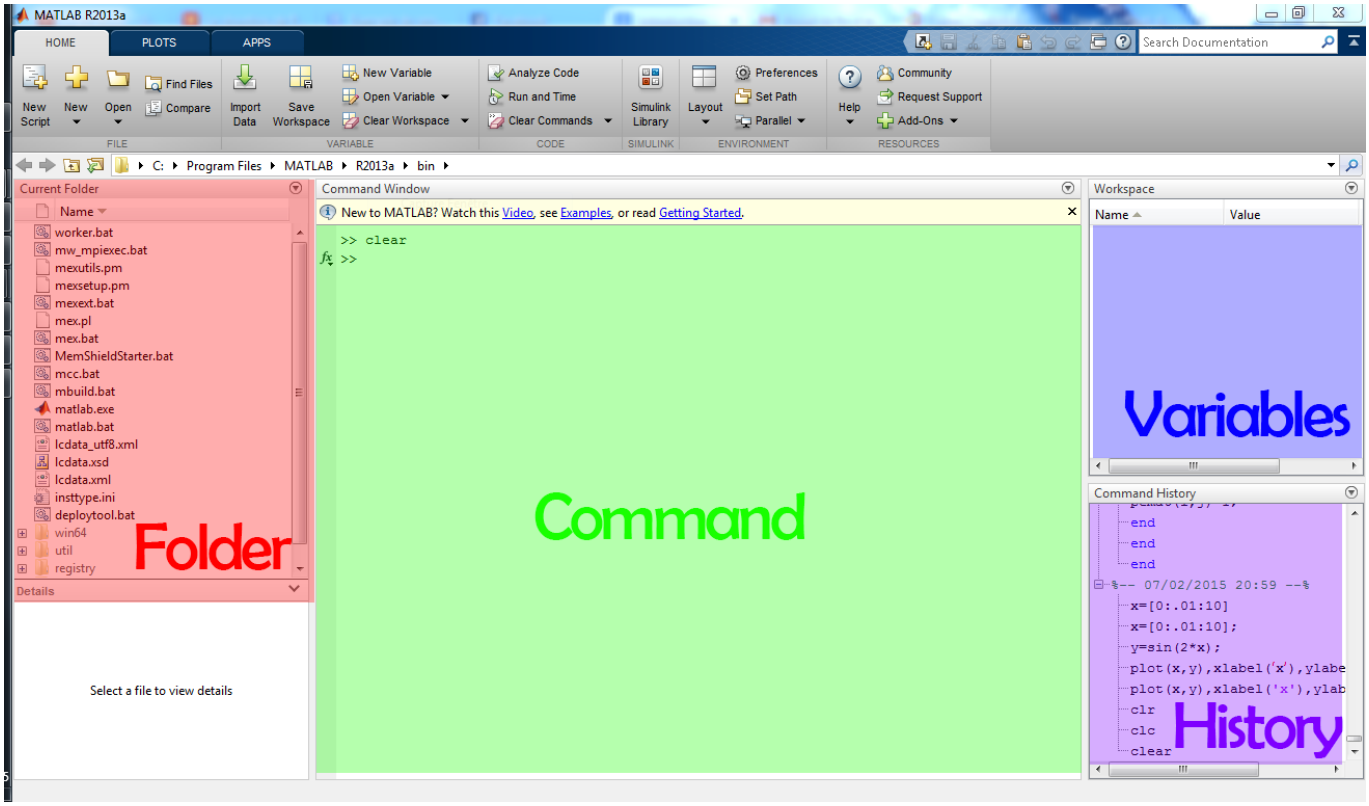
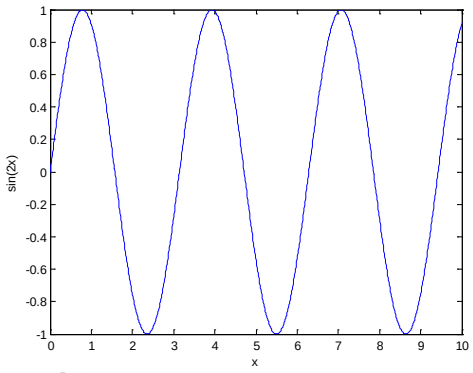


Figure 5: Environnement MATLAB

1.3. Exemples d'utilisation : Création d'un signal sinusoïdal

Le code ci-dessus fournit le résultat ci-contre

```
>> x=[0:.01:10];
>> y=sin(2*x);
>> plot(x,y),xlabel('x'),ylabel('sin(2x)')
```



Dans l'exemple suivant, on va créer une fonction sous Matlab, cette fonction va nous générer un bruit et on va appliquer ce bruit sur un morceau vocal (MP3, WMA…) pour écouter ce bruit, et après on va le supprimer a nouveau.

Avant de commencer le programme voilà la syntaxe d'une fonction sous Matlab :

```
function [ output_args ] = Untitled2( input_args )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here


end
```

le mot clé **Function** détermine le début de fonction, suit par les deux crocher [**les argument sortie**] ex [a1,a2,a3]

- ✓ on ajoute que Untitled2 indique le nom de fonction
- ✓ (input_args) c'est les arguments que l'utilisateur peux entrer ex (aBruit, FileName, N_0)
- ✓ Après écrire le protocole de la fonction on passe au code de la fonction

- ✓ Puis terminer avec le mot clé **end**.

Dans cette exemple on a utiliser des fonctions prédéfinis pour lire et extraire les composants d'un fichier audio, parmis ces fonction on a :

- ✓ [Y_out,Fs]audioread(Filename)
- ✓ sound(y_out);

```
function [Y_out, FS ] =CalMoy( aBruit, FileName, N_0)
[Y, FS]=audioread(FileName);
Y_1=Y(:,1);
Y_2=Y(:,2);
N=length(Y_1); % ecraser la valeur de N encienne
test=5*N_0;
if N < test
    Y_out=0;
    FS=0;
else

    Bruit= abs(aBruit*rand(N,1)); % generation du bruit

    Y_Bruit= Y_1 + Bruit; % ajouter le bruit au fichier audio

    Y_out=0*Y_1;
    % Calcul de la moyenne (debruittage du signal audio)
    for i=1:N-N_0

        Y0=Y_Bruit(i:i+N_0-1);
        % Moyenne
        Y_out(i)=sum(Y0)/N_0;
    end
    Y_out=Y_out*max(Y_1)/max(Y_out);
end
end
```

On a aussi bien implémenté la boucle **FOR** afin de parcourir les matrices résultats

Pour appeler une fonction sous MATLAB, il suffit d'indiquer les variables de sortie, suivi de nom de la fonction et les variables d'entrés, en prend l'exemple de fonction de bruit qu'on a déjà créé : Les mots surligner sont les noms de variables (E/S).

```
>> [Y_out, FS ] =CalMoy( 5, 'im_son.wav', 5);
```

Après l'exécution de cette instruction, voilà les variables résultats (les variables d'entré doivent être dans le même chemin ou se trouve le programme).

Finalement pour lire le fichier son résultat on peut utiliser la fonction sound(Y_out) afin d'entendre le son avec le bruit ajouté.

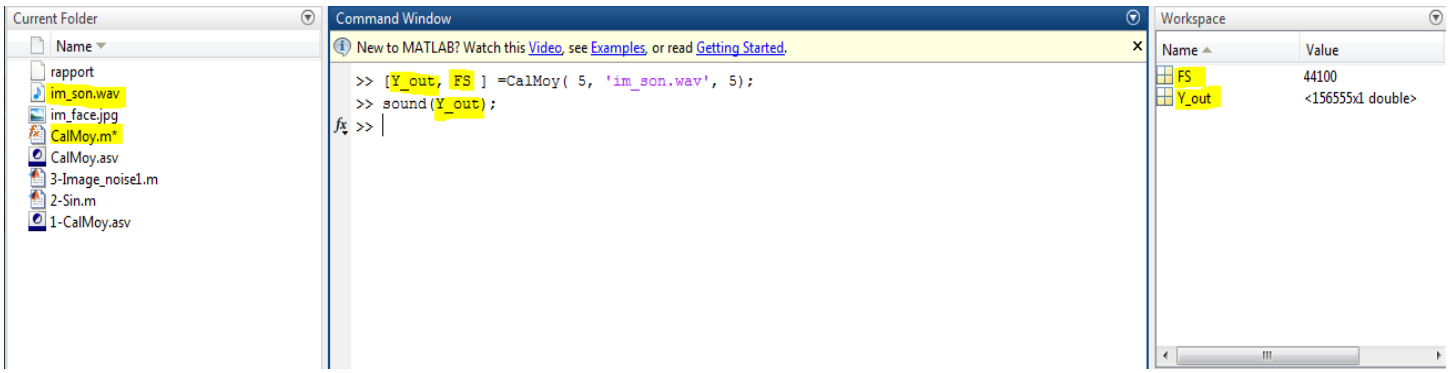


Figure 6: Les variables utilisés

1.4. Ajouter de bruits une image :

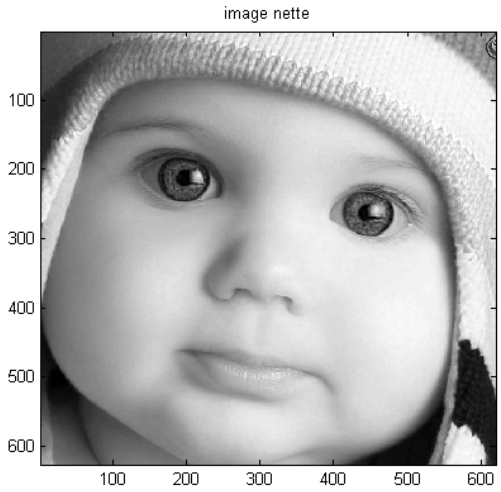
Il est assez simple de voir la déférence entre une image bruité en la comparant avec une image nette et claire, le code et les astuces suivantes conduisent à générer du bruit et le ajouté à une images, ce bruit sera générer aléatoirement avec la fonction rand, Le bruit d'image est la présence d'informations parasites qui s'ajoutent de façon aléatoire aux détails de la scène photographiée numériquement. Il est plus particulièrement visible dans les zones peu éclairées.

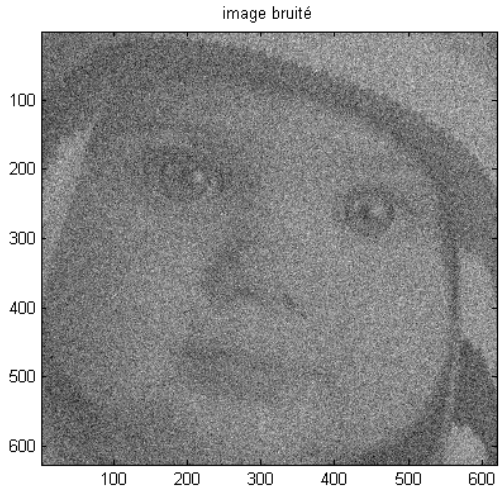
```
% lire une  Image
im=imread('im_test.jpg');

%extraire les matrices composantes de l'image
im1=double(im(:,:,1));
im2=double(im(:,:,2));
im3=double(im(:,:,3));
```

Les *images en couleurs*, en revanche, peuvent être représentées par trois matrices. Chaque matrice détermine la quantité de **rouge**, de **vert** et de **bleu** qui constitue l'image. Ce modèle de couleur est appelé RVB (il existe de nombreux autres modèles de couleur, pour diverses utilisations: La quadrichromie ou CMJN (en imprimerie), Y'IQ (pour la télévision analogique NTSC), etc.). Les éléments de ces matrices sont des nombres entiers compris entre 0 et 255 qui déterminent l'intensité de la couleur de la matrice pour le pixel correspondant. Ainsi, avec le modèle RVB, il est possible de représenter $2563 = 2^{24} = 16777216$ couleurs différentes.

```
%Bruit
[m, n] = size(im_gD);
Bruit=100.0;
im_n = Bruit*randn(m,n);
```





2. Notions pour le traitement des images

1.1. Définition d'une image :

C'est la représentation d'une personne ou d'un objet par la peinture, la sculpture, la photographie, etc. C'est aussi un ensemble structuré d'informations qui après affichage sur écran, ont une signification pour l'œil humain. Elle peut être écrite sous forme de fonction I(x,y) ou I est une fonction d'intensité lumineuse ou de couleur aux coordonnées spatiales (x,y) . De cette façon l'image est exploitable par la machine, d'où la nécessité de sa numérisation.

1.2. L'image numérique :

Il est clair que les images manipulées par l'ordinateur sont numériques (série de bits). L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés pixels, ayant comme caractéristique le niveau de gris ou de couleur. La numérisation d'une image est la conversion de celle-ci en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x,y) qui sont les niveaux de gris des coordonnées réelles (x,y).

1.3. Caractéristiques d'une image numérique :

Pixel :

C'est le plus petit point de l'image. Chaque pixel a une valeur numérique qui représente le niveau de gris ou de couleur selon la nature de l'image.

Dimension :

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentant les intensités lumineuses (pixel).

Résolution :

C'est la clarté ou la finesse des détails atteinte par un moniteur ou une imprimante dans la production d'image, sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixel par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels affichable horizontalement ou verticalement sur un moniteur; plus grand est ce nombre, meilleure est la résolution.

Niveaux de gris :

C'est la valeur numérique qui reflète l'intensité de la luminosité d'un point. Pour niveaux de gris compris entre et, chaque pixel sera codé sur bits, et ses niveaux de gris seront obtenus après dégradation de la couleur noire, représente le blanc et représente le noir.

Figure 7 Niveaux de gris

Image en couleur :

C’est une image ou chaque pixel est codé dans l’espace de couleur RGB (rouge, vert, bleu). Donc c’est une représentation dans un espace tridimensionnel de la valeur d’intensité lumineuse du pixel. Ce dernier sera codé sur trois octets, un pour chacune des couleurs.

Bruit :

C’est un phénomène de brusques variations d’un pixel par rapport à ces voisins suivant un certain seuil. Il existe quatre sources de dégradation induisant le bruit, qui sont : le bruit lié au contexte d’acquisition, le bruit lié au capteur, le bruit lié à l’échantillonnage,

Histogramme:

En imagerie numérique, l’histogramme représente la distribution des intensités (ou des couleurs) de l'image. C'est un outil fondamental du traitement d'images, avec de très nombreuses applications. Les histogrammes sont aussi très utilisés en photographie et pour la retouche d'images.

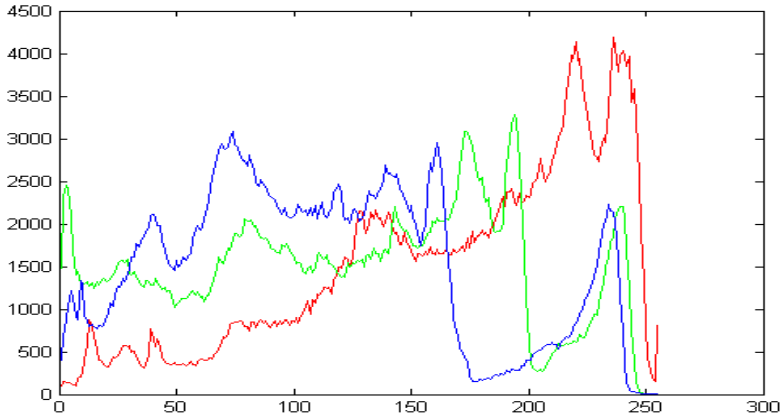


Figure 8: Histogramme RGB

Binarisation:

C'est la transformation de l'image à plusieurs niveaux de gris en une image binaire (à deux niveaux de gris). Elle peut être réalisée en choisissant un certain seuil, et tous les pixels ayant des valeurs inférieures à celui-ci deviennent noirs, tous ceux de valeurs supérieures deviennent blancs. Nous pouvons aussi effectuer la Binarisation en choisissant un certain intervalle [Min, Max], et les pixels qui se trouvent à l'intérieur de ce dernier sont forcés à 1 et les autres à 0.

Figure 9: Binarisation

3. Détection de visages (ACP)

Préambule

L'algorithme ACP, PCA en anglais (Principal Component Analyses) est né des travaux, Il est aussi connu sous le nom d'Eigen faces car il utilise des vecteurs propres et des valeurs propres. Cet algorithme s'appuie sur des propriétés statistiques bien connues et utilise l'algèbre linéaire. Il est relativement rapide à mettre en œuvre mais il reste sensible aux problèmes d'éclairement, de pose et d'expression faciale.

Nous cherchons à effectuer de la détection de visage, en employant des systèmes simples comme une webcam. Le sujet se plaçant alors face à la caméra, nous partons donc de la remarque, qui nous simplifie grandement notre démarche, selon laquelle nous pratiquons de la reconnaissance à deux dimensions.

1. Acquisition des images par WEBCAM, ou par un fichier:

Pour acquérir une image depuis une webcam, Matlab a déjà défini des fonctions pour utiliser se périphérique, dans la fonction qu'on a créé on a implémenté plusieurs fonction (`FramesPerTrigger, videoinput..),` Le code suivant présente la fonction qui nous permet de faire un Snapshot ou capture par la webcam.

```
function im_cam = get_im_cam(showVid)

% showVid != 1 : Avoir une image depuis la WEBCAM
% showVid = 1 : Avoir une image et lire la video

if(~isdeployed)
    cd(fileparts(which(mfilename)));
end

%% Start up the video and get image
vid = videoinput('winvideo', 1);
vid.FramesPerTrigger = 1;

if (showVid==1)
    preview(vid);
end

start(vid);

% Get image
im_cam = getdata(vid);

end
```

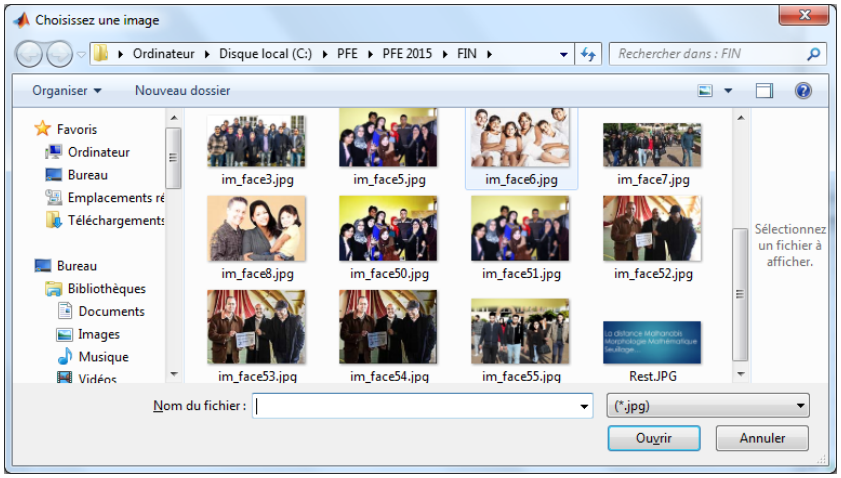
le paramètre d'entré **showVid** doit être initialiser avant d'entrer dans la fonction, il sert a choisir ou bien juste faire un Snapshot sans afficher la vidéo de Cam, ou faire le capture et afficher la vidéo de la webcam.

Appelle de la fonction :

```
>> showVid=1;
>> im_cam = get_im_cam(showVid);
>>
```

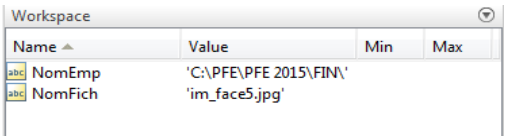
Si on souhaite de sélectionner une image à partir d'un fichier dans l'ordinateur, on peut utiliser cette commande :

```
[NomFich,NomEmp] = uigetfile({'*.jpg';},'Choisissez une image'); % Choisir une image
im_in=imread(NomFich);
im_test=double(rgb2gray(im_in));
imshow(im_in);
```



Cette commande nous donne l'accès à l'ordinateur afin de sélectionner le fichier qu'on veut choisir.

On peut aussi choisir le type de fichier à sélectionner, le chemin par défaut, et on stock le nom, et l'extension dans deux variables Matlab.



2. Calcule de modèle moyen :

Dans le cas de notre projet, notre approche consiste à représenter un visage comme étant la combinaison linéaire d'un ensemble d'images, ces dernières formant une base de référence. Mathématiquement, cela revient à parvenir à l'équation :

$$\Phi_i = \sum_{i=1}^{n} p_i d_i$$

Nous allons chercher à trouver les visages propres ; tout d'abord, nous devons prendre **un nombre M** de visages d'apprentissage. Chacune de ces images, qui sont en pratique des **matrices N N** sont alors transformées en une unique vectrice colonne de **longueur N2**.

Matrice $N \times N$ initiale :

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{1i} & \alpha_{1N} \\ \alpha_{21} & \alpha_{22} & \alpha_{2i} & \alpha_{2N} \\ \dots & \dots & \dots & \dots \\ \alpha_{N1} & \alpha_{N2} & \alpha_{Ni} & \alpha_{NN} \end{pmatrix}$$

```
for i=1:M
    Chemin = strcat(CheminVisages,'v',num2str(i),'.',ExtensionVisage);

    im1=imread(Chemin);
    imRz1=imresize(im1,[N N]);
    im_g1=double(imRz1(:,:,k));

    %% converssion matrice to table
    im_1=im_g1';
    phi1 = im_1(:);
    Q(:,i)=phi1;
end
```

transformée en :

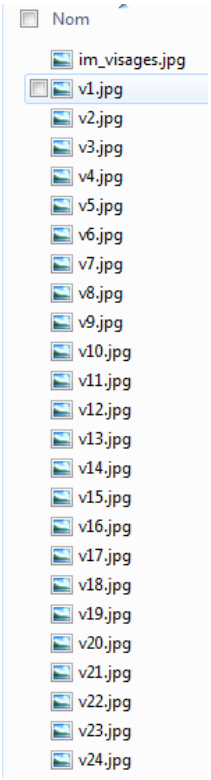
$$\begin{pmatrix} \alpha_{11} \\ \vdots \\ \alpha_{N1} \\ \vdots \\ \alpha_{1N} \\ \vdots \\ \alpha_{NN} \end{pmatrix}$$

Nous devons par la suite déterminer le **visage moyen**, déduit des M visages d'apprentissages.

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

Ce visage moyen va servir dans l'analyse d'images, on soustrait en effet ce visage moyen aux visages d'apprentissages, ce qui nous laisse alors les informations propres à ce visage, nous récupérons alors dans i uniquement les informations qui sont particulières à ce visage d'apprentissage.

- Exemple d'utilisation :



[22]Après avoir une base de données pour les visages, on va calculer d'abord un visage moyen on utilisant la fonction **get_model :**

```
>> im_model = getModel(N,M, CheminVisages, ExtensionVisage);
>> im_model= tab2mat(im_model,N,N);
>> colormap(gray);
>> imagesc(im_model);
>>
```

CheminVisages	'visages/'		
ExtensionVisage	'jpg'		
M	110	110	110
N	250	250	250
im_in	<200x400x3 uint8>	0	255
im_model	<250x250 double>	1.2102...	2.9397...

Résultats :

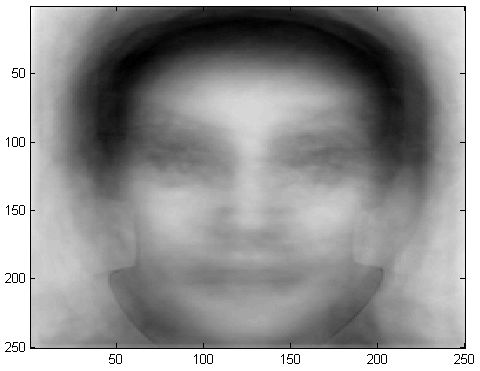


Figure 10 Visage moyen (110 images)

Figure 11 Exemples de visage moyen

3. La distance Mahalanobis

Lorsqu'on souhaite comparer deux vecteurs de caractéristiques issus du module d'extraction de caractéristiques d'un système biométrique, on peut soit effectuer une mesure de similarité (ressemblance), soit une mesure de distance (divergence).

- **Distance City-Block :**

Pour p=1, on obtient la distance City-Block (ou de Manhattan) :

$$L_1(x,y) = \sum_{i=1}^{N} |x_i - y_i|$$

- **Distance euclidienne :**

Pour p=2, on obtient la distance euclidienne :

$$L_2(x,y) = \sqrt{\sum_{i=1}^{N} |x_i - y_i|^2}$$

Les objets peuvent alors apparaître de façons très différentes selon la mesure de distance choisie :

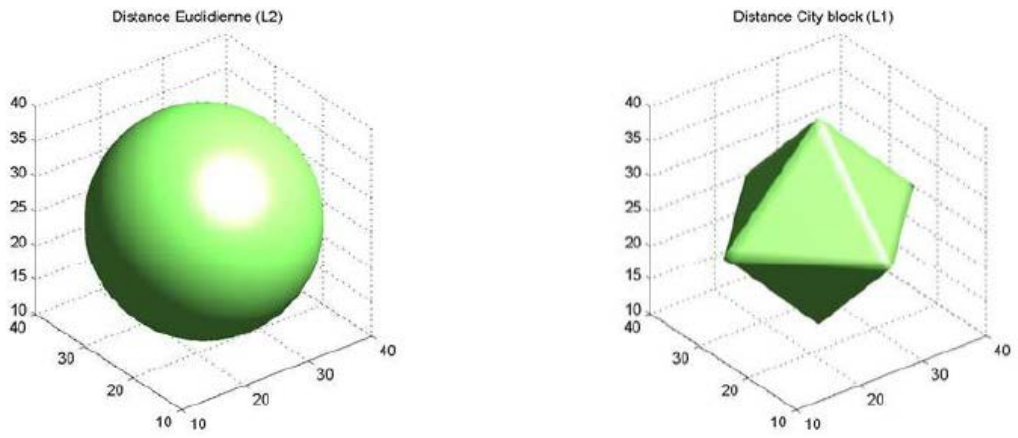


Figure 12 représentation d'une sphère avec Distances (euclidienne et City-Block)

Distance Mahalanobis: cette distance est exactement la même que la distance City-Block sauf que les vecteurs sont projetés dans l'espace de Mahalanobis. Ainsi, pour des vecteurs propres u et v de projections respectives m et n sur l'espace de Mahalanobis, la distance Mahalanobis est définie par :

$$Mah_{L_1}(u,v) = \sum_{i=1}^{N} |m_i - n_i|$$

```
>> for i=N/2+1:m-N/2
    for j=N/2+1:n-N/2
        im1=im_test(i-N/2:i+N/2-1,j-N/2:j+N/2-1);
        v_im1=im1(:);
        dist_mal0 = mahal(v_model,v_im1);
        dist_sort=sort(dist_mal0);
        dist_sort=1./(dist_sort+eps);
        dist_mal(i,j)=sum(dist_sort(end-N_min+1:end));
    end
end
```

4. Seuillage et morphologie mathématique

- **Le seuillage**

Le seuillage permet d'éliminer des points en lequel la différence d'intensité est trop faible pour que le point soit un point de contour. On peut pousser le seuillage afin d'obtenir une image binaire : tout point de l'image est un point de contour ou n'en est pas un. On peut cependant vouloir conserver une information moins binaire tout en éliminant certaines valeurs d'intensité.

Le seuillage le plus simple consiste à **éliminer les points dont le module de contour est en dessous d'un seuil minimum**. Il ne tient pas compte de la topologie et peut conserver un point bruité isolé mais éliminer une portion de contour d'intensité faible mais réelle.

```
>> %% Seuillage moyenne de l'image
ValMax=1e5;
dist_mal1=floor(ValMax*dist_mal/max(max(dist_mal)));
dist_mal2 = SeuilMoy( dist_mal1, 1, 2.5e4 )
```

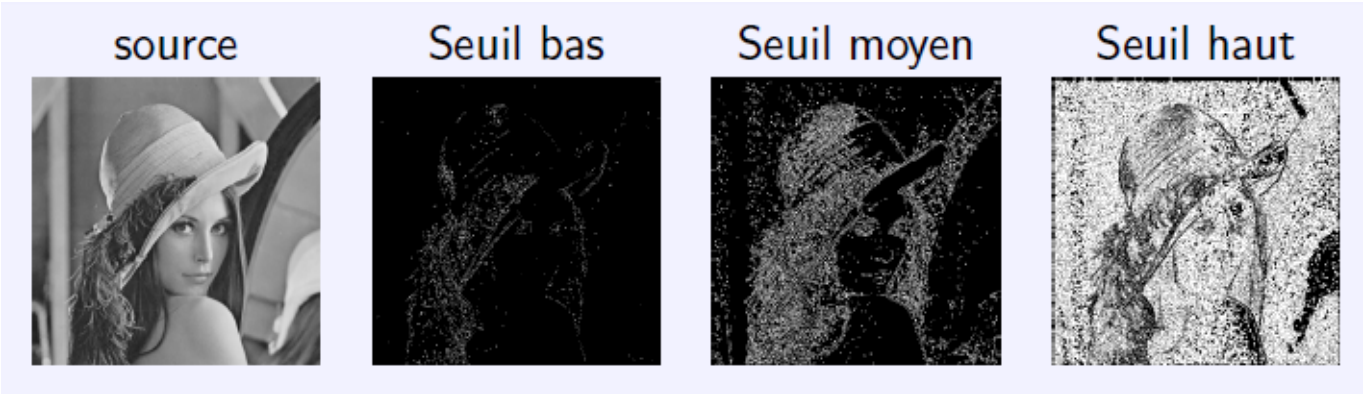


Figure 13 Exemple seuillage

- **Morphologie mathématique**

Qui dit la morphologie mathématique, dit la délation et l'érosion, Le résultat de la **dilatation** de l'ensemble A par l'ensemble B est l'ensemble des points tels que lorsque B est centré sur un de ces points il y a une intersection non vide entre A et B.

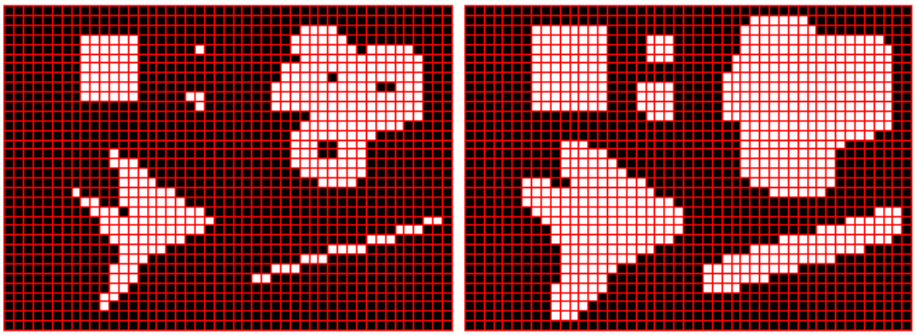


Figure 14 Dilatation, A gauche : Originale

Erosion : L'érodé de A par B correspond à l'ensemble des points tels que si B est centré sur ces points, B est entièrement inclus dans A.

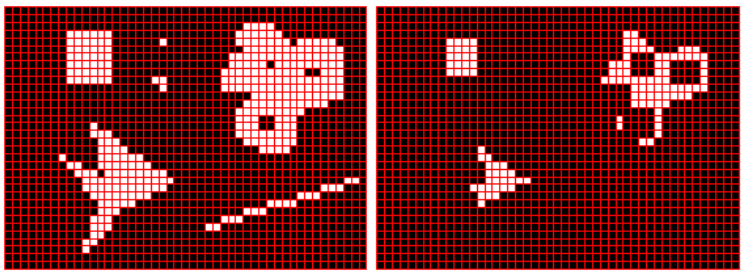


Figure 15 Erosion, A gauche : Originale

```
>> %% Morphologie mathematique

se = strel('square',5);
% se = strel('ball',11,11);
im_op1 = imerode(dist_mal2,se);
im_op1 = imerode(im_op1,se);

BW1 = edge(im_op1,'prewitt');
BW2 = edge(im_op1,'canny');
BW3 = edge(im_op1,'sobel');
```

Les opérateurs morpho mathématiques se sont initialement appliqués sur des images en noir et blanc (Matheron et Serra, 1965). Ils ont ensuite été étendus à des images en niveaux de gris par Dougherty en 1978. Pour les appliquer à des images couleurs, il suffit alors de les appliquer séparément à chaque composante couleur.

5. Traçage du carré

Après avoir une région binaire qui contient (dans l'image RGB) le visage, alors on va d'abord tracer une carre sur cette région qui va encadrer le visage.

La première étape pour tracer ce cadre c'est de calculer un barycentre pour chaque visage (point centrale) dont il sera le centre de carré qu'on va tracer, et c'est effectuer par cette fonction:

```
>> function [x, y] = CalcBary( index_reg, im_segm )

% x = sum(xi * pi)/sum(pi)
% y = sum(yi*pi)/sum(pi)
% sum(xi*pi ) = x1*p1 + x2*p2 + x3*p3...

im_segm=double(im_segm);
[m0, n0] = find(im_segm==index_reg);
Taille = length(m0);

sum_pi = Taille*index_reg;
% sum_pi = sum(im_segm(m0(1:end),n0(1:end)));

somme_xi=0.0;
somme_yi=0.0;

for i=1:Taille
    somme_xi=somme_xi + m0(i)*im_segm(m0(i),n0(i));
    somme_yi=somme_yi + n0(i)*im_segm(m0(i),n0(i));
end

x=round(somme_xi/sum_pi);
y=round(somme_yi/sum_pi);

end
```

```
>> function mat_out = TracerCarer( mat_in,N0, x, y)

%creation de la matrice m*n
[m, n] = size(mat_in);
mat_out=zeros(m,n);

%Creer et renplir la petite matrice
pemat=zeros(N0*2+1,N0*2+1);

pemat(1,1:end)=255;
pemat(1:end,1)=255;
pemat(end,1:end)=255;
pemat(1:end,end)=255;

% Transferer la petite matrice au
mat_out(x-N0:x+N0, y-N0:y+N0) = pemat(:,:);

end
```

Après le calcul du barycentre, on va tracer la carre à l'aide d'une fonction qu'on a définie, nommé TracerCarre :

6. Affichage des résultats

La dernière étape de notre projet consiste à afficher les résultats, on affiche 4 figures, chaque une de ces figures illustre comment l'image se présente à chaque stage de traitement, à savoir le stage de Binarisation, morphologie mathématique, traçage de cadre dans l'image noir et blanc, puis traçage de cadre sur l'image en couleur. Sous Matlab et pour afficher les figures on utilise les figures, et les subplots comme l'illustre le code suivant :

```
>> %% Affichage
figure(1)
subplot(121)
imagesc(im_op2);
colormap(gray)

subplot(122);
imagesc(im_op3);
colormap(gray)


figure(2)
imagesc(im_rgb_out);

figure(3)
imagesc(mat_out);
colormap(gray)
```

Cette première figure affiche l'image au stade du Binarisation, (seuillage)

Cette deuxième figure affiche l'image au stade d'érosion/délations

La troisième étape consiste à affiché les cadres tracés mais juste en une matrice avec des 1 (vide) et 0 (pour carre)

La dernière étape consiste à afficher l'image finale en couleur et avec le carré tracé

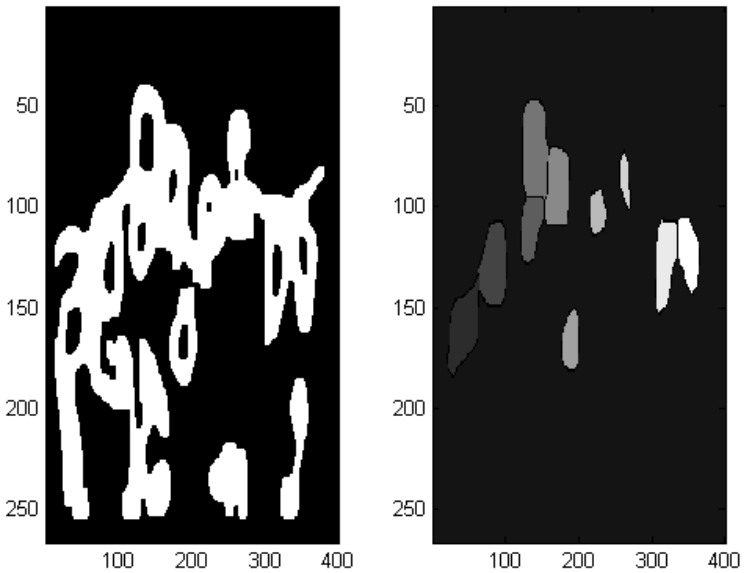


Figure 16 Binarisation et morphologie mathématiques

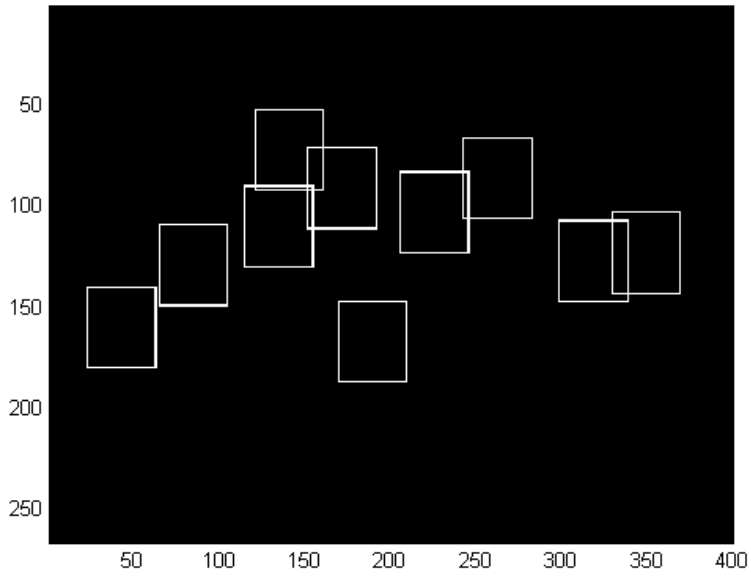
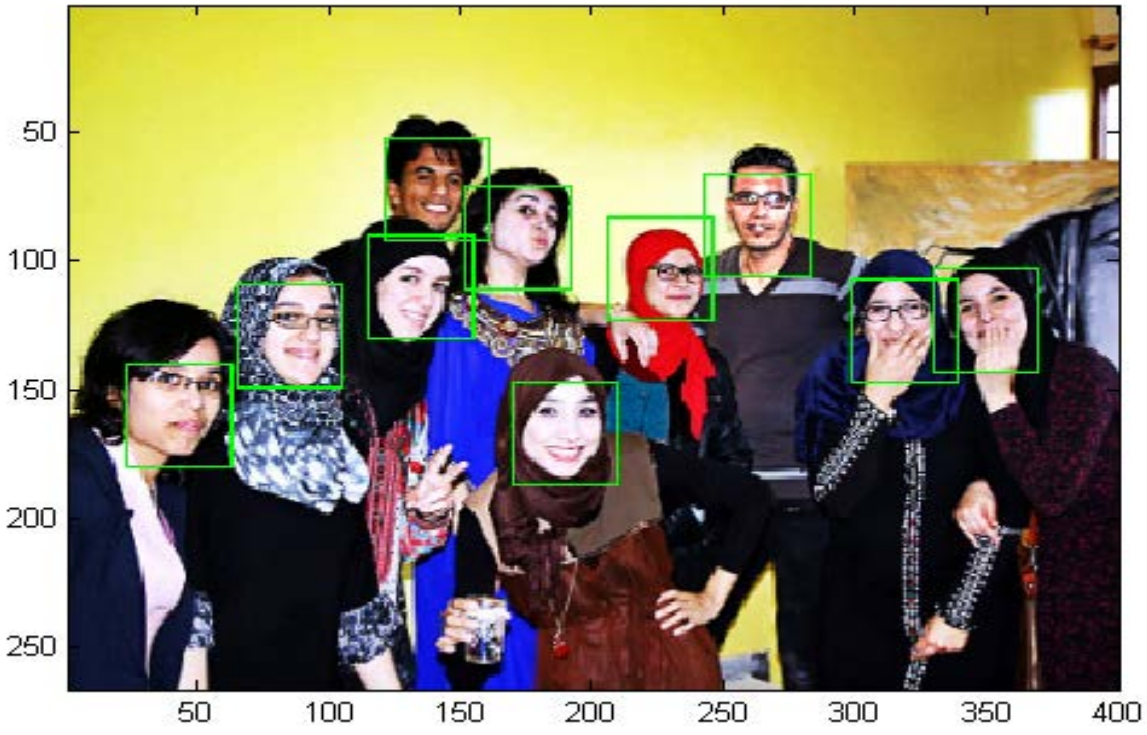


Figure 17 Matrice vide (avec carrés)

RESULTAT



7. Les limites de cette méthode

Malgré le grand succès connu par cet algorithme, il présente plusieurs d´efficiences dans certaines condition ce qui donne quelques limites d´détectées à l'étape de teste de l'application.

- **Angle de visage**

Le programme ne détecté rien et ne fonctionne pas quand des têtes présentées dans l'image est inclinée latéralement ou verticalement.

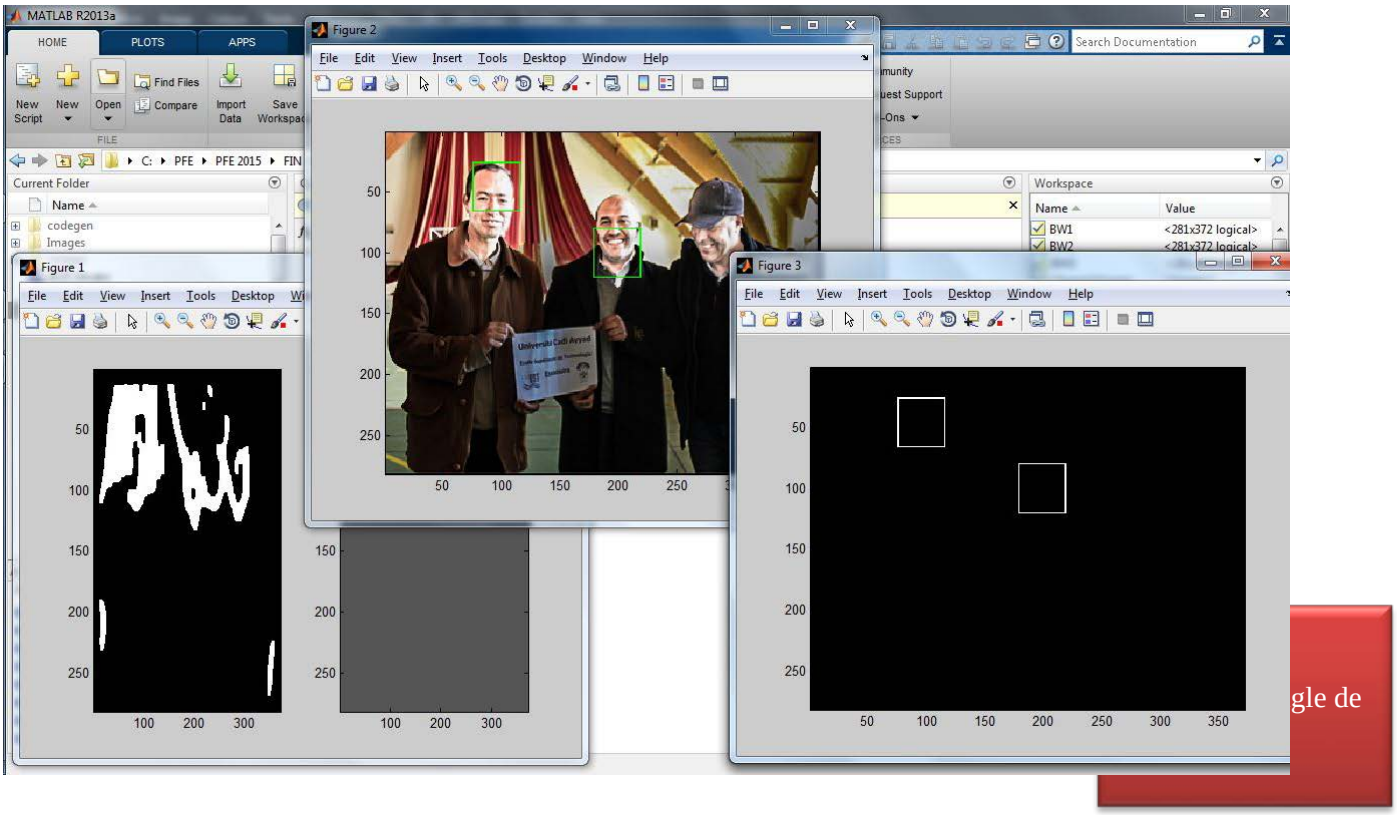


Figure 18 résultat de d´détection de visage incline

- **Occlusion de visage :(une partie du visage est cachées)**

Dans ce cas l'algorithme s'exécute mal si un visage est partiellement cache par

Un autre ou par un objet quelconque.

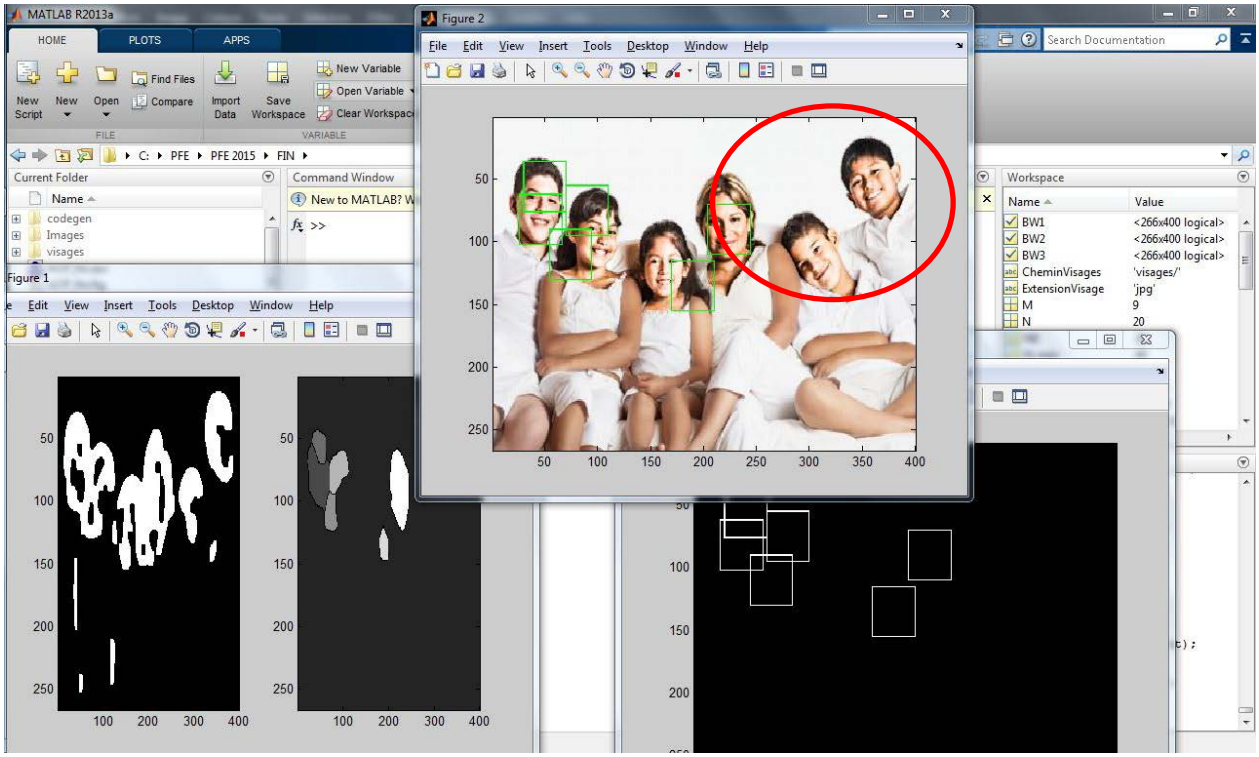


Figure 19 Résultat de la d´détection des visages partiellement cachent

- **les faux Positifs : (false positives)**

Bien que la valeur des faux positifs fixée pour le logiciel ait une probabilité relativement basse ils apparaissent toujours en quelques photos. Généralement ils apparaissent dans des mains car ils ont une couleur presque exacte à celle de la peau et si le font du cote du visage est petit il produit aussi des corrélations significatives.

4. L’interface graphique (GUI)

1-2: Nous avons créé l’interface de notre application à l’aide de le GRAPHICAL USER INTERFACE sous Matlab, cette interface consiste dans un premier temps à prend une photo soit avec une WEBCAM ou bien dans les fichiers d’utilisateur à l’aide d’un explorateur intégrer sous Matlab.

3-4: Après l’ajout de fichier, l’utilisateur peut calculer le model moyen, il peut évidemment l’afficher à l’aide d’un bouton «Afficher modèle moyen».

5-6: Après le calcule de modèle moyen, l’utilisateur demande de calculer la distance Mahalanobis et aussi terminer les autre calcules. Enfin l’utilisateur peut afficher et voir l’image résultats.

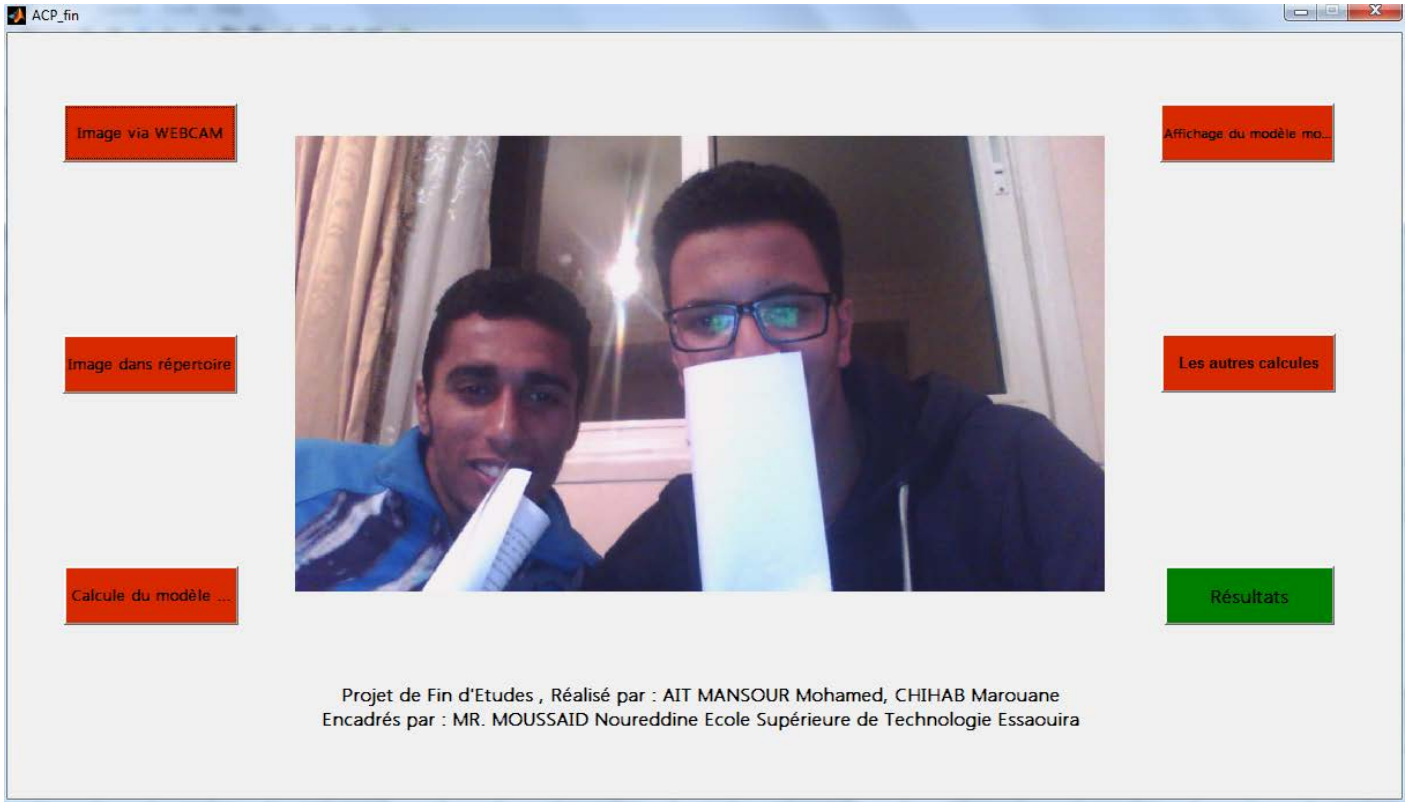


Figure 20 L'interface graphique

Conclusion

L’objectif de ce mémoire est de concevoir et d’implémenter une application de détection de visage capable, en temps réel, de détecté les visages. Vu la quantité de logiciels potentiels (sécurité, réseaux sociaux,...) pouvant se baser sur cette application, celle-ci doit répondre à des exigences de rapidité et de robustesse des résultats.

Notre travail consiste à mettre au point un algorithme efficace destiné à détecter un visage d’un individu en utilisant la méthode ACP. L’ACP est une méthode mathématique qui peut être utilisée pour simplifier un ensemble de données, en réduisant sa dimension. Elle est utilisée pour représenter efficacement les images de visages, qui peuvent être approximativement reconstruites à partir d’un petit ensemble de poids et d’une image de visage standard.

Il faut savoir que l’ACP est considéré comme la méthode la plus simple et la plus précise de détection de visage, mais beaucoup d’autres méthodes ou combinaisons de méthodes multiples (beaucoup plus compliqué) peuvent être utilisées.

En effet, capturer une image d’un visage, en particulier à travers une caméra 2D, est simple et non invasif. C’est donc une modalité biométrique facilement tolérée par les utilisateurs, mais les performances de la détection des visages sont toujours bien en delà de ce que l’on pourrait espérer pour de telles applications.

Ce projet nous a permis de découvrir plus profondément plusieurs aspects du développement d’une application complexe. Il nous a fallu d’abord nous renseigner sur le côté algorithmique de la reconnaissance de visage, et plus généralement de la vision par ordinateur, qui est un domaine en pleine expansion. Cette recherche nous a donc menés à la réalisation d’une application de détection de visage brut. Il nous a fallu résoudre plusieurs problèmes algorithmiques ayant plus ou moins de rapport avec les mathématiques, discipline importante dans le traitement d’images en général.

Si nous avions plus de temps, nous aurions aimé passer de la détection à la reconnaissance faciale, et développer un peu plus l’interface Graphique afin de compléter son rôle. Il aurait été également intéressant d’affiner un moteur de reconnaissance, en trouvant un moyen de prendre des images correctement cadrées et en déterminant les seuils de façon rigoureuse.