
Promo 2019

# Facificator

Dexemple François, Luiggi Tristan, Ngo Jean-François,
Utard Yannick

7 Decembre 2015

*Nom du groupe :* D.U.N.L. Studio
*Chef de projet :* Dexemple François

# Contents

**1 Introduction** **3**

**2 Groupes** **4**
2.1 François Dexemple . . . . . 4
2.2 Tristan Luiggi . . . . . 4
2.3 Jean-François Ngo . . . . . 5
2.4 Yannick Utard . . . . . 5

**3 Nature et but du projet** **7**
3.1 Qu'est-ce que ? . . . . . 7
3.2 Principe . . . . . 7
3.3 Règles à respecter . . . . . 7

**4 Objectifs détaillés** **8**
4.1 Soutenance 1 . . . . . 8
4.1.1 Traitement d'un visage : . . . . . 8
4.1.2 Construction d'une base de données de caractéristiques de visage avec les noms associés : . . . . . 8
4.2 Soutenance finale . . . . . 8
4.2.1 Extraction de la liste des personnes : . . . . . 8
4.2.2 Interface utilisateur permettant d'effectuer toutes les opérations, notamment : . . . . . 8

**5 Les tâches effectuées** **10**
5.1 Tableau récapitulatif . . . . . 10
5.2 Le traitement d'image . . . . . 11
5.3 La base de données . . . . . 13
5.3.1 La création . . . . . 13
5.3.2 L'insertion . . . . . 13
5.3.3 La suppression . . . . . 13
5.3.4 L'affichage . . . . . 13
5.3.5 La destruction de la base . . . . . 13
5.3.6 L'insertion en masse . . . . . 14
5.4 La méthode de Viola et Jones . . . . . 15
5.4.1 L'image intégrale . . . . . 15
5.4.2 Les caractéristiques pseudo-Haar . . . . . 17
5.4.3 L'apprentissage et la classification . . . . . 19
5.4.4 Adaboost . . . . . 20
5.4.5 La cascade de classifieurs . . . . . 24

5.4.6 La détection . . . . . 26
5.5 Les eigenfaces . . . . . 28
5.5.1 Pré traitement . . . . . 28
5.5.2 Matrice de covariance et valeurs propres . . . . . 29
5.6 L'interface graphique . . . . . 30
5.6.1 La programmation événementiel . . . . . 30
5.6.2 La forme . . . . . 30
5.6.3 Le fond . . . . . 31

**6 Conclusion** **32**

# 1 Introduction

Ce document est un rapport de projet et a pour but de présenter le logiciel Facificator après 13 semaines de travail sur ce dernier. Un logiciel réalisé par D.U.N.L Studio. Pour rappel, le logiciel réalisé est un logiciel de reconnaissance faciale développé en langage C, sous un environnement Unix. Celui-ci comporte plusieurs fonctionnalités telles que la reconnaissance faciale notamment. Malgré la charge de travail conséquente, nous avons tenté tant bien que mal de tenir les objectifs fixés que vous pouvez retrouver dans ce rapport. Ainsi, ce projet nous a permis d'apprendre à coder en C grâce aux cours de programmation et aux travaux pratiques d'Epita. Mais également grâce à de nombreux tutoriels sur internet, qui nous ont permis de nous familiariser avec ce langage. Ce projet nous a également appris et fait découvrir la programmation événementielle avec le GTK. Afin de réaliser ce projet de reconnaissance faciale, nous nous sommes basés sur la méthode de Viola et Jones publié en 2001.

# 2 Groupes

## 2.1 François Dexemple

J'ai toujours aimé bidouiller les ordinateurs que ce soit sous Windows ou Linux, ou encore au niveau matériel. Par contre je n'ai jamais réellement touché à un langage de programmation à part ce que nous avons fait en C# l'année dernière. Je suis repartis cette année pour faire ce nouveau projet qui est peut être moins amusant à faire mais tout aussi intéressant. Pour ce projet je me suis remis avec Yannick, avec lequel je m'étais très bien entendu lors du projet de l'année dernière et deux autres anciens camarades de classe.
Étant très curieux à tout ce qui touche l'informatique j'apprends relativement vite les nouvelles choses qui m'intéressent. C'est donc avec enthousiasme que je commence ce projet. Pour ce projet je me suis porté volontaire pour faire la base de données car je suis en train de faire en parallèle une application androïd qui utilise une base SQLite, c'était donc un moyen de m'entrainer et de commencer assez vite à travailler puisque je connaissais déjà les bases. De plus, connaissant déjà très l'environnement Linux et git, grâce au projet de l'année dernière, j'ai pu commencer très vite

## 2.2 Tristan Luiggi

Etant passionné d'informatique depuis l'enfance, intégrer l'EPITA était un choix évident lors de mon orientation en Terminale. Ainsi après une année passer des cette école j'ai pu commencer à apprendre quelques langages et me lancer dans des projets. Ces projet me permettent de me confronté a ce que pourrait être mon future métier d'ingénieur et d'améliorer plusieurs conpetances avec celui ci tel aue le travail d'equipe ou encore l'autonomie. Le projet de l'année de SUP imposait peu de contrainte et donnait un accèe à un environnement de devellopement assez fourni, cette année nous âssons à la vitesse superieur avec l'arrivé de Linux et du language C. De plus la reconnaissance faciale nous oblige à abordé des notions totalement nouvelles. ce projet estun reel defis qui nous rendera plus autonome et competent à l'aboutissment.

## 2.3 Jean-François Ngo

Depuis tout jeune, l'informatique a toujours été un domaine qui m'intéressait fortement même si je ne saurai expliquer précisément pourquoi. C'est pourquoi j'ai décidé d'intégrer une école comme Epita qui propose une formation dans ce sens. Après une première année ponctué par l'apprentissage d'un nouveau langage, le C#, et la mise en pratique de ce langage dans un projet de réalisation d'un jeu vidéo qui a permis d'acquérir de nombreuses compétences et connaissances. Une deuxième année commence avec un nouveau projet à réaliser dans un nouveau langage qu'est le C, sous Linux. Ce projet, bien plus compliqué que celui de l'année dernière pour moi, est un nouveau défi qui nous oblige à énormément lire afin de comprendre comment marche la reconnaissance faciale, qui constitue un énorme travail en amont avant de commencer la programmation même par rapport au projet de l'année dernière. De plus, tout comme le projet de l'année dernière, je pense que ce projet me permettra de me plonger dans un travail d'équipe, de groupe, ce qui je pense m'attend dans ma future vie d'ingénieur.

## 2.4 Yannick Utard

Comme l'année dernière nous revoilà sur un nouveau projet en informatique. L'an dernier, il s'agissait d'un jeu vidéo, ce qui était un sujet très attrayant pour tout amateur en informatique. Le sujet de cette année était à première vue moins ludique, cependant il s'est révélé être très intéressant. Je n'avais jusqu'à présent jamais traité d'image de cette façon, mis à part lors d'un TP de C# en SUP. De plus le C est un nouveau langage mais j'ai réussi à m'adapter rapidement grâce aux tutoriels sur internet et aux TPs de l'école; afin de progresser dans ce projet. Dans l'équipe, le D.U.N.L Studio, il y a une bonne ambiance étant donné que nous nous connaissons depuis la SUP et que j'ai travaillé sur le projet de l'an dernier avec François. Le projet avance donc dans la bonne humeur même si nous avons parfois eu du mal avec la gestion du temps car le travail est assez conséquent. Enfin, je pense que nous avons atteint nos objectifs pour cette première soutenance et nous ne sommes pas prêts de nous arrêter là !

Le projet, réalisé par un groupe de quatre personnes sur une durée d'environ 13 semaines, est un logiciel de reconnaissance faciale. Le logiciel devra permettre à partir d'une photo de groupe d'identifier les personnes présentes sur la photo (Par exemple, pour une photo de classe, le logiciel devra être en mesure d'indiquer les personnes présentes sur la photo mais aussi de déterminer les absents). C'est un projet qui s'articule sur deux axes, l'apprentissage d'une base de visage à partir de photos d'identité et l'extraction des visages présents sur une photo de groupe comme dit précédemment. Ainsi, le logiciel final devra permettre de charger un ensemble de visages associées avec le nom de la personne correspondante pour construire la base de données, elle devra permettre d'ajouter de nouveaux visages ou de corriger les informations associées à ceux-ci également, mais elle permettra aussi de charger une photo de groupe et d'obtenir la liste des présentes sur la photo.

Ce projet sera développé sous Linux et sera codé en C uniquement (norme C99). Avec l'aide de toutes les bibliothèques disponibles ou directement installables sur le rack de l'école via les paquets, en particulier la bibliothèque SDL et ses composants pour le chargement des images et la bibliothèque GTK pour l'interface graphique.

# 3 Nature et but du projet

## 3.1 Qu'est-ce que ?

C'est un logiciel de recherche de personnes présentes sur une photo. A partir d'une photo le logiciel vous donnera la liste des personnes présentes sur la photo et se trouvant dans sa base de données.

## 3.2 Principe

L'utilisation du logiciel se découpe en deux parties : dans un premier temps l'utilisateur va remplir la base de données de visages dont seront extraites les informations pertinentes pour l'identification des personnes ; une fois la base remplie, le logiciel permettra à partir d'une photo de trouver la liste des personnes présentes.

Le logiciel devra également permettre de gérer la base en fournissant des opérations d'ajout, de suppression ou de correction des personnes enregistrées dans la base.

## 3.3 Règles à respecter

Pour que le développement de ce projet se passe le mieux possible, nous allons respecter les règles de programmations suivantes :

Pour le code :
- Indentation des fonctions obligatoire.
- Les noms de fonctions, variables, constantes et macros devront être en anglais.
- Établir un découpage correct et cohérent du code en plusieurs fichiers avec les fichiers d'en-têtes correspondant.
- 80 caractères par ligne.
- Pas d'espace en fin de ligne dans les fichiers .c et .h.
- Aucun warning à la compilation sur votre code (les seuls warning tolérés sont ceux produits par le code des bibliothèques utilisées.)

Pour le projet :
- Votre projet devra disposer d'un Makefile utilisable à l'école et sans modification. Celui-ci doit être compatible avec GNU Make.
- Votre Makefile devra compiler votre projet par défaut (appel à make sans paramètre ou avec all).
- Votre Makefile devra contenir une règle clean pour effacer tous les produits de la compilation.
- Vous pouvez utiliser indifféremment gcc ou clang pour compiler.

- Le code devra compiler avec les options suivantes : -Wall -Wextra -std=c99 -pedantic -03.
- Un fichier README, en anglais, qui explique comment compiler et utiliser votre projet.
- Un fichier AUTHORS listant les membres du projet, un par ligne sous la forme : * login (NOM Prénom)

# 4 Objectifs détaillés

## 4.1 Soutenance 1

### 4.1.1 Traitement d'un visage :

- Détection du visage sur une photo d'identité (ou assimilée).
- Pré-traitement pour accentuer les caractéristiques intéressantes.
- Détection des éléments clefs pour l'analyse.
- Construction de la représentation caractéristique d'un visage.
- Option avancée : détection et découpage des visages dans une photo de groupe.

### 4.1.2 Construction d'une base de données de caractéristiques de visage avec les noms associés :

- Opérations de maintenance (ajout, suppression et modification).
- Ajouts en masse de visage dans la base (mode batch).
- Recherche d'une personne à partir d'un visage seul (photo d'identité ou visage extrait d'une photo de groupe.)

## 4.2 Soutenance finale

### 4.2.1 Extraction de la liste des personnes :

- Chargement et pré-traitement d'une photo de groupe.
- Finalisation de la détection et du découpage des visages dans une photo de groupe.
- Recherche des visages dans la base (identification par le nom associé d'un visage).

### 4.2.2 Interface utilisateur permettant d'effectuer toutes les opérations, notamment :

- Gestion complète de la base de données.
- Identification de visage(s) sur photo (individuelle ou de groupe).

- Comparaison entre deux visages.

# 5 Les tâches effectuées

## 5.1 Tableau récapitulatif

| | François | Tristan | Jean-François | Yannick |
|---|---|---|---|---|
| Conversion d'une image en niveau de gris | • | • | • | 100% |
| Conversion en image intégrale | • | • | • | 100% |
| Base de données | 100% | • | • | • |
| Interface graphique | 70% | • | 30% | • |
| Adaboost | • | • | • | 100% |
| Caractéristiques | • | • | • | 100% |
| Détection | • | • | • | 100% |
| Reconnaissance | • | 100% | • | • |

## 5.2 Le traitement d'image

Le traitement d'image est l'élément essentiel de ce projet en C. Il a été très important pour nous de directement nous informer le traitement des pixels d' une image être a l'aise sur ce projet ensuite. Pour charger une image, récupérer ses pixels ou encore l'afficher nous utilisons la bibliothèque SDL.

```
int32 getpixel(SDL_Surface *surface, unsigned x, unsigned y) {
 Uint8 *p = pixelref(surface, x, y);
 switch(surface->format->BytesPerPixel) {
 case 1:
   return *p;
 case 2:
   return *(Uint16 *)p;
 case 3:
   if(SDL_BYTEORDER == SDL_BIG_ENDIAN)
     return p[0] << 16 | p[1] << 8 | p[2];
   else
     return p[0] | p[1] << 8 | p[2] << 16;
 case 4:
   return *(Uint32 *)p;
 }
 return 0;



oid putpixel(SDL_Surface *surface, unsigned x, unsigned y, Uint32 pixel) {
 Uint8 *p = pixelref(surface, x, y);
 switch(surface->format->BytesPerPixel) {
 case 1:
   *p = pixel;
   break;
 case 2:
   *(Uint16 *)p = pixel;
   break;
 case 3:
   if(SDL_BYTEORDER == SDL_BIG_ENDIAN) {
     p[0] = (pixel >> 16) & 0xff;
     p[1] = (pixel >> 8) & 0xff;
     p[2] = pixel & 0xff;
   } else {
```

Tout le traitement d'image se situe dans notre fichier pixel_operations.c. Pour simplifier notre code, nous avons créés plusieurs structures en C dans le fichier types.h et quelques méthodes (d'allocation en particulier) pour ces structures dans types.c. Par exemple, la structure training_image_t sert d'exemple d'apprentissage pour Adaboost (méthode de boosting, voir page...). Cette structure contient les champs: - SDL_Surface* image (une image de 24 sur 24 pixels) - features_array_t features(le tableau de 162 336 feature_t, feature_t est également une structure représentant une caractéristique pseudo-Haar) - int is_face (entier determinant si l'image est un visage ou non) - double weight (le poids de l'image pour les calculs d'Adaboost)

```c
//image data set
typedef struct {
  SDL_Surface* image;
  features_array_t* features;
  int is_face;
  double weight;
} training_image_t;

typedef struct {
  training_image_t* images;
  int size;
  int capacity;
} image_dataset_t;

image_dataset_t* alloc_image_dataset();
void free_image_dataset(image_dataset_t* dataset);
void add_training_image(SDL_Surface* image, int is_face, image_dataset_t* dataset);
image_dataset_t* fill_dataset(image_dataset_t* dataset);
```

## 5.3 La base de données

La base de donnée a été créé en SQLite. Nous pouvons l'utiliser basiquement pour l'instant. La base de donnée a été créé de façon a attribuer un unique identifiant à chaque personne. Nous pouvons actuellement créé la base de donnée, insérer un couple de valeur (Nom d'une personne et sa photo d'identité), supprimer un couple de valeur, afficher la base de donnée dans la console, tout supprimer.

### 5.3.1 La création

La fonction `create_db()` permet de créé la base de donnée avec la table Facificator. Pour l'instant sont créés : Name, Path. Path est le chemin relatif de l'image. D'autres champs seront créés selon notre besoin à venir.

### 5.3.2 L'insertion

La fonction `int insert(char *name, char *image)` permet pour l'instant d'insérer le nom d'une personne et sa photo d'identité. La photo d'identité doit être au préalable stocker dans Database_Pictures. Nous avons mis au point une fonction pour simplifié la copie de l'image, l'image doit être donnée en paramètre, dans ce dossier. Nous avons fait cela pour des raison évidente de conflit entre chemin absolu sur les différents ordinateur.

### 5.3.3 La suppression

La fonction `delete_id_db(char *id)` permet de supprimer une personne avec toutes les données associées. Pour faire cela nous avons juste besoin de mettre en paramètre l'id que la base de donnée à attribuer la base de donnée. Il faut savoir que l'id restera en "position" occupé.

### 5.3.4 L'affichage

La fonction `create_db()` permet d'afficher la base de donnée dans la console pour pouvoir avoir un visuel.

### 5.3.5 La destruction de la base

La fonction `int destroy_db()` permet de supprimer toute les données que la base contient actuellement, cependant les id ne seront pas réinitialisé.

### 5.3.6 L'insertion en masse

La fonction `int insert_folder (char *folder)` permet d'insérer de multiples personnes dans la base de donnée. Elle n'est pour l'instant pas entièrement fonctionnelle. Nous avons prévu de donner en paramètre le chemin absolu du dossier où ce trouve toutes les images. Les images seront ensuite copié dans notre dossier d'images. Les images devront porter le nom de la personne. La fonction va donc copier la première image et ensuite appelé la fonction insert avec le nom de l'image et le chemin de l'image copier.

## 5.4 La méthode de Viola et Jones

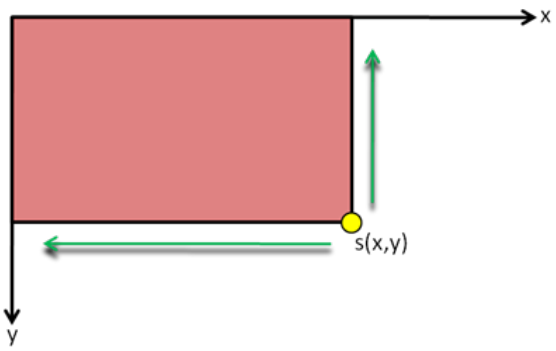
La méthode de Viola et Jones est basé sur un apprentissage supervisé. Des exemples d'objets doivent être analysés à l'avancé afin d'être classifiés. Ensuite, pendant l'analyse de l'image, des caractéristiques sont choisies par boosting ce qui permet de classifier les caractéristiques, et de séparer les exemples positifs des exemples négatifs par cascade de décision.

Nous avons choisi cette méthode car elle nous parait la plus pratique à mettre en œuvre et la plus optimisée.

### 5.4.1 L'image intégrale

La notion d'image intégrale permet de définir plusieurs zones rectangulaires au sein d'une image. L'intérêt de cette technique est qu'elle offre la possibilité d'accéder à la valeur des autres zones à gauche et au-dessus de la zone sur laquelle nous sommes. Ces zones permettent de créer des caractéristiques pseudo-Haar, qui sont en fait des masques permettant de déterminer plusieurs pattern.

L'image intégrale est utilisée comme un moyen rapide et efficace de calculer la somme des valeurs (valeurs de pixel) dans une image donnée. Si l'on souhaite utiliser l'image intégrale, il est préférable de s'assurer d'abord que l'image est en niveaux de gris. Quand on créé une image intégrale, on a besoin de créer un tableau. Dans ce tableau, en prenant n'importe quel point (x, y), on aura une certaine valeur. Cette valeur est la somme de toutes les valeurs de pixel de gauche, de dessus et bien évidemment, la valeur de pixel d'origine (x, y).

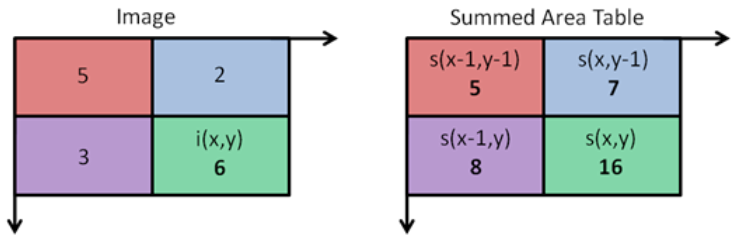



Le tableau peut être construit avec seulement un passage sur l'image donnée. En effet, chaque valeur du tableau est calculée grâce à l'équation suivante :

$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)$$

Cette équation signifie que l'on prend la valeur de pixel d'origine, puis on ajoute les valeurs se trouvant sur sa gauche avec s(x-1, y) puis ceux se trouvant au-dessus avec s(x, y-1) et enfin, on soustrait la valeur se trouvant au-dessus à gauche de la valeur de pixel d'origine avec s(x-1, y-1).

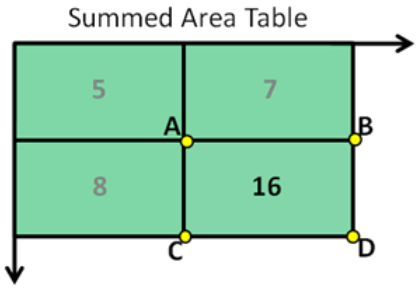
Voici un exemple concret d'un calcul :




Une fois que nous avons réussi à remplir le tableau, on peut aisément calculer n'importe quel sous-ensemble de l'image originale. Pour cela, il suffit simplement d'utiliser quatre valeurs du tableau. Avec ces quatre valeurs, on peut les additionner ou les soustraire entre eux afin d'obtenir la bonne valeur de pixel que l'on recherche. On utilise ainsi l'équation suivante :

$$i(x',y') = s(A) + s(D) - s(B) - s(C)$$

Voici un exemple :



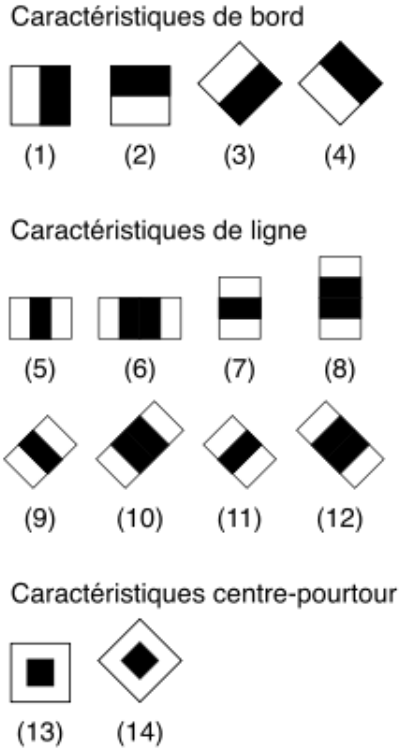

En utilisant l'équation, notre résultat devrait être égal à 6, ce qui est le cas, avec A = 5, B = 7, C = 8 et D = 16.

Pour cette soutenance, nous sommes capable de calculer l'image intégrale de n'importe quelle image et de la stocker dans une matrice. Les calculs de l'image integrale se situe dans pixel_operations.c et la structure integral_image contenant le champ int** integral_image (une matrice de la taille de l'image d'origine) se situe dans types.h.

### 5.4.2 Les caractéristiques pseudo-Haar

Les caractéristiques pseudo-Haar permettent de détecter des motifs. Par exemple, la reconnaissance des visages est rendue possible par exemple par la variation de l'intensité de la lumière entre les yeux et le nez ou la variation de l'intensité de la lumière entre les yeux et les pommettes. La méthode de Viola et Jones repose donc sur l'utilisation de ces caractéristiques pseudo-Haar et des images intégrales, améliorant ainsi la vitesse de traitement.

Pour cette soutenance, nous pouvons calculer l'ensemble des caractéristiques de n'importe quelle image de 24 x 24 pixels. Il y a 162 336 caractéristiques dans une telle image. Nous avons utilisés 5 types de caractéristiques: 3 caractéristiques de ligne et 2 caractéristiques de bord; cela est suffisant pour la reconnaissance faciale.



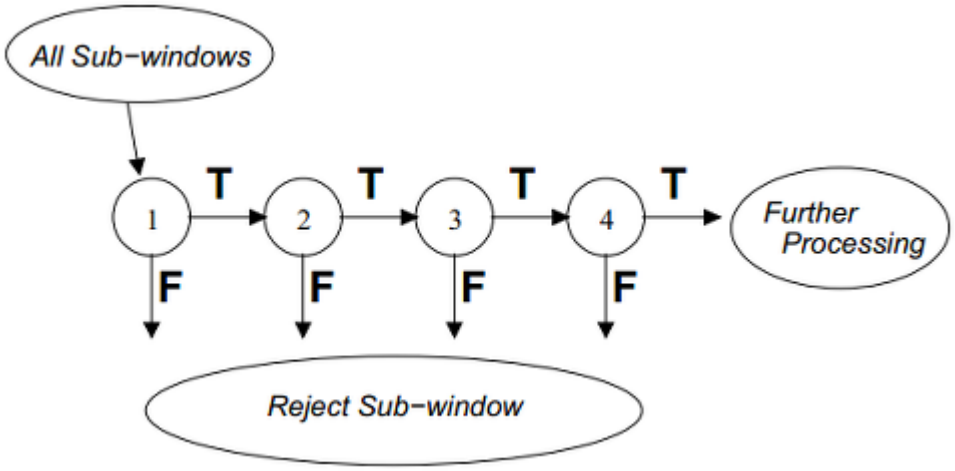
Le calcul de ces caractéristiques se situe dans pixel_operations.c et nous nous sommes aidés de cet algorithme :

**Algorithm 1** Computing a $24 \times 24$ image's Haar-like feature vector

**Input:** a $24 \times 24$ image with zero mean and unit variance
**Output:** a $d \times 1$ scalar vector with its feature index $\mathfrak{f}$ ranging from 1 to $d$
Set the feature index $\mathfrak{f} \leftarrow 0$
*Compute feature type* (a)
**for** all $(i, j)$ such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  **for** all $(w, h)$ such that $i + h - 1 \leq 24$ and $j + 2w - 1 \leq 24$ **do**
    compute the sum $S_1$ of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum $S_2$ of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    record this feature parametrized by $(1, i, j, w, h)$: $S_1 - S_2$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  **end for**
**end for**
*Compute feature type* (b)
**for** all $(i, j)$ such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  **for** all $(w, h)$ such that $i + h - 1 \leq 24$ and $j + 3w - 1 \leq 24$ **do**
    compute the sum $S_1$ of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum $S_2$ of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    compute the sum $S_3$ of the pixels in $[i, i + h - 1] \times [j + 2w, j + 3w - 1]$
    record this feature parametrized by $(2, i, j, w, h)$: $S_1 - S_2 + S_3$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  **end for**
**end for**
*Compute feature type* (c)
**for** all $(i, j)$ such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  **for** all $(w, h)$ such that $i + 2h - 1 \leq 24$ and $j + w - 1 \leq 24$ **do**
    compute the sum $S_1$ of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum $S_2$ of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    record this feature parametrized by $(3, i, j, w, h)$: $S_1 - S_2$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  **end for**
**end for**

### 5.4.3 L'apprentissage et la classification

Le classifieur permet de déterminer l'ensemble des zones rentrant sous la coupe d'une caractéristique pseudo-Haar, en déterminant les seuils pouvant déterminer les exemples positifs des négatifs. Ceci demande donc une phase d'apprentissage, qui permet de définir les seuils utilisés. Ensuite, un classifieur est une association entre une caractéristique pseudo-Haar et un seuil. Il existe plusieurs méthodes de boosting (apprentissage et classification) et pour notre projet nous avons choisi Adaboost car elle est très recommandée par sa simplicité a être codée.



### 5.4.4 Adaboost

Adaboost repose sur la sélection itérative de classifieurs faibles en fonction d'une distribution des exemples d'apprentissage. Chaque exemple est pondéré en fonction de sa difficulté avec le classifieur courant. Ensuite, la somme des classifieurs faibles et d'une constante (alpha dans notre code), calculée à partir de l'erreur minimale, nous donne le classifieur fort. Ce classifieur fort retourne 1 si l'image (24x24)est un visage et 0 sinon.

Nous utilisons une base d'apprentissage de 5000 images contenant un visage et de 5000 images ne contenant pas de visages (ce sont des training_image_t dans notre code). Ces images sont toutes de taille 24x24 pixels et sont en nuances de gris. La structure image_dataset_t (types.h) contiendra toutes ces training_image_t.

Notre Adaboost est capable de calculer les seuils pouvant déterminer les exemples positifs et négatifs; il calcule T classifieurs faibles, avec, pour chacun, l'erreur la plus minimale possible. Toutes les fonctions concernant Adaboost (que ce soit l'apprentissage ou la classification) se trouve dans le fichier adaboost.c. Nous avons utilise le multi-coeur pour l'apprentissage final. Ce qui a permis de faire l'apprentissage 8 fois plus rapidement que la normale car nous avons déployé 8 threads (une thread par coeur) lors de la sélection des caractéristiques.

```
const int N_THREADS = 8;
SDL_Thread* threads[N_THREADS];
const int features_per_thread = 162336 / N_THREADS;
for (int i = 0; i < N_THREADS; ++i) {
  int lower_feature = i * features_per_thread;
  int upper_feature = i == N_THREADS - 1 ? 162336 : (i + 1) * features_per_thread;
  threads[i] = thread_adaboost_train_feature(s_adaboost, lower_feature, upper_feature, dataset);
}
for (int i = 0; i < N_THREADS; ++i) {
  SDL_WaitThread(threads[i], NULL);
}
```

Nous nous sommes aider de cet algorithme pour réaliser notre version d'Adaboost en C :

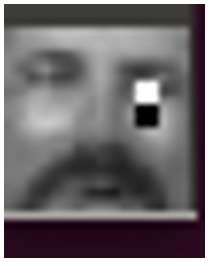
- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where $m$ and $l$ are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that $w_t$ is a probability distribution.
  2. For each feature, $j$, train a classifier $h_j$ which is restricted to using a single feature. The error is evaluated with respect to $w_t$, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, $h_t$, with the lowest error $\epsilon_t$.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example $x_i$ is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Voici des exemples de caractéristiques intéressantes que notre Adaboost a sélectionne :

Les yeux sont plus sombres que le front.

Le nez est plus clair que les yeux.

Encore une fois la différence entre le nez et l'œil droit.



L'œil est souvent plus sombre que la joue.

La barbe est également une caractéristique intéressante qui est plus sombre que le nez et la bouche.

### 5.4.5 La cascade de classifieurs

Nous nous sommes aider de cet algorithme pour réaliser notre cascade de classifieurs en C:

*Table 2.* The training algorithm for building a cascaded detector.

- User selects values for $f$, the maximum acceptable false positive rate per layer and $d$, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, $F_{target}$.
- $P$ = set of positive examples
- $N$ = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
  - $i \leftarrow i + 1$
  - $n_i = 0$; $F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use $P$ and $N$ to train a classifier with $n_i$ features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine $F_i$ and $D_i$.
    - Decrease threshold for the $i$th classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects $F_i$)
  - $N \leftarrow \emptyset$
  - If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N

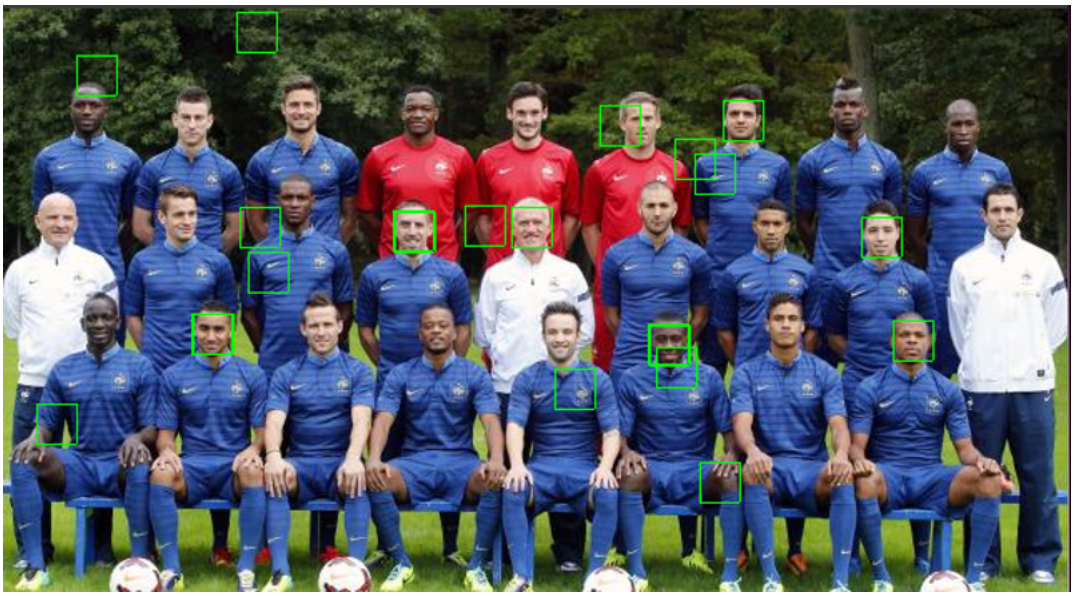
La cascade finale possède 10 niveaux de 2, 5, 10, 20, 25, 25, 30, 40, 50 et 60 classifieurs forts.

Voici le fichier qui contient notre cascade de classifieurs :

```
File Edit Options Buffers Tools Text Help
layer=1 strong_threshold=0.000001 nb_features=2
0 feature=75681 alpha=0.722176 threshold=46581444 parity=-1 j=5 i=1 w=15 h=4 charac=3
1 feature=53638 alpha=1.538520 threshold=-191852388 parity=1 j=2 i=5 w=6 h=5 charac=2
layer=2 strong_threshold=3.714108 nb_features=10
0 feature=22783 alpha=1.217510 threshold=-460551 parity=1 j=10 i=7 w=1 h=1 charac=1
1 feature=34967 alpha=1.074090 threshold=-3684408 parity=1 j=13 i=13 w=3 h=4 charac=1
2 feature=97830 alpha=1.034871 threshold=-131586 parity=1 j=4 i=9 w=1 h=1 charac=3
3 feature=94916 alpha=1.090693 threshold=-866362224 parity=1 j=0 i=8 w=21 h=5 charac=3
4 feature=151527 alpha=1.386142 threshold=723723 parity=1 j=10 i=6 w=1 h=1 charac=5
5 feature=94916 alpha=1.012619 threshold=-866362224 parity=1 j=0 i=8 w=21 h=5 charac=3
6 feature=97080 alpha=1.380879 threshold=-65793 parity=1 j=19 i=8 w=1 h=1 charac=3
7 feature=154200 alpha=1.116907 threshold=263172 parity=1 j=18 i=8 w=1 h=1 charac=5
8 feature=99020 alpha=1.038661 threshold=-250342365 parity=1 j=15 i=9 w=9 h=4 charac=3
9 feature=30108 alpha=1.433782 threshold=723723 parity=1 j=0 i=11 w=1 h=2 charac=1
layer=3 strong_threshold=11.373997 nb_features=15
0 feature=94592 alpha=1.430130 threshold=-12434877 parity=1 j=17 i=7 w=3 h=3 charac=3
1 feature=141650 alpha=1.253688 threshold=-6974058 parity=1 j=0 i=0 w=3 h=5 charac=5
2 feature=1296 alpha=1.062513 threshold=-197379 parity=1 j=5 i=0 w=1 h=1 charac=1
3 feature=36927 alpha=1.106509 threshold=328965 parity=1 j=2 i=15 w=1 h=1 charac=1
4 feature=94472 alpha=1.410826 threshold=-220011792 parity=1 j=15 i=7 w=6 h=4 charac=3
5 feature=20682 alpha=1.668287 threshold=42568071 parity=1 j=13 i=6 w=5 h=11 charac=1
6 feature=141860 alpha=1.311819 threshold=-1167365199 parity=1 j=1 i=0 w=7 h=11 charac=5
7 feature=99783 alpha=1.761151 threshold=855309 parity=1 j=3 i=10 w=1 h=1 charac=3
8 feature=141860 alpha=1.511378 threshold=-1167365199 parity=1 j=1 i=0 w=7 h=11 charac=5
9 feature=147576 alpha=1.758656 threshold=657930 parity=1 j=12 i=3 w=1 h=1 charac=5
10 feature=53638 alpha=1.538520 threshold=-191852388 parity=1 j=2 i=5 w=6 h=5 charac=2
11 feature=153308 alpha=1.816067 threshold=7237230 parity=1 j=2 i=8 w=5 h=1 charac=5
12 feature=53638 alpha=1.633174 threshold=-191852388 parity=1 j=2 i=5 w=6 h=5 charac=2
13 feature=3456 alpha=1.774096 threshold=2368548 parity=1 j=0 i=1 w=1 h=1 charac=1
14 feature=141704 alpha=1.711180 threshold=-1186247790 parity=1 j=0 i=0 w=9 h=9 charac=5
layer=4 strong_threshold=15.240194 nb_features=20
0 feature=20538 alpha=1.313188 threshold=1907997 parity=-1 j=12 i=6 w=1 h=4 charac=1
1 feature=157332 alpha=1.135034 threshold=-15461355 parity=-1 j=2 i=12 w=10 h=4 charac=5
2 feature=80132 alpha=1.236385 threshold=37765182 parity=-1 j=11 i=2 w=4 h=11 charac=3
3 feature=151354 alpha=1.331712 threshold=-149218524 parity=1 j=7 i=6 w=3 h=5 charac=5
4 feature=25136 alpha=1.556613 threshold=-263172 parity=1 j=10 i=8 w=1 h=1 charac=1
5 feature=154937 alpha=1.433231 threshold=-394758 parity=-1 j=10 i=9 w=1 h=1 charac=5
6 feature=22797 alpha=1.475513 threshold=-30988503 parity=1 j=10 i=7 w=1 h=3 charac=1
7 feature=89762 alpha=1.784586 threshold=-153034518 parity=-1 j=0 i=6 w=15 h=3 charac=3
8 feature=22797 alpha=1.632759 threshold=-30988503 parity=1 j=10 i=7 w=1 h=3 charac=1
9 feature=95189 alpha=1.818183 threshold=-4473924 parity=1 j=2 i=8 w=14 h=1 charac=3
10 feature=68847 alpha=1.543177 threshold=8882055 parity=1 j=19 i=17 w=1 h=1 charac=2
11 feature=57430 alpha=1.498296 threshold=31317468 parity=-1 j=6 i=7 w=4 h=2 charac=2
12 feature=57430 alpha=1.430099 threshold=30659538 parity=1 j=6 i=7 w=4 h=2 charac=2
13 feature=57424 alpha=1.467650 threshold=13355979 parity=-1 j=6 i=7 w=4 h=1 charac=2
14 feature=95189 alpha=1.432592 threshold=-100202739 parity=1 j=2 i=8 w=14 h=1 charac=3
15 feature=2182 alpha=1.967360 threshold=-2039583 parity=1 j=9 i=0 w=1 h=11 charac=1
16 feature=11127 alpha=1.774737 threshold=-44870826 parity=-1 j=5 i=3 w=4 h=7 charac=1
17 feature=7590 alpha=1.515619 threshold=-41054832 parity=1 j=3 i=2 w=5 h=8 charac=1
18 feature=16161 alpha=1.504832 threshold=-46713030 parity=-1 j=1 i=5 w=6 h=9 charac=1
19 feature=57143 alpha=1.628824 threshold=165600981 parity=1 j=3 i=7 w=4 h=7 charac=2
layer=5 strong_threshold=10.435910 nb_features=25
0 feature=17916 alpha=1.176145 threshold=2302755 parity=-1 j=12 i=5 w=1 h=5 charac=1
```

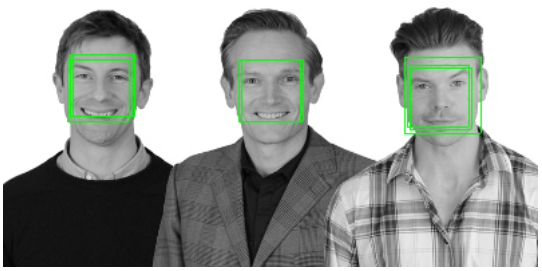
### 5.4.6 La détection

Voila le résultat de notre première cascade sur une photo de l'équipe de France de football :

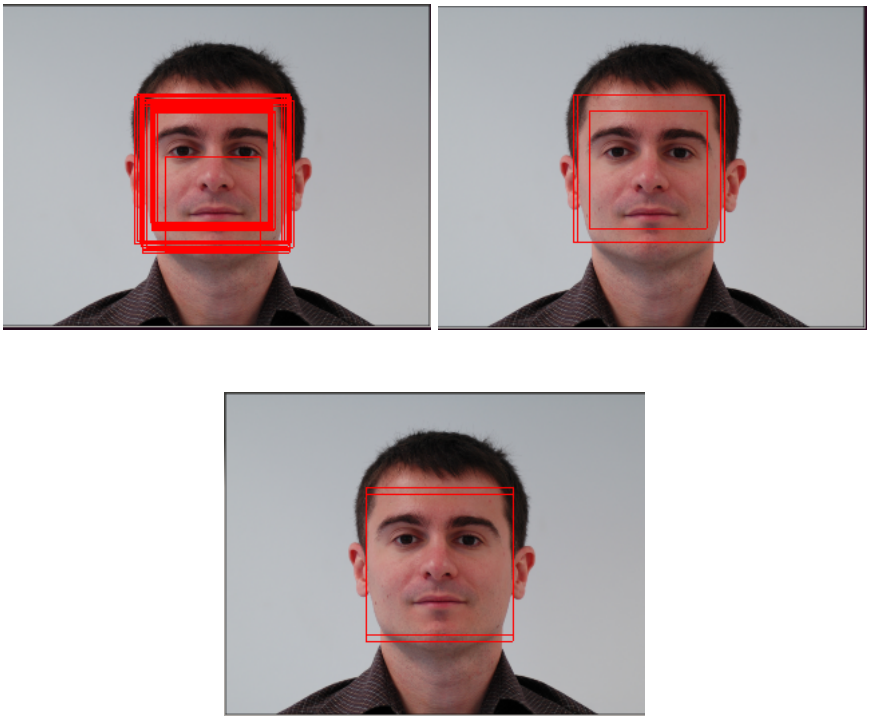


Nous voyons que quelques visages sont détectés mais il y a encore des "faux positifs", c'est-a-dire des non visages qui sont encadres.

Après 12 heures d'apprentissage et plusieurs versions d'adaboost et de cascade, nous avons obtenu une détection très fiable de visages vus de face.



Voici a présent la différence de détection avec 5, 8 et 10 niveaux de notre cascade finale :



Nous remarquons que plus il y a de niveaux, plus la détection va être précise.

## 5.5 Les eigenfaces

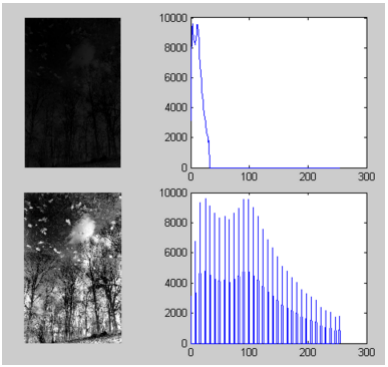
Alors que la détection de visage est mis en place la suite logique et leurs reconnaissance. Afin de reconnaitre ces visage plusieurs calcul sont a effectuer afin de déterminer si oui ou non un visage soumit appartient a une base de donnée. mais avant de commencer les calcul en eux même il faut effectuer un pretraitement de l'image.

### 5.5.1 Pré traitement

Le pré traitement vise en le redimensionnement, l'égalisation et la normalisation de l'image afin de permettre une cohérence des résultats de calculs sur tout visage soumis au logiciels. Chaque visage est soumis au logiciel sous forme de matrice des pixels qui le compose.

- Redimensionnement : Vise à rogner l'image dans une taille standard afin de rendre possible certains calculs

- Normalisation : La normalisation consiste à améliorer le contraste d'une image à partir de son histogramme. Cela consiste à repartir de manière équitable les nuances de gris sur la globalité de l'image exemple:




Sur l'ordonnée on constate l'intensité du niveau de gris et en abscisse le nombre de pixel possédant une nuance en particulière ainsi une meilleure répartition horizontale améliore le contraste.

- L'égalisation : Consiste à réduire la différence d'amplitude entre les niveaux de gris

### 5.5.2 Matrice de covariance et valeurs propres

Maintenant que les visage sont pretraités les calculs d'identification peuvent commencer. Ainsi la première étage consiste à calculer un visage moyen à partir de l'ensemble des visages contenus dans la base de données. ce visage G est calculé par

$$G = \frac{1}{N} \sum_{k=1}^{N} V_i$$

avec N le nombre de visage de la base de donne et V un visage de celle ci
A partir de ce visage générique on calcule la différence entre chaque visage et le visage moyen. Notons T ce vecteur. Les calculs sont les suivants .

$$U_i = V_i - G$$

$$T = \{U_1, U_2, ..., U_N\}$$

A partir de ce vecteur nous pouvont créer une matrice A en concaténant chaque élément de celui-ci. Ensuite il nous faut calculer la matrice de covariance C donner par la formule suivante

$$C = AA^t$$

Cette matrice de covariance nous permet de calculer une valeur propre P grâce à la limite suivante

$$P = \lim_{n \to +\infty} W_n$$

$$W_n = \frac{C^n * X}{|C^n * X|}$$

Ou X est un vecteur colonne arbitraire de même taille que C

Cette valeur propre sera comparer à celle de l'image à analyser et si elle sont similaire cela signifie que le visage appartiendra la base de données. La similarité est déterminée par un seuil d'erreur définis arbitrairement.

## 5.6 L'interface graphique

L'interface graphique permet un contrôle du programme de manière beaucoup plus intuitive que du simple code. Ce qui est d'une grande importance étant donné que toute la population n'est pas experte en informatique. Ainsi le développement d'une interface graphique permet à de potentiels clients ou collaborateurs d'exploiter le programme sans avoir à passer une formation spécialisée.

### 5.6.1 La programmation événementiel

L'interface graphique permet le déclenchement d'événements en fonction de l'utilisateur. Or, la programmation classique, c'est-à-dire l'exécution d'algorithmes de manière séquentielle, ne permet pas ce genre d'interactions. C'est pourquoi une nouvelle philosophie de programmation a vu le jour. Tout comme la programmation orientée objet, la programmation événementielle possède ses propres règles.

La structure se décompose en une boucle générale dans laquelle toutes les possibilités d'actions de l'utilisateur sont prises en compte dans laquelle se trouvent différents modules. Dans cette boucle plusieurs modules, appelés widgets (”chose” en anglais), sont définis. Ces widgets permettent l'affichage de différentes fenêtres, boutons, textes etc. Pour que l'utilisateur puisse interagir avec les différentes fenêtres définies plus tôt, une nouvelle notion devra être intégrée : les fonctions callback. Ces fonctions ont un but très simple : déclencher un événement (ouverture d'une fenêtre, affichage de texte etc.) lorsque l'utilisateur effectue une action précise (survole avec le curseur, clique sur un bouton). Pour effectuer cette tâche, les fonctions callback possèdent toutes la même structure, trois paramètres : un widget, une action et une fonction. Le paramètre widget sert a indiquer quel objet devra subir l'action pour déclencher l'événement, le paramètre action définit le type d'action (clique, survole...) et enfin le dernier paramètre est une fonction. Cette fonction décrit l'ensemble des instructions à effectuer si l'action est réalisée.

La combinaison des widgets et des fonctions callback permettent la création d'une interface graphique très complète et modulable à souhait.

### 5.6.2 La forme

L'interface graphique contient tout d'abord une barre de menu contenant 4 sous-menus, ”Fichier”, ”Édition”, ”Base de donnée” et ”Aide” et un cadre affichant une image de base. Mais également quatre boutons, ”Charger une image”, ”Détection”, ”Ajout” et ”Suppression”.

### 5.6.3 Le fond

Pour chacun des boutons, le signal ”clicked” a été associé à une fonction spécifique. Concernant le bouton ”Charger une image”, lorsque l'on clique sur le bouton, la fonction qui a été connecté à ce bouton, grâce au signal ”clicked”, permet d'afficher une nouvelle fenêtre. Cette nouvelle fenêtre contient un bouton permettant de sélectionner l'image que l'on souhaite et de l'afficher sur la fenêtre principale tout en fermant la fenêtre secondaire qui avait été ouverte.

Concernant le bouton ”Détection”, lorsque l'on clique sur le bouton, la fonction qui a été connecté à ce bouton, grâce au signal ”clicked” aussi, va lancer l'algorithme adaboost, qui va ainsi permettre de lancer la détection de visage sur l'image que l'on aura au préalable charger grâce au bouton présenté précédemment.

Concernant le bouton ”Ajout”, lorsque l'on clique sur le bouton, la fonction qui a été connecté à ce bouton, grâce au signal ”clicked”, permet d'afficher une nouvelle fenêtre

# 6 Conclusion

Les premières semaines de projet avaient été très motivantes. Toutes les difficultés que nous avions rencontrées avaient été surmonté. De plus, nous avions réussi à respecter notre planning, voir sur certaines parties nous avions pris une légère avance. Nous espérions garder cette avance pour pouvoir surmonter toutes les difficultés qui nous attendaient. Cependant, l'avance pris après la première soutenance fut très vite dissiper. En effet, il s'avère que le projet fut beaucoup plus difficile que ce que nous pensions. Mais nous sommes malgré la difficulté parvenus à obtenir un logiciel permettant la reconnaissance de visage dans une photo, c'est-à-dire de nous indiquer si un visage est présent ou non sur une photo. Malgré que le projet ne soit pas totalement fini car, les objectifs n'ont pas été tous atteint, nous pouvons dire que nous sommes assez fier du logiciel que nous avons obtenu.