Table des matières

Table des figures

Liste des abréviations

DAB : Discrete AdaBoost
FP : False Positive
GAB : Gentle AdaBoost
HSV : Hue Saturation Value
IHM : Interaction Homme Machine
LS : Learning Samples
OpenCV : Opensource Computer Vision
RAB : Real AdaBoost
RVB : Rouge Vert Bleu
SVM : Support Vector Machine
TP : True Positive
TSV : Teinte Saturation Valeur
WE : Fonction d'Emphasis pondérée
WL : Weak Learner

Introduction Générale

La détection et le suivi d'un visage à partir d'un flux vidéo est problématique en plein essor depuis plus de deux décennies dans la communauté de vision par ordinateur. Deux raisons principales expliquent un tel effort de recherche :

1. L'explosion des champs d'applications envisageables : la vidéosurveillance, la télémédecine, la téléconférence, l'interaction homme machine (IHM) destinée à de nouvelles interfaces utilisateurs, la robotique, illustrent un grand panel des utilisations possibles.
2. La détection et le suivi par des algorithmes temps-réel sont éventuellement requis, comme tâches préalables, à d'autres algorithmes pour éffectuer des calculs sur le visage comme la reconnaissance, l'identification ou l'étude des expressions.

Malgré l'énorme progrès dans le domaine de la vision par ordinateur et l'augmentation de la vitesse des processeurs, ainsi l'apparition des processeurs spécialisés (processeur de traitement de signal, les processeurs graphiques), des algorithmes génériques de détection visage n'ont toujours pas été dégagés. De plus, le suivi robuste d'un visage à partir d'un flux vidéo demeure insaisissable. En effet, il reste de nombreux verrous scientifiques. Notamment celui de surmonter la forte variabilité d'apparence du visage au cours du temps. Cette variabilité dépendent de plusieurs facteurs :

- La position : sur une image, un visage peut être vu de face, de profil, ou d'un angle quelconque.
- L'expression faciale : l'apparence d'un visage dépend aussi de son expression.
- La présence d'attributs : une personne peut avoir une casquette, voile, des lunettes, une moustache, une barbe, une cicatrice, cheveux teintés,...
- Les conditions extérieures : la couleur, l'intensité, la taille, la texture sont différentes sur chaque image.
- L'occultation : une partie du visage peut être cachée par un autre objet ou une autre personne.

La détection de visage peut être définie comme : Etant donnée une image, le but est de déterminer si un ou des visages sont apparents dans l'image et s'il y en a, de localiser chacun des visages. Les techniques de la détection de visages sont développées et employées dans plusieurs domaines, surveillance, identification, biométrie, etc. Les méthodes de la détection de visages peuvent être classifiées en quatre catégories

— Méthodes par a priori. Ces méthodes basées sur des règles tentent de modéliser la connaissance de ce qui caractérise un visage. Classiquement, ces règles représentent des relations en caractéristiques faciales.
— Approches par caractéristiques invariantes. Ces approches se basent sur des caractéristiques structurelles qui existent même quand la pose, le point de vue, ou les conditions d'illumination varient, et les utilisent pour localiser les visages.
— Méthodes basées modèles. Plusieurs modèles standard de visages sont utilisés pour définir un modèle de visage ou des modèles de caractéristiques faciales séparément. La corrélation entre une image présentée et la base des modèles est évaluée pour détecter la présence de visage.
— Méthodes par apprentissage. Par contraste avec les méthodes basées modèles, les modèles sont ici appris à partir d'un ensemble d'images d'apprentissage qui doivent permettre de caractériser la variabilité de l'apparence d'un visage. Ces modèles appris servent ensuite à la détection.

Dans notre travail, nous proposons une approche précise, robuste et rapide de détection de visages dans les plans-séquences combinant les avantages de deux méthodes : un algorithme de détection d'objet par AdaBoost [16] devenu une référence de détection des visages et la poursuite des visage dans une sceine par le filtre particulaire.

CHAPITRE 1

Chapitre1 : Méthodes d'apprentissage en traitement des images vidéo

1.1 Introduction

En vision par ordinateur, la détection de visage a pour finalité la détection et la localisation d'un nombre inconnu de visages dans une image fixe ou dans un flux vidéo. Liés à l'essor de l'Intelligence Artificielle, les premiers travaux abordent la question de détection de visage dès la fin des années soixante [4]. Les premières approches utilisaient des techniques heuristiques et anthropométriques simples afin de détecter des visages dans les images numériques avec diverses hypothèses simplificatrices (photographie de passeport avec un fond uni et le visage de face). Cependant ces hypothèses induisent souvent un manque de flexibilité, de robustesse et répondent trop partiellement à la problématique de la détection de visage. Ces difficultés peuvent être justifier par :

- les différences des caractéristiques morphologiques propres à l'individu (forme du nez, couleur des yeux, couleur de peau, etc.).
- la diversité des aspects du visage liée aux changements d'expression de pose.
- la présence éventuelle d'artefacts visuels (chapeau, lunettes, ou la barbe, moustache, tatouage, cicatrice,. . .).
- les modifications des conditions de la scène dues aux variations d'éclairage et du fond.
- la présence d'occultations.

Un très grand nombre d'approche proposés pour résoudre le problème de détection de visage peuvent être classer en deux grandes catégories, différentes par leur considération de l'information a priori du visage [3] :

1. **Méthodes basées connaissances à priori** [11] : s'appuient explicitement sur les connaissances du visage (la couleur de la peau, les relations spatiales entre la bouche, le nez, les deux yeux, template prédéfini, template déformable).
2. **Méthodes basées apprentissage** [25] : considèrent le problème de détection de visage comme un problème de reconnaissance de forme à deux classes : visage et non-visage, où chaque classe est représentée par une base d'apprentissage. Ces méthodes s'appuient sur des apprentissages statistiques permettant de construire une fonction de décision qui intègre implicitement les caractéristiques d'un visage.

Ces techniques voient la détection de visage comme un problème général d'identification, mettent en valeur les propriétés globales de la forme et traitent le visage comme un tout. Sans extraction ni analyse de composantes (le nez, les yeux , la bouche), l'approche holistique est basée sur l'apprentissage d'un modèle de visage à partir d'une base d'exemples.

Parmi ces deux grandes classes, les approches basées apprentissage ont prouvé leur supériorité en terme de performance et de robustesse. C'est pourquoi nous nous intéressons à ces approches dans la suite de ce chapitre.

1.2 Méthodes basées apprentissage

Dans le cadre de la détection de visage, une méthode basée apprentissage s'appuient sur des images d'apprentissage pour construire un modèle permettant de discriminer des instances de la classe du visage par rapport à toutes les instances de la classe non-visage. Les performances d'une telle méthode sont conditionnées par la qualité de la base d'apprentissage qui se doit d'être la plus représentative possible. En effet, cette base doit permettre de capter la variabilité d'apparence présente au sein de la classe

visage et non-visage. La constitution de la base d'apprentissage est un point important et est abordée ultérieurement. Une fois la base d'apprentissage constituée, un algorithme d'apprentissage est appliqué sur l'ensemble des images d'apprentissage ou sur des descripteurs associés. Différents algorithmes d'apprentissage supervisé et non-supervisé en été proposé dans la littérature comme des SVM (Machine à Vecteur Support :Support Vector Machine) [4] , des réseaux de neurones [12], des réseaux bayésiens [15] ou des classifieurs bayésiens naïfs [10], boosting [15], et réseaux de neurones convolutionnels[20]. Le travail le plus marquant pour la détection de visage en temps-réel est celui de Viola et Jones [4]. Ce dernier a utilisé des descripteurs simples ainsi qu'une technique rapide de calcul de ces descripteurs appeler boosting. L'algorithme de boosting est utilisé à la fois pour sélectionner les meilleurs descripteurs mais aussi pour former un ou plusieurs classifieurs forts associés en cascade pour permettre une détection de visage en temps-réel.

1.2.1 Méthodes basées boosting

Les performances du détecteur de Viola et Jones [24] sont dues à la combinaison de trois (03) éléments qui seront étudier par la suite :

1. un apprentissage de classifieurs par une méthode de boosting .
2. des descripteurs simples et rapides à calculer .
3. une structure de classifieurs en cascade.

Les algorithmes de boosting

Les algorithmes de boosting sont destinés à un apprentissage supervisé, i.e. qui utilisent une base d'apprentissage labellisée. Notons par $B = \{(x_i, y_i) \in \mathbb{R}^{n \times n}\}_{i=1,\ldots,N}$ la base d'apprentissage ou x_i est un vecteur de données représentant un exemple d'apprentissage et $y_i \in \{-1, 1\}$ est le label associé à x_i (généralement, le label 1 représente la classe des visages et le label -1 la classe des non-visages). Le but de ces méthodes est de construire un classifieur $H(\mathbf{x}) : \mathbb{R}^n \to \{-1, 1\}$: permettant d'associer un label à un visage inconnu. Des règles de décision précises sont générées en utilisant des règles de décision produites par des classifieurs faibles, i.e. des classifieurs ayant un taux de réussite un peu meilleur que le hasard. D'une manière générale, l'algorithme de Boosting s'inspire d'un concept très simple : il est rare d'avoir à sa disposition un expert omniscient permettant de prendre la meilleure décision et par conséquent, on a plutôt recours à un comité d'experts plus ou moins compétents pour ensuite combiner leurs avis et prendre une décision [1].
Le premier algorithme de Boosting a été proposé par Schapire [1] et permet d'obtenir un classifieur après avoir entraîné un classifieur faible sur trois sous-ensembles des données d'apprentissage. Cet algorithme a été amélioré par la suite en ajoutant deux autres critères :

1. La pondération adaptative des votes par une technique de mise à jour multiplicative.
2. La modification de la distribution des exemples disponibles pour entraîner chaque classificateur, en surpondérant au fur et à mesure les exemples mal classés.

De la même manière que la SVM [10], les méthodes de boosting sont issues de considérations théoriques, ce qui permet de connaître certaines propriétés. En particulier, on sait que l'erreur d'apprentissage diminue exponentiellement même si le classifieur faible. Cependant, la prédiction de l'erreur globale est plus difficile à prédire.

1.3 Descripteur de primitive

Nous avons assisté ces dernières années à l'émergence de plusieurs algorithmes d'apprentissage qui utilisent la notion de descripteur de primitive[27]. Ce dernier désigne tout vecteur de données $\mathbf{x} \in \mathbb{R}^n$qui permet de donner une description d'une ou une partie d'une image à l'aide des outils statistique ou non comme l'histogramme locale, moyenne, variance,... Pour une image en niveaux de gris de taille $W \times H$ et φ une image de caractéristiques de taille $W \times H \times d$ extraite à partir de I :

$$\varphi(\mathbf{p}) = \Phi(I, \mathbf{p}) \tag{1.1}$$

où Φ est une fonction qui associe à chaque pixel $\mathbf{p}$ de coordonnées $(\mathbf{x}, \mathbf{y})$ de I un ensemble de d caractéristiques $\varphi(\mathbf{p}, i)$ ($\varphi(\mathbf{p}, i)$ est donc la ième caractéristique du pixel $(\mathbf{x}, \mathbf{y})$). Une caractéristique est une donnée caractérisant un pixel comme son intensité, sa couleur dans l'espace HSV ou son gradient. Une fois l'image caractéristique calculé φ , on peut calculer n'importe quel vecteur de descripteurs $\mathbf{x}$ associé à une région Ω de I :

$$\mathbf{x} = f(\varphi, \Omega) \tag{1.2}$$

où Ω est défini par les coordonnées du coin supérieur gauche, une largeur et une hauteur. Le type de descripteur utilisé dans ce cas est représenté par f. Ce Processus est illlustré sur la figure ci-dessous. Dans leurs travaux, Viola et Jones proposent d'utiliser comme caractéristique l'intensité des pixels de I, on a donc $\varphi(\mathbf{p}) = I(\mathbf{p})$, $(\mathbf{x}, \mathbf{y})$. Les valeurs d'un pixel ne nous informent que sur la luminance et la couleur d'un point donné. Il est donc plus judicieux de trouver des détecteurs fondés sur des caractéristiques plus globales de l'objet. C'est le cas des descripteurs de primitive [16]. Les descripteurs de primitives (features) sont des fonctions permettant de connaître la différence de contraste entre plusieurs régions rectangulaires contiguës dans une image. On code ainsi les contrastes existants dans un visage et les relations spatiales (Figure.1.1). En effet, ces descripteurs permettent de calculer la différence entre la somme des pixels dans les zones blanches et la somme des zones noires. La valeur de descripteur est calculée par :

$$f_i = Sum(r_{i,blanche}) - Sum(r_{i,noire}) \tag{1.3}$$

Ces descripteurs sont calculés dans une fenêtre de taille fixe (ex. 24x24 pixels).

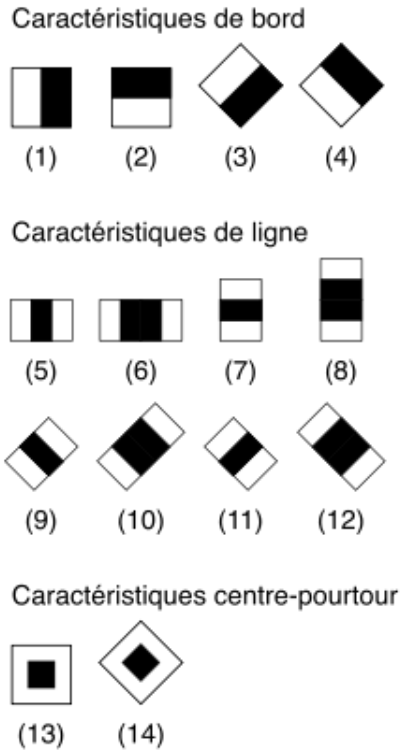

FIGURE 1.1 – Descripteurs de Primitive

Généralement, ils sont classifiés en 3 catégories : 2-rectangles, 3-rectangles et 4-rectangles descripteurs (Figure.1.2). Les 2-rectangles descripteurs sont utilisés horizontalement et verticalement(Figure.1.2)(1) et (2). Les régions blanches ont des poids positifs et les régions noires ont des poids négatifs [26].

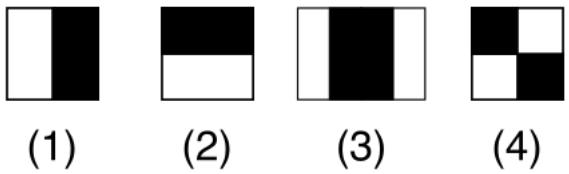

FIGURE 1.2 – Descripteurs de primitive de Haar dans une fenêtre : 2-,3-,4-rectangles détecteurs

Un descripteur de primitive est caractérisé par :

- Le nombre de rectangles (2,3 ou 4)
- La position (le sommet supérieur gauche) (x,y) de chaque rectangle
- la largeur w et la hauteur h de chaque rectangle avec :

$$0 < x, x + w < W; 0 < y, y + h < H$$

- Les poids positifs ou négatifs de chaque rectangle

Un exemple est donné sur la (Figure.1.3).

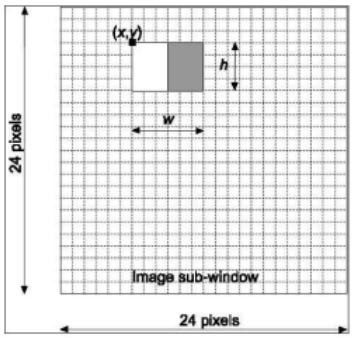

FIGURE 1.3 – Définition d'un descripteur de primitive de Haar dans une fenêtre

Les descripteurs de primitive sont très simples mais très nombreux du fait des variations de taille et de position. Etant donné une fenêtre de résolution 24x24 pixels, on peut définir environ 160 000 détecteurs possibles dans cette fenêtre selon [16], sans compter les descripteurs " 45 dégrées " proposés par [19].Quelques exemples sont montrés dans la (Figure.1.4). La nécessité de balayer pour chaque sous-fenêtre tous les pixels de l'image est un processus trop coûteux en temps. L'idée d'image intégrale est donc introduite afin d'accélérer le calcul [16].

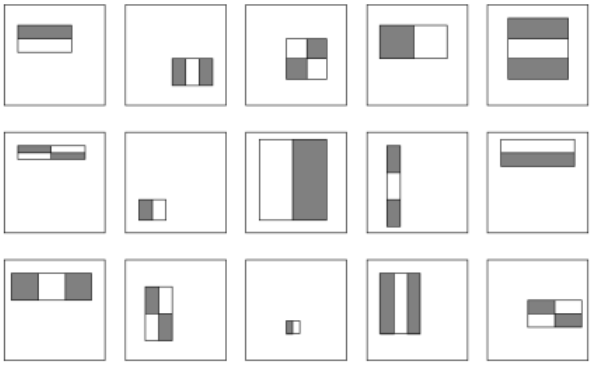

FIGURE 1.4 – Exemples des descripteurs de primitive de Haar dans une fenêtre 24x24

1.3.1 Image Intégrale

L'image intégrale est une nouvelle représentation introduite par Viola et Jones[17] pour calculer rapidement les attributs du descripteur. L'idée est de calculer seulement une fois la somme de tous les pixels de l'image [16]. Le pixel **p** à la position (x, y) de l'image intégrale contient la somme de tous les pixels, de l'image initiale, supérieurs et à gauche de la position (x, y) (voir Figure 1.5).

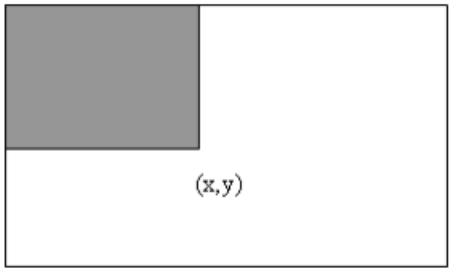

FIGURE 1.5 – Image Intégrale

Soient l'image intégrale $I_{int}(x, y)$ et l'image originale $I(x, y)$, l'image intégrale s'obtient par : $I_{int}(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$ Avec l'image intégrale, à partir de $I_{int}(x, y)$ n'importe quel rectangle de l'images peut être calculé avec 4 points :

- 2 = A + B
- 3 = A + C
- 4 = A + B + C + D
- D = 4 + 1 -(2 + 3)

au lieu de l'addition de touts les pixels de région D (Figure1.6). Cela nous permet de réduire le calcul en temps constant [26].

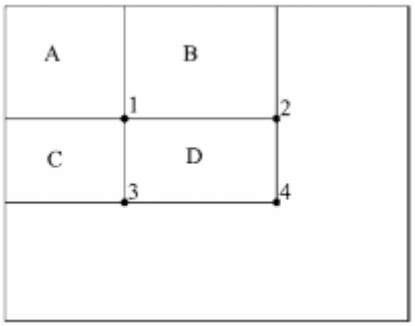

FIGURE 1.6 – Calcul de la somme du rectangle D avec l'image intégrale

1.4 Classifieur en cascade

La détection de visage dans une image passe par le concept de fenêtre glissante. En effet, cela consiste à tester différentes fenêtres de l'image dont la taille et la position varient. Ces fenêtres sont ensuite introduites dans un classifieur pour déterminer la présence éventuelle d'un visage. Ce classifieur pourra être entraîné à l'aide d'un algorithme de boosting et pourra être appliqué sur toutes les fenêtres potentielles. Le but est d'avoir un rapport vrais positifs(True Positive :TP)/faux positifs(False Positive :FP) très élevé. Le taux de TP correspond au nombre de visages correctement classés par rapport au nombre total de fenêtres testées et le taux de FP correspond au nombre de fenêtres négatives classées comme visage par rapport au nombre total de fenêtres testées. Pour atteindre cet objectif, une solution consiste à avoir un grand nombre de classifieurs faibles associes à chaque classifieur fort. Cependant, le cout de la classification de toutes les fenêtres de l'image devient très élevé. Pour résoudre ce problème, Viola et Jones [16] proposent d'utiliser plusieurs classifieurs forts successifs organisés en cascade.

Les classifieurs forts des premiers niveaux sont chargés de rejeter les négatifs les plus simples alors que les classifieurs forts des derniers niveaux essayent de discriminer les visages des négatifs les plus difficiles (ceux qui ressemblent aux positifs). Le classifieur fort du niveau j est de la forme :

$$sign\left(H_j\left(\mathbf{x}\right)-\tau_j\right)=sign\left(\sum_{i=1}^{T_j}\alpha_{ji}h_{ji}\left(\mathbf{x}\right)-\tau_j\right) \tag{1.4}$$

où τ_j est un seuil fixé pendant l'apprentissage pour obtenir un taux de vrais positifs très élevé d_{min} tout en assurant un taux de faux positifs modeste f_{max}. Le taux de détection D d'une cascade de L niveaux ainsi que son taux de faux positifs F.

$$D=\prod_{j=1}^{L}d_{\min} \tag{1.5}$$

Et

$$F=\prod_{j=1}^{L}f_{\max} \tag{1.6}$$

Pour classer une fenêtre, il suffit d'appliquer les classifieurs forts successifs tant que la fenêtre est classée comme visage. Dès qu'un classifieur fort classe la fenêtre comme non-visage, le processus s'arrête. Cette approche, illustrée sur la figure ci-dessous, permet de rejeter la majorité des fenêtres dans les premiers niveaux de la cascade, ce qui assure un temps d'exécution faible, tout en conservant un maximum de visages, ce qui assure des performances élevées. Une cascade est ainsi définie par l'ensemble des classifieurs.

1.4.1 Classifieur faible et classifieur fort

Avec les descripteurs de primitives, un ou plusieurs classifieurs faibles peuvent être formé. Un classifieur faible $h(x)$,composé d'un descripteur f, d'un seuil θ , et d'une parité ρ, donne une prédiction sur la classe visage ou non-visage (1 pour visage et -1 pour non visage).

$$h(x,f,\rho,\theta)=\begin{cases}1\ if\ \rho f(x)<\rho\theta\\0\ otherwise\end{cases} \tag{1.7}$$

L'algorithme de Boosting vise à combiner plusieurs classifieurs faibles pour obtenir un classifieur fort plus efficace [16]. Etant donné les exemples négatifs et positifs, les poids de chaque exemple sont initialisés. Par la suite, le processus de boosting est appliqué : T itérations sélectionnant T classifieurs faibles $h_t(x)$ avec chacun un poids α_t pour obtenir finalement un classifieurs fort H(x). Pour chaque itération, un classifieur est choisi en minimisant le taux d'erreur calculé avec les poids courant des exemples. Par la suite les poids des exemples mal classifiés sont augmentés pour la sélection suivante. L'itération se termine quand le taux de bonnes détections et le taux de faux positifs atteignent le compromis choisi au départ. Cette démarche est décrite par l'algorithme ci-dessous :

Algorithm 1 Algorithme de Boosting

Require: $(x_1, y_1), ..., (x_n, y_n)$ where $y_i = -1, 1$ for negative and positive examples respectively.

Require: Initialize weights $w_{l,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = -1, 1$ respectively , where m and l are the number of negatives and positives respectively.

while $t<T$ **do**

$w_{t,i} \longleftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

$\epsilon_t = \min_{f,\rho,\theta} \sum_i w_i \mid h(x_i, f, \rho, \theta) - y_i \mid$

$h(x_t = h(x, f_t, \rho_t, \theta_t)$ where f_t, ρ_t,and θ_t are the minimizers of ϵ_t

$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$

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 : $H(x) = \begin{cases} 1 \ \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \ Otherwise \end{cases}$

where $\alpha_t = log\frac{1}{\beta_t}$

end while

Les classifieurs de précision plus élevée (taux d'erreur entre 0.1 et 0.3) sont sélectionnés au début de l'apprentissage, et les classifieurs moins précis (taux d'erreur entre 0.4 et 0.5) sont sélectionnés dans les dernières itérations. La (Figure1.7) présente deux exemples de descripteurs les plus discriminants sélectionnés par l'algorithme de boositng à partir d'une base d'images de visages [16]. Le premier descripteur caractérise la différence d'intensité entre la zone des yeux et la zone des pommettes. Le second descripteur mesure la différence d'intensité entre les yeux et la zone au dessus du nez [26].

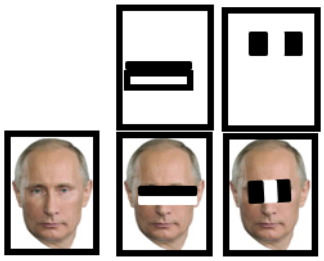

FIGURE 1.7 – Deux descripteurs de Haar les plus discriminants

1.4.2 Mise en oeuvre de classifieur

Pour réaliser un traitement efficace, il est nécessaire d'avoir l'avis de plusieurs classifieurs forts (Figure1.8). Une cascade de classifieurs est un arbre de décision dégénéré dans laquelle, chaque étape est entraînée pour détecter un maximum d'objets intéressants tout en rejetant une certaine fraction des objets non-intéressants.
La structure de la cascade reflète le fait que l'image est constituée majoritairement de sous-fenêtres négatives. Une sous-image doit passer tous les classifieurs afin d'être acceptée

comme visage. Le déclenchement de tous les classifieurs par un résultat positif devient ainsi un événement rare. Il faut que le nombre de sous-images éliminées dès les premières étapes de la cascade soit très élevé.
La cascade des classifieurs commence par un taux de détection de presque 100% mais avec un taux de faux positifs élevé. Il diminue rapidement avec quelques itérations. Ainsi on rejette immédiatement un grand nombre de régions négatives, donc cela accélère l'apprentissage et la détection [26].

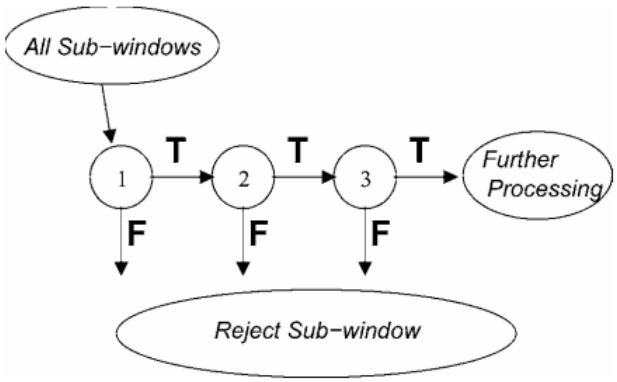

FIGURE 1.8 – Cascade de classifieurs forts

Pour une cascade, le taux global de faux positifs (fenêtres négatives déclarées comme positives) est le produit du taux de faux positifs de chaque étape :

$$F = \prod_{i=1}^{K} f_i$$

Le taux global de bonnes détections est défini selon la même formule :

$$D = \prod_{i=1}^{K} d_i$$

Par exemple pour obtenir une cascade avec un taux global de bonne détection de 0.9 et un taux global de faux positifs de 6×10^{-6}, on peut entraîner 10 étapes avec $d_i = 0.99\ et f_i = 0.3$.
Dans [16] un algorithme par cascade a été présenté pour effectuer l'apprentissage des exemples visage, non visage. Le taux maximum de faux positifs pour une étape f est de 90% , le taux minimum de détection pour une étape de t est de 60% le taux global de faux positifs acceptés F_{target} sont définis à l'entrée de l'apprentissage. Les étapes de la cascade sont construites par entraînements successifs de classifieurs avec Boosting puis ajustement de leur seuil afin de minimiser le nombre de faux positifs. Chaque étape est entraînée en ajoutant des classifieurs faibles jusqu'à ce que les taux de détection demandés soient atteints. Des étapes sont ajoutées à la cascade jusqu'à ce que les taux de la cascade entière soient atteints. Après chaque étape, on diminue l'ensemble des exemples négatifs et on ne garde que les exemples mal classifiés pendant la dernière étape.

Algorithm 2 Algorithme d'apprentissage par une cascade de boosting

Require: User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer
Require: User selects target overall false positive rate, F_{target}
Require: P=set of positive examples
Require: N=set of negative examples
Require: $F_0 = 1.0;\ D_0 = 1.0$
Require: $i = 0$
  while $F_i > F_{target}$ **do**
    $i \longleftarrow i + 1$
    $n_i = 0;\ F_i = F_{i-1}$
    while $F_i > f \times F_{i-1}$ **do**
      $n_i \longleftarrow n_i + 1$
      Use P and N to train a classifier with n_i features using Boosting.
      Evaluate current cascaded classifier on validation set to determine F_i and D_i.
      Decrease threshold for the i the classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$(this also affects F_i)
      $N \longleftarrow \emptyset$
      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
    end while
  end while

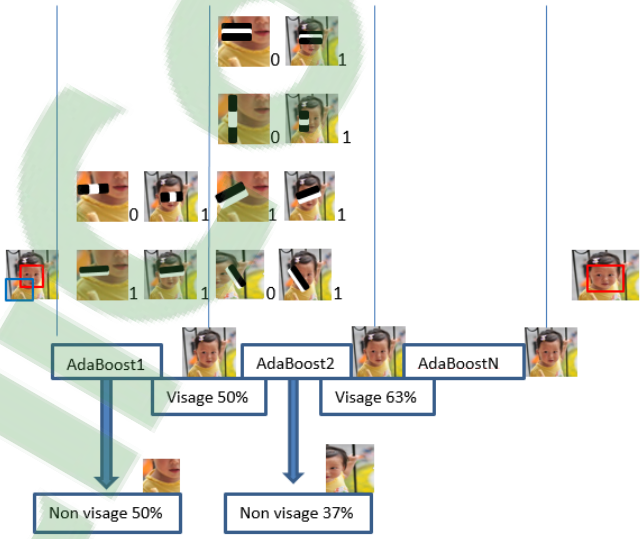

FIGURE 1.9 – Exemple d'une cascade pour la détection d'un visage

Sur l'image de la(Figure1.9), les deux descripteurs faibles votent « visage » pour la sous-image en rectangle rouge. Après la décision, le classifieur fort 1 classe la sous-image dans le rectangle rouge comme un visage et la passe au classifieur fort 2 qui va, lui aussi, avec d'autres descripteurs indépendants des descripteurs précédents, valider « visage » ou non et on itère N fois.
Par contre, la plupart des sous-images « visage » comme la sous-image dans le rectangle vert sont rejetés dans les premières étapes. Ainsi on réduit le nombre de faux positifs [26].

1.5 conclusion

En particulier, l'approche proposé par Viola et Jones [16][17][24] a été détaillée. Cette démarche est devenu l'essence de la majorité des systèmes de detection de visage :

- Présenter un algorithme de boosting et connaitre une fonction de décision. ce type d'algorithme permet également de sélectionner les descripteurs les plus pertinents parmi un ensemble de descripteurs potentiels :
- Des descripteurs rapides à calculer. Les ondeletettes de Haar sont majoritairement utilisées car elles sont très rapides à calculer ;
- Une structure en cascade. Pour accélérer le temps de classification, plusieurs classifieurs sont séquentiellement associés.

  Dans le prochain chapitre,nous abordons quelques méthodes d'apprentissage et de détection par Boosting, nous présentons les méthodes les plus intéressente en terme de la performance et cout de calcul.

CHAPITRE 2

Chapitre2 : Développement de la méthode Boosting

2.1 Introduction

La détection de visage peut être définie comme : étant donnée une image, le but est de déterminer si un ou des visages sont apparents dans l'image et s'il y en a, de localiser chacun des visages. Les techniques de la détection de visages sont développées et employées dans plusieurs domaines, surveillance, identification, biométrie, etc.
Dans l'étape de détection et de localisation des visages, nous proposons une approche par l'algorithme robuste et rapide basé sur la densité d'images, Boosting, qui combine des descripteurs simples pour un classifieur fort.
La notion de Boosting était proposée en 1995 par Freund [6]. L'algorithme de Boosting utilise les hypothèses faibles (taux d'erreur $\epsilon < 0.5$) des connaissances a priori pour construire une hypothèse forte. En 1996 Freund et Schapire ont proposé l'algorithme de Boosting qui permet de choisir automatiquement les hypothèses faibles avec des poids adaptés. L'algorithme proposé ne dépend pas de connaissances a priori [7].
En 2001, Viola et Jones ont appliqué ce même algorithme dans la détection des visages pour la première fois. Avec des descripteurs simples (Haar feature), la méthode de calcul de valeur de descripteurs (l'image intégrale), la cascade des classifieurs. Cette méthode est devenue par la suite une référence de détection de visage par ses qualités de rapidité et robustesse. La Figure2.1 nous donne un des résultats du travail de Viola et Jones [16].

FIGURE 2.1 – Un des résultats de Viola et Jones 2001

En 2002, Lienhart et al. ont étendu les descripteurs de primitive, expérimentés dans plusieurs algorithmes de Boosting : Discrete boost, Real boost, Gentle boost and Logitboost. Ces codes d'apprentissage et de détection par l'algorithme AdaBoost sont publiés dans la librairie de fonctions OpenCV (Open source Computer Vison) [18] et [19]. L'algorithme Adaboost est désormais développé et amélioré dans plusieurs publications : [17] ont utilisé cet algorithme pour toutes les poses et tous les angles de rotations de visages, aussi appelé Multi-View (Figure2.2) ; [24] ont appliqué cette méthode pour la détection de piéton, combinant les informations de mouvement et d'apparence (Figure2.3) ;[27] utilisent les descripteurs des histogrammes de gradient orienté pour la détection des humains ou des vélos.

FIGURE 2.2 – Résultats de Viola et Jones 2003

FIGURE 2.3 – Un des résultat de Viola, Jones et Snow

Adaboost est une nouvelle méthode de Boosting principalement utilisée pour stimuler les performances des algorithmes d'apprentissage. Depuis son apparition jusqu'à aujourd'hui, il y a eu la naissance de plusieurs générations. Les fondateurs de cette méthode, Schapire et Freund, ont essayé tout au long de leurs travaux d'optimiser l'algorithme d'où il y a eu l'apparition de plusieurs variantes .
En effet, il existe plusieurs dérivés de l'approche Boosting qui optimisent différemment la pondération (W_i)
Dans ce qui suit,nous allons passer en revu les différentes méthodes de Boosting ainsi que les variantes en découlant. En particulier nous allons présenter les Real boost (RAB), Gentle boost (GAB), Discrete boost (DAB), Logitboost , Modest Boost and Emphasis Boost.
La figure2.4 représente une échéance des variantes d'AdaBoost pour l'apprentissage supervisé du classifieurs binaires.

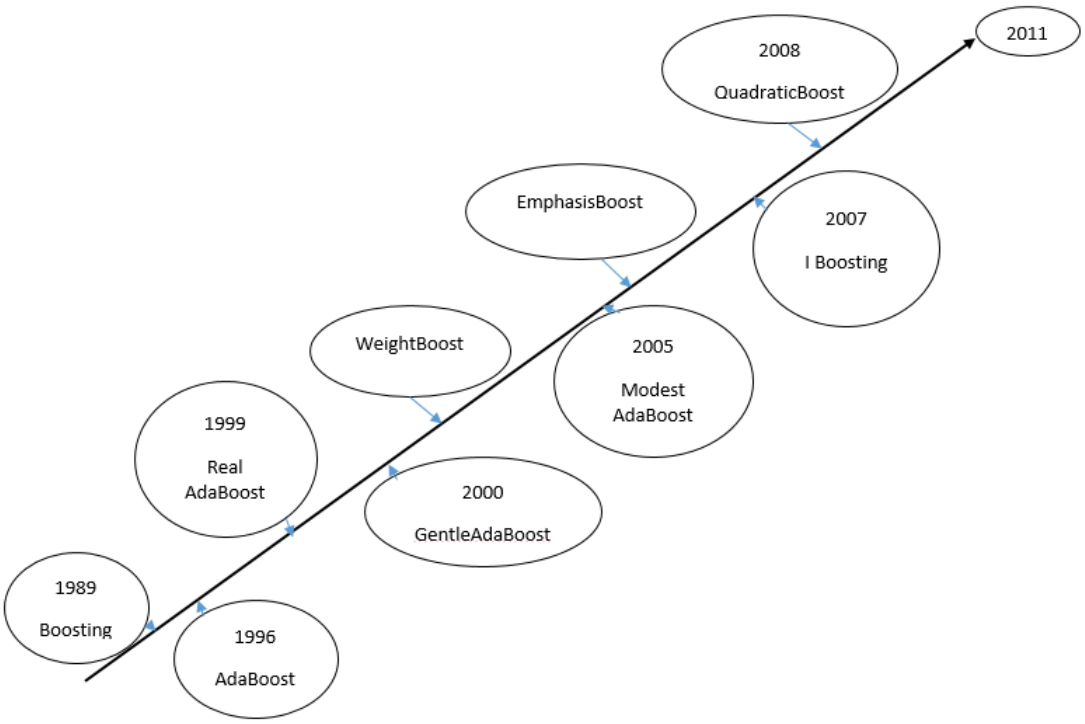

FIGURE 2.4 – A (possibly incomplete) time line of AdaBoost variants for supervised learning of binary classifiers

Ces codes d'apprentissage et de détection par l'algorithme AdaBoost sont publiés dans la librairie de fonctions OpenCV.

2.2 Position du problème

On dispose d'un ensemble de données d'apprentissage :$(x_i, y_i)_{i=1,...,N}$ où x_i est le vecteur des valeurs caractéristiques et $y_i[-1, 1]$ est la classe à laquelle appartient un individu. On définit :

$$H(x) = \sum_{1}^{T} \alpha_t h_t(x_i)$$

où $h_t(x)$ est le classifieur qui aboutit à une valeur égale à +1 ou -1, α_t est une constante et T est le nombre maximal de classifieurs. L'affectation d'une nouvelle observation se fait selon le signe de $H(x)$.
Le principe de cet algorithme est d'attribuer un poids à chaque observation de l'échantillon d'apprentissage, ce poids correspond au niveau de difficulté rencontré pour prédire la classe de cet individu.
L'algorithme commence par construire un premier classifieur sur la totalité des données d'apprentissage. Initialement, tous les poids sont identiques et sont utilisés pour construire le premier classifieur, mais aux cours des itérations, les données qui se trouvent dans une classe qui ne leur corresponde pas vont avoir des poids croissants et ceux qui sont bien placés dans leurs classes auront des poids décroissants [9].

2.3 Amélioration de l'algorithme

2.3.1 L'approche Real Boost

Real Boost a été mis en œuvre en 1999 par Freund et Schapire. Real boost est une amélioration de la méthode d'origine ou il faut ajouter une fonction qui mesure le degré de confiance en manipulant des données d'apprentissage soi-disant « faible ». La particularité de cette méthode réside dans la classe de probabilité estimée qui convertit les taux de logarithme en une valeur réel d'échelle. Cette valeur est utilisée afin d'observer les contributions des sorties du modèle en question. De plus, Real boost mesure la probabilité qu'une donnée d'apprentissage appartient à une classe donnée alors que Adaboost consiste, tout simplement, à classifier les données et calculer l'erreur pondérée. En générale, Real Boost tend à minimiser $e^{-yF(x)}$ afin de mieux trouver la classification optimale générer par le modèle utilisé.
D'autre part, la méthode Real boost accorde un taux de confiance aux classifieurs faible qui transforme l'ensemble des instances X et la prédiction booléenne en un espace réelle R. L'algorithme est comme suit :

Algorithm 3 Algorithme RealBoost

Require: $S =< (x_1, y_1), ..., (x_M, y_M) >$, avec m l'ensemble des échantillons et $(x_i, y_i \in X * -1, 1$, est l'ensemble des etiquettes.
-L'algorithme d'apprentissage **WeakLearner**.
-T est le nombre des itérations.
-Initialiser la distribution $D_t(i) = \frac{1}{m}$.
Pour $t = 1, 2, ..., T$:
1- Pour chaque instance faire :
a. Partitionner X en un bloc séparé (x_M, y_M)
b. pour chaque distribution D_t calculer :

$$W_l^j = P(x_i \in X_j, y_i = l) = \sum_{i:x,eX,y_i-1} D(i)$$

avec $l = \pm 1$.
c. Définir l'hypothèse h pour chaque X_j comme :

$$\forall x \in X, h(x) = \frac{1}{2} ln(\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon})$$

avec ε une petite constante positive.
2- Sélectionner h_t : $Z_t = min_{heH} Z$ $h_t = arg_{heH} min Z$
3- Mise à jour de la distribution $D_{t+1} = D_t(i) exp[-y_i h_t(x_i)]$
Et normalisation de D_{t+1}
4- L'hypothèse finale : $\sum H$: $H(x) = sign[\sum_{t-1}^{T} h_t(x) - b]$ avec b est le seuil(par defaut= 0)
La fonction de confiance est définie par : $Conf_x =\mid \sum_t h_t(x) - b \mid$

Néanmoins, il a y eu l'apparition des nouvelles versions de Boosting qui leurs performances dépassent souvent Real Boost [8].

Friedman et Al. ont proposé une extension de cette approche appelée Gentle Boost.

2.3.2 L'approche Gentle Boost

Gentle Boost est considérée comme autant une extension du Real Boost. Cette approche est plus performante que Real Boost étant donné sa stabilité et sa robustesse quand il s'agit des données bruitées et aux aberrants. Gentle Boost minimise la fonction exponentielle de perte d'Adaboost en utilisant les étapes de Newton.
D'autre part, Gentle Boost est une variation de l'approche Real Boost qui stimule la performance de cette dernière en appliquant une régression par la méthode des moindres carrées pondérées. En outre, Gentle Boost et Real Boost ne normalisent pas l'ensemble des apprenants pondérés de la même manière, puisque la fonction de normalisation de Real Boost est donnée par :

$$H_m(x) = P_w(y = 1 \mid x) - P_w(y = -1 \mid x)$$

alors que pour Gentle Boost, la mise à jour de la classe de probabilité pondérée est donnée par la fonction suivante :

$$h_m(x) = \frac{1}{2} log \frac{P_w(y = 1 \mid x)}{P_w(y = 1- \mid x)}$$

L'algorithme du Gentle Boost est comme suit [5] :

Algorithm 4 Algorithme GentleBoost

Require: Les poids $W_i = \frac{1}{N}, i = 1, ..., N$ et $H(x) = 0$

1. Faire pour $m = 1, 2, ..., M$.

Require: a. Estimer $h_m(x)$ en utilisant les moindres carrées pondérées de y à x.
Require: b. Mise à jour de $H(x) \leftarrow H(x) + h_m(x)$.
Require: c. Définir $W_i \leftarrow W_i exp[-y_i h_m(x_i)]$, avec $i = 1, 2, ..., N$
Require: puis normaliser $\sum_i W_i = 1$

2. L'hypothèse finale :

$$[H(x)] = sign[\sum_{m=1}^{M} h_m(x)]$$

2.3.3 L'approche LogitBoost

Les fondateurs de l'approche , Freund et Schapire en 1995, ont toujours essayé d'améliorer cet algorithme, ce qui explique l'apparition de plusieurs variantes de cette méthode qui optimisent différemment la pondération w_i. On s'intéresse ici à l'algorithme « LogitBoost » qui répond au mieux au problème de classification basé sur l'utilisation de la régle de Bayes par le calcul de la probabilité a posteriori $P(y = \frac{j}{x})$ ou **j** représente la classe à laquelle une observation **x** appartient.
Les algorithmes « Boosting » sont considérés comme des procédures d'estimation pour la conception d'un modèle de régression logistique additive

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x_i)$$

conçue pour minimiser l'espérance de l'exponentielle de la fonction de perte $J(F) = E(exp(-yH(x)))$. L'exponentielle de la fonction de perte évolue de façon exponentielle avec l'erreur de classification ce qui implique la vulnérabilité et l'hyper sensibilité de l'algorithme « AdaBoost ». Pour remédier à ce problème, [12] proposent d'utiliser une fonction de perte binomiale qui est la log-vraisemblance de la fonction de perte $L(J(H)) = E[-log(1+exp(-y(H(x))))]$ qui évolue linéairement avec l'erreur de classification et rend ainsi l'algorithme « LogitBoost » plus robuste et plus stable face aux données bruitées et aberrantes.
« LogitBoost » minimise la log-vraisemblance de la fonction de perte binomiale en utilisant les étapes de Newton [13].
L'algorithme « LogitBoost » se déroule comme suit :

Algorithm 5 Algorithme LogitBoost

On considère la probabilité pondérée $p(x) \in [0, 1]$ donné par la relation suivante :

$$p(x) = P(y) = \frac{1}{x} = \frac{exp(H(x))}{exp(H(x)) + exp(-H(x)))}$$

Etant donné :
L'ensemble des données d'apprentissage $S = (x_1, y_1), ..., (x_N, y_N)$ avec $x_i \in X et\ y_i \in Y = -1, +1$,(on a en tout **T** itérations).

Initialisation : On commence dans la première itération par fixer le poids $w_0(i) = \frac{1}{N}$ avec $i = 1, ..., N; H_0(x_i) = 0$ et la probabilité $p(x) = \frac{1}{2}$

Faire pour $t = 1, 2, ..., T$:

i. Calculer des poids $w_t(i)$ et des valeurs attendues $z_t(x_i)$ du classifieur faible pour chaque observation :

Require:

$$z_t(x_i) = \frac{y_t^*(x_i) - p(x_i)}{w_t(i)} = \frac{y_t^*(x_i) - p(x_i)}{p(x_i)(1 - p(x_i))}$$

avec

$$y_t^*(x_i) = \frac{y_t(x_i) + 1}{2}$$

et

$$w_t(i) = p(x_i)(1 - p(x_i))$$

ii. Construire la fonction $h_t(x_i)$ en utilisant les moindres carrées pondérés de $z_t(x_i)$ sur x_i en se basant sur les poids $w_t(i)$

iii. Mise à jour de $H(x) : H_{t+1}(x_i) \leftarrow H_t(x_i) + \frac{1}{2}h_t(x_i)$ et

$$p_{t+1}(x_i) \leftarrow \frac{exp(H_t(x_i)}{exp(H_t(x_i) + exp(-H_t(x_i)}$$

Fin : Les sorties de cet algorithme aboutiront au classifieur final

$$H(x) = sign[H(x)] = sign[\sum_{t=1}^{T} h_t(x_i)]$$

2.3.4 L'approche Modest Boost

Modest Boost a été inspire par Vezhnevets et Al en 2005 [23]. Cette variante d'AdaBoost vise a amélioré la généralisation des erreurs pendant la classification. Cette approche tend à donner des résultats meilleurs que ceux calculés par Gentle Boost. Dans ce but, les auteurs avaient utilisé une distribution inversée : $\overline{w} = 1 - w$. Cette distribution a permis, donc, de fournir aux données d'apprentissage correctement classifiées des poids élevés.
L'algorithme de Modest Boost est donné par le schéma suivant [23] :

Algorithm 6 Algorithme Modest Boost

Require: $S = (x_1, y_1), ..., (x_n, y_n)$
Require: Initialiser la distribution $D_0(i) = \frac{1}{N}$
Require: 1. Pour $m = 1, 2, ..., M$ et tant que $\int_m \neq 0$.
Require: a. Entrainer l'algorithme d'apprentissage $h_m(x)$ en utilisant la distribution $D_m(i)$ par la moindre carré pondérée.
Require: b. Calculer la distribution inversée
Require: $\overline{D}_m(i) = (1 - D_m(i))\overline{Z}_m$
Require: c. Calculer :
$P_m^{+1}(x) = P_{Dm}(y = +1 \cap h_m(x))$
$\overline{P}_m^{-1}(x) = P_{\overline{D}m}(y = -1 \cap h_m(x))$
$P_m^{-1}(x) = P_{Dm}(y = -1 \cap h_m(x))$
$\overline{P}_m^{-1}(x) = P_{\overline{D}m}(y = -1 \cap h_m(x))$
d. Définir :

$$h_m(x) = (P_m^{+1}(1 - \overline{P}_m^{+1}) - P_m^{-1}(1 - \overline{P}_m^{-1}))(x)$$

e. Mise à jour de la distribution :

$$D_{m+1}(i) = \frac{D_m(i)exp(-y_i h_m(x_i))}{Z_m}$$

2. L'hypothèse finale :

$$[H(x)] = sign[\sum_{i=1}^{M} h_m(x)]$$

2.3.5 L'approche Emphasis Boost

La variante Emphasis Boost utilise une fonction d'Emphasis pondérée (WE). Chaque modèle d'entrée est pondéré selon un critère (paramétré par λ), à travers la fonction WE, de telle sorte que le processus de formation se concentre sur les motifs "critiques" (près de la limite de classification) ou sur l'erreur quadratique de chaque motif.
L'algorithme suivant présente les détails de **Emphasis Boost** [12] :
LA fonction WE est définit par :

$$w_i = exp(\lambda(\sum_{j=1}^{m}(\alpha_j H_j(x_i) - y_i)^2) - (1 - \lambda)(\sum_{j=1}^{m} H_j(x_i))^2)$$

Et contrôler où l'accent est placé. Cette formulation flexible permet de choisir combien on peut considérer les termes de proximité au moyen d'un paramètre de pondération ($0 \leq \lambda \leq 1$). De cette façon, nous avons une impulsion par pondération limite et erronée d'une technique d'échantillons. Concernant la valeur de λ, trois cas particuliers sont intéressants :

- $\lambda = 0$ Mettez l'accent sur les modèles «critiques» car seule la «proximité» de la frontière est prise en compte

$$w_i = exp[-(\sum_{j=1}^{m} H_j(x_i)^2]$$

- $\lambda = 0.5$, Nous obtenons la fonction d'Emphasis classique de Real Boost

$$w_i = exp[(\frac{\sum_{j=1}^{m}(H_j(x_i) - y_i)^2}{2}) - \frac{(\sum_{j=1}^{m}(H_j(x_i))^2}{2}]$$

- $\lambda = 1$, La fonction d'Emphasis fait attention à l'erreur quadratique de chaque modèle.

$$w_i = exp[\sum_{j=1}^{m}(H_j(x_i) - y_i)^2]$$

Algorithm 7 Algorithme Emphasis Boost

Input : Dataset $Z = z_1, z_2, ..., z_N$, with $z_t = (x_1, y_i)$, where $x_i \in X$ and $y_i \in -1, +1$.
M, the maximum number of classifiers.
λ, weighting parameter ($0 \leq \lambda \leq 1$).
Output : $H(x)$, a classifier suited for the training set.

Require: 1. Initialize the weights $w_i = \frac{1}{N}, i \in 1, ..., N$.
Require: 2. **for** $m = 1$ to M and while $H_m \neq 0$ **do**
Require: 3. Fit a classifier $H_m(x)$ to the training data using weights w_i.
Require: 4. Let $err_m = \sum_{i=1}^{N} \frac{w_i y_i H_m(x_i)}{w_i}$
Require: 5. Compute $\alpha_m = 0.5log(\frac{1 + err_m}{1 - err_m})$
Require: 6. Set $w_i = exp(\lambda(\sum_{j=1}^{m}(\alpha_j H_j(x_i) - y_i)^2) - (1 - \lambda)(\sum_{j=1}^{m} H_j(x_i))^2)$
Require: 7. Renormalize to $\sum_i w_i = 1$.
Require: 8. **end for**
9. Final classifier :

$$H(x) = sign(\sum_{j=1}^{M} \alpha_j H_j(x))$$

.

2.4 La vitesse de convergence

En plus du problème de sur-apprentissage rencontré par le boosting dans les bases de données modernes déjà évoqué précédemment, il existe un autre problème qui est celui de la vitesse de convergence des algorithmes de boosting (spécialement Adaboost). En effet, en cas de présence de données fortement bruitées, l'erreur optimale de l'algorithme d'apprentissage utilisé est atteinte tardivement. En d'autres termes, Adaboost "perd" du temps, et donc des itérations à pondérer ces exemples qui ne méritent aucune attention,

puisqu'il s'agit de bruit. Des recherches ont été menées pour détecter les données bruitées et améliorer ainsi les performances du boosting en termes de convergence tel que iBoost [2], qui vise à spécialiser les hypothèses faibles sur les exemples qu'elles sont supposées correctement classer. iAdaboost est aussi une approche qui contribue à améliorer Adaboost contre sa convergence. En fait, l'idée de base de l'amélioration est la modification du théorème de [2]. Cette modification est réalisée afin d'intégrer le risque de Bayes et de mettre en exergue les situations où certains exemples de classes différentes partagent la même représentation. Les effets de cette modification sont une convergence plus rapide vers le risque optimal et une réduction du nombre d'hypothèses faibles à construire.
Enfin, RegionBoost [2] est une nouvelle stratégie de pondération des classifieurs. Cette pondération est évaluée au moment du vote par une technique basée sur les k plus proches voisins de l'exemple à étiqueter. Cette approche permet de spécialiser chaque classifieur sur des régions de l'ensemble d'apprentissage [2].

2.4.1 iBoost

Cet algorithme est conçu afin de spécialiser les classifieurs faibles sur les exemples sur lesquels leurs prédictions sont performantes. Achaque itération, iBoost utilise des variables indicatrices pour pondérer l'exemple en se basant sur les hypothèses faibles $h(t)$. Ces variables peuvent etre considérées comme des coefficients présentants l'adéquation de l'exemple avec $h(t)$. Les résultats montrent la performance de iBoost surtout de point de vue de succès [2] .

2.4.2 iAdaBoost

Cette approche est conçue non seulement pour améliorer AdaBoost face au risque de sur-apprentissage mais également pour améliorer sa convergence. en fait, l'idée de base de l'amélioration de la convergence de iAdaBoost est la modification du théorème de [2], qui prouve que la borne sur l'erreur d'apprentissage issu du classifieur final H est atteinte par minimisation du produit des Z_t, obtenus pour chacune des T hypothèses faibles. Cette modification est réalisée afin d'intégrer le risque de Bayes et mettre en exergue les situations où certains exemples de classes différentes partagent la meme représentation. Les effets de cette modification sont une convergence plus rapide vers le risque optimal et une réduction du nombre d'hypothèses faibles à construire.
L'ensemble de ces idées ont donc été reprises dans un nouvel algorithme, baptisé, iAdaBoost. L'algorithme est le suivant :

Algorithm 8 Algorithme iAdaBoost

Data : A learning samples LS, a number of iterations T, a weak learner WL
Result : An aggregated classifier H
Build the k nearest neighbor graphs on LS ;
Initialise distribution : $\forall x \in LS, D_1(x) = \frac{1}{\mid LS \mid}$;
for $t = 2$ to T **do**
$h_t = WL(LS, D_t)$;

$$\varepsilon_t = \sum_{e:y(x) \neq h_t(x)} D_t(x);$$

$\alpha_t = \frac{1}{2} log(\frac{1-\varepsilon_t}{\varepsilon_t})$;
Distribution Update : $\forall x \in LS$;
Require: $D_{t+1}(x) = D_t(x) e^{-\alpha_t y(x) h_t(x)}$;
Require: $D'_{t+1}(x) = \frac{4 I_t(x) D_{t+1}(x)}{Z'_t}$;
where $I_t(x) = \mid \gamma_t(x) \mid (1 - \gamma_t(x) \mid)$;
and $\gamma_t(x) = \frac{\mid n^+(x) - n^-(x) \mid}{(n^+(x) - n^-(x))}$;
/* Z'_t is a Normalization Factor*/
Return H s.t $H(x) = \frac{1}{T}(\sum_{t=1}^{T} \alpha_t h_t(x))$;

ce pseudo-code ressemble fortement à celui d'AdaBoost, puisque la principale modification relève de l'information recueillie au sein du graphe de voisinage, et de son utilisation dans le calcul du coefficient de confiance de $D_{t+1}(x)$ [22] .

2.4.3 Region Boost

Region Boost utilise une nouvelle stratégie de pondération pour chaque classifieur. Cette pondération se base sur les k plus proche voisins de l'exemple à étiqueter lors du vote. Cette approche permet de spécialiser chaque classifieur sur des régions de l'ensemble d'apprentissage d'où son nom. Malgré cette amélioration, les résultats de Region Boost une certaine sensibilité au bruit [14] .
The pseudo-code of RegionBoost is the following :

Algorithm 9 Algorithme RegionBoost

Initialize an empty subset S_0
$E_0 = 100$;
$n = 0$;
while $n\langle k$
To search the classifier C_j so that the equal error rate obtained combining the classifiers that belong to S_{n-1} and C_j is minimum.
S_n is constructed by adding to the subset S_{n-1} the classifier C_j.
E_n = equal error rate obtained combing by sum rule the classifiers that belong to S_n.
$n = n + 1$.
While $n > 2$
To test each combination of $(n-1)$ classifiers that belong to S_n, we denote B_e the combination that obtains the lowest equal error rate and w_e denote its equal error rate with E_{Be}.
if $E_{Be} < E_{n-1}$.
$E_{n-1} = B_e$;
$n = n - 1$;
end
end

2.5 Conclusion

Dans ce chapitre, nous avons revu les algorithmes des méthodes de Boosting afin de mettre en avant la meilleure méthode quel soit possible et adapter à la détection de visage.Les deux approches de AdaBoost et Modest Boost apparaissent comme le meilleur approche. Cependant il est toujours difficile d'évaluer la qualité des résultats fournit par les deux approches. Pour cela nous avons mis en conccurence les deux approches pour évaluer les résultats.
En chapitre 3 nous allons présenter les résultats essentiel obtenues ainsi que la démarche adapté.

CHAPITRE 3

Chapitre3 : Application

3.1 Introduction

Après la spécification détaillée des principaux besoins, nous passons à l'étape de conception qui permet de bien mener la phase d'implémentation. Nous allons, donc, choisir les outils de développement qui sont les plus adéquats avec nos différents besoins, ce qui va nous permettre d'atteindre au mieux et le plus rapidement nos objectifs. Dans ce qui suit, nous donnerons une vue d'ensemble du système.

3.2 Vue d'ensemble du système

On n'utilise une caméra statique placée à une hauteur suffisante telle que. Les têtes des piétons sont visibles. Pour détecter les têtes, on utilise la technique Viola et Jones. Le processus de formation est en ligne et crée un classifieur utilisant AdaBoost. La vidéo acquise est entrée au module de détection, qui détecte les têtes dans la première image et crée des trajectoires initiales. Le module de suivi utilise un cadre probabiliste composé d'un modèle de mouvement et d'un modèle d'apparence. Le modèle probabiliste est basé sur un filtre à particules qui est utilisé pour suivre les têtes dans la trame suivante. Le modèle de mouvement est utilisé pour prédire la position des têtes dans la trame suivante. Le modèle d'apparence donne une probabilité basée sur un histogramme de couleur, qui sert à mettre à jour l'état de l'objet.

1. Acquérir la vidéo de foule à partir d'une caméra
2. Dans la première image, détecter et compter le nombre des visages détectées.
3. Initialisez les trajectoires avec les positions initiales.
4. Initialiser les poids de la trajectoire
5. Initialisez le modèle d'apparence (histogramme couleur) de chaque région de visage
6. Recherche de nouveau visage dans la scène

Dans la figure ci-dessous (figure3.1 nous présentons une vue globale de notre système de détection de visage :

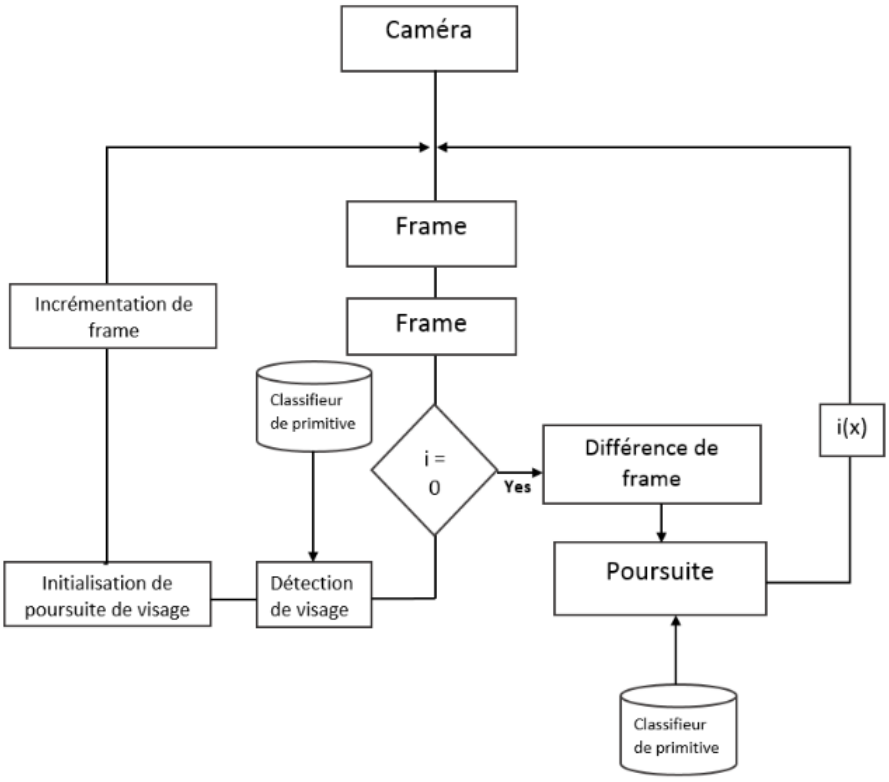

FIGURE 3.1 – Schéma d'apprentissage

3.2.1 Conception globale

Parmi les outils utilisés on cite :

QT 5.8

QT est une suite de logiciels de développement pour Windows et Linux conçu par Nokia. QT 5.8 est un ensemble complet d'outils de développement permettant de générer des applications Windows/linux ou encore pour, des Services Web XML, des applications bureautiques et des applications mobiles.

3.2.2 OpenCV

OpenCV est une bibliothèque graphique libre, développée à la base par Intel, spécialisée dans le traitement d'image temps réel.

3.3 Les Etapes d'apprentissage

- Collect des images d'apprentissage positives et negatives
- Noter les images positives à l 'aide de *objectmarker* or *ImageClipper* tools
- Création d'un fichier vecteur *.vec* basé sur les images notées positives à l'aide *./createsamples*
- Entrainement du classifieur using *./haartraining*
- Exécuter le classifieur en utilisant la fonction $cvHaarDetectObjects()$

Etape 1 : Collect des images positives et images négatives
Nous avons utilisé des images de visage libres pour l'ensemble des images positives et des images de nature et autre que visage pour créer l'ensemble des images négatives.
Il n'y a pas de règle précise pour construire l'ensemble des images négatives et les images positives. Cependant il est conseillé d'avoir un nombre très élevé des images positives et négatives afin de rendre notre classifieur par Boosting beaucoup plus précis.

Etape 2 : Organisation des images négatives
Les ensembles des images négatives sont enregistrés dans un sous répertoire $.../training/negative$ et la liste des images est créé par le fichier de commande $createlist.sh$
$ls - a * .jpg > bg.txt$
Le fichier $bg.txt$ contient une liste des noms des images négatives :
image0010.jpg
image0011.jpg
image0012.jpg
...
Cette liste des images est utlisée par la suite pour l'entrainement du classifieur par Boosting. Dans la figure ci-dessous, nous présentons un exemple des images négatives

FIGURE 3.2 – Exemple des image négatives

Etape 3 : Crop et annotation des images positives
Dans cette étape, nous devons créer un fichier de données (vecteur). Ce fichier contient les noms des images positives et la position des visages dans ces mêmes images.
Le fichier vecteur peut créer à l'aide de l'utilitaires : *Objectmarker*.
Les images positives sont sauvegardées dans le répertoire *../training/positive/rawdata* positive images.
L'utilitaire *objectmaker* est sauvegardé dans le répertoire *../training/positive* avec un fichier contenant les annotations des images positives.

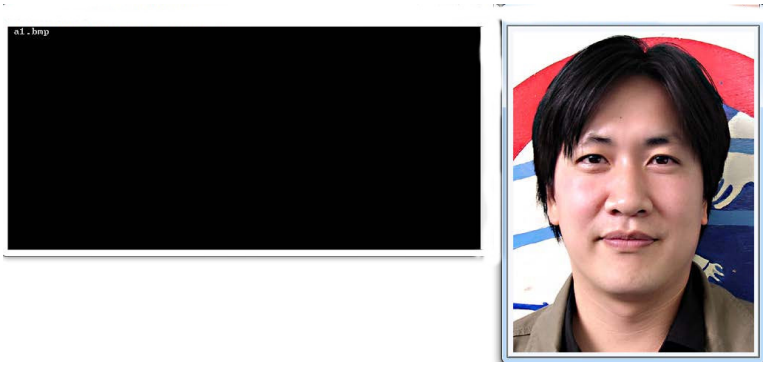

FIGURE 3.3 – Annotation d'une image positive

L'annotation du visage sur une image positive s'éffectue par l'execution de l'utilitaire *./objectmaker*. Dans la figure ci-dessous nous proposons un exemple avec deux fenêtres sur les terminaux :
1-Une fenêtre avec le nom de l'image.
2-L'image positive.
Le visage est séléctionné sur l'image positive à l'aide de la souris cette dernière est positionnée sur le coin supérieur et puis un deuxième point est séléctionné au bas du visage. Cette procédure est réalisée pour toutes les images positives. A la fin nous aurons un fichier texte $info.txt$ avec le contenu suivant : $rawdata/image0010.bmp134127424$
$rawdata/image0011.bmp3352570394095809212040453 6$
$rawdata/image0012.bmp21024909045689982$
Le premier chiffre donne une information sur le nombre de visage existant sur chaque image ainsi que leurs positions. Pour l'image $image0010.bmp$, le nombre 1 veut dire qu'il y a 01 visage situé entre $x = 34$, $y = 12$, $width = 74$, and $height = 24$.

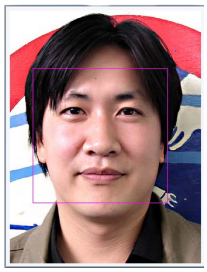

FIGURE 3.4 – Crop d'un visage sur une image positive

Etape 4 : Création du vecteur des images positives
Dans le répertoire *../training/* nous utilisons le fichier des commmandes *samplescreation.sh* pour créer le fichier vecteur : $createsamples - infopositive/info.txt - vecvector/facevector.vec - num200 - w24 - h24$
Main Parameters :

- info *positive/info.txt* chemin pour le fichier info pour les images positives
- vec *vector/facevector.vec* chemin pour le fichier vecteur résultant
- num 200 le nombre des images positives concatinées dans le fichier vecteur
- w 24 la largeur des visages
- h 24 Hauteur des visages

Le fichier de commande charge le fichier $info.txt$ et compresse les visages dans le fichier vecteur nommé $facevector.vec$ dans le repertoire *../training/vector*.

Etape 5 : Apprentissage des primitives de Haar
Dans le répertoire *../training* , le fichier de commande ou le script *haartraining.sh* donne l'exécution de l'apprentissage :
$haartraining - datacascades - vecvector/vector.vec - bgnegative/bg.txt - npos200 - nneg200 - nstages15 - mem1024 - modeALL - w24 - h24 - nonsym$

- Data cascades Chemin pour enregistrer le classifieur en cascade
- vec *data/vector.vec* Le répertoire pour avoir le fichier vecteur
- bg *negative/bg.txt* Répertoire vers le fichier fonds
- npos 200 Nombre des images positives $\leq$ no. en format *bmp*
- nneg 200 Nombre des images négatives (patches) $\geq$ npos
- nstages 15 Le nombre de fois que l'apprentissage est executé
- mem 1024 La taille mémoire assignée en MB
- mode ALL pour fixer la fenêtre de largeur $w24$ et $h24$ Sample size
- nonsym Uniquement lorsque les visages ne sont pas bien allignés.

Les dimensions $-W$ and $-H$ utilisées dans *harrtraining.sh* sont identiques à ceux utilisées pour la création des images de visages $sample - creation.sh$ *./Harrtraining* collecte des nouveaux ensembles d'images négatives à chaque étape de l'apprentissage, l'utilisation de l'option $-nneg$ limite la taille de ses ensembles d'images.
Lorsque le script *./haartraning.sh* est excuté nous obtenons quelques informations qui seront affichées sur la figure ci-dessous :
noeud père : Defines the current stage under training process
N : Nombre des étapes et les primitives utilisées
F :visage symetrique ou ps
ST.THR : seuil de l'étape
HR : taux dépendant ST.THR
FA : Fausse Alarme
EXP. ERR : Erreur de classifieur fort

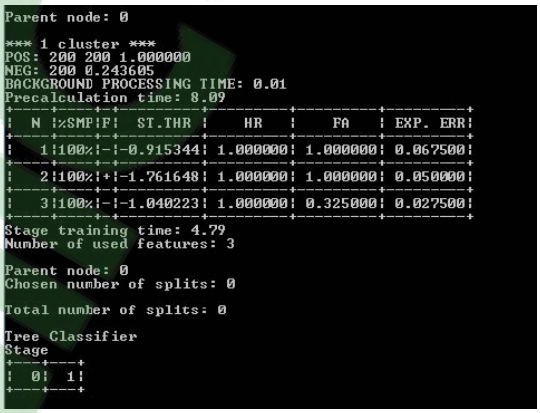

FIGURE 3.5 – Classification correcte des images positives

Dans la figure ci-dessous un autre exemple d'apprentissage réalisé en 10 étape pour le classifieur :

- Le nombre de primitive est important
- Fausse détection augmente
- Le temps de calcul devient important

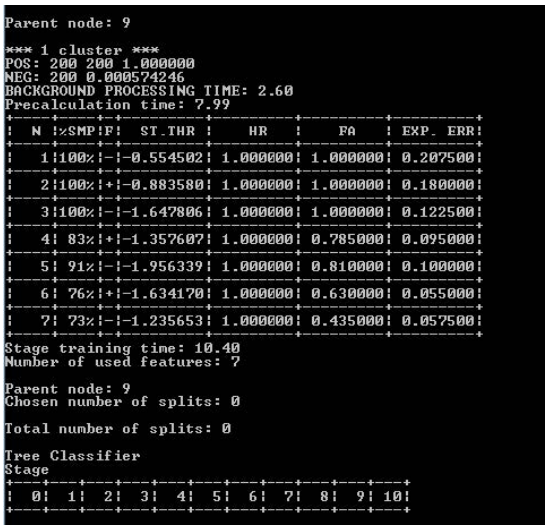

FIGURE 3.6 – Classification avec les erreurs des images positives

Etape 6 : Creating the XML File
A la fin de l'apprentissage, dans le repértoire *../training/cascades/* un catalogue noté de 0 jusqu'à "N-1", avec N le nombre des étapes définies dans *./haartraining.sh*.Dans ces catalogues nous devons avoir un fichier texte :$AdaBoostCARTHaarClassifier.txt$ file. Tous les répertoires de 0..N-1 sont déplacés dans le répertoire *../cascade2xml/data/*.Tous les étapes créées (classifieurs faibles) dans un seul classifieur fort enregistré dans un fichier XML file. Un script est exécuté pour combiner les classifieurs faible en un seul classifieur fort par :*convert.sh* at *../cascade2xml/* et par la suite le script : $./haarconvdatamyfacedetector.xml24x24$. Le fichier $myfacedetecor.xml$ est le fichier de sortie et 24x24 reprénsente les dimensions de visage sur une image (W et H).

3.4 Détection de visage

Après avoir obtenu un classifieur fort à partir de l'étape d'entrainement, nous utilisons ce classifieur pour classer les régions comme visage ou non à partir du flux vidéo. Nous avons utilisé une version modifiée de la fonction $cvHaarDetectObjects()$. Cette nouvelle fonction analyse les images d'entrées plusieurs fois à différentes échelles et cherche des régions rectangulaires dans l'image classée comme positive par la primitive de Haar. Par la suite ces régions sont renvoyées comme une séquence de rectangles.
La mise en échelle peut être controler par augmenter ou réduire la taille des visages. Cependant, il est important de noter que ce paramètre échelle ne peut augmenter indéfiniment.

3.5 Suivi du visage

Nous avons proposé une simple stratégie de suivi des visage à l'aide d'un algorithme représenté sur la figure ci-dessous :

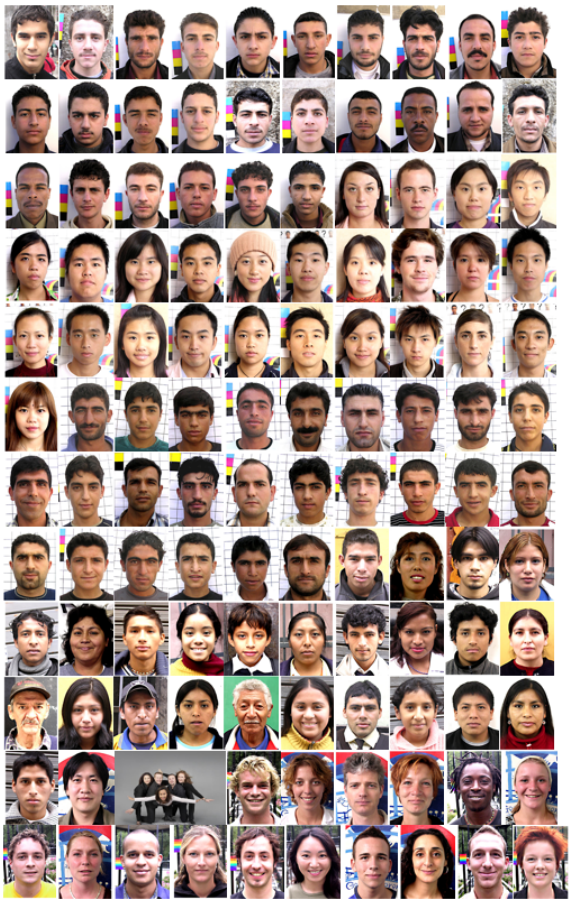

FIGURE 3.7 – Algorithme de poursuite de visage dans une foule

3.5.1 Suivi des filtres à particules

Nous avons utlisé le filtre à particules qui est bien connu dans le domaine de suivi des objets. L'incertitude sur la position d'un visage est représentée comme un ensemble de particules pondérées, chaque particule représentant un état possible. Le filtre propage les particules particulières du temps. Les pondérations de notre filtre sont estimées à partir d'un modèle de mouvement probabiliste, puis rééchantillonne les particules selon leur poids.
La première distribution du filtre est basée sur l'emplacement du visage la première fois qu'il est détecté.Voici les differentes étapes :

1. Pronostiquez : prédire la distribution sur la position du visage à un instant donné.
2. Mesurer : pour chaque particule propagée k, mesurons le poids du filtre à l'aide d'un modèle d'apparence basé sur l'histogramme de couleur normalisé.
3. Rééchantillonnage : on rééchantillonne les particules pour éviter les poids dégénérés

Sans rééchantillonnage, au fil du temps, la particule de poids le plus élevé tendrait à peser un poids et l'autre tendraient à zéro. Le rééchantillonnage supprime beaucoup de particules à faible poids et se multiplie.

3.5.2 Extraction de caractéristiques de la couleur

Espace de couleur HSV

L'espace HSV (Hue, Saturation, Value) ou TSV (Teinte Saturation Valeur) est un espace colorimétrique, défini en fonction de ses trois composantes :

• Teinte (H) : le type de couleur. La valeur varie entre 0 et 360.
• Saturation (S) : l'intensité de la couleur. La valeur varie entre $0et100\%$. Plus la saturation d'une couleur est faible, plus l'image sera "grisée" et plus elle apparaîtra fade, il est courant de définir la "désaturation" comme l'inverse de la saturation.
• Valeur (V) : la brillance de la couleur, elle varie entre $0et100\%$.

Dans OpenCV, la valeur H est normalisée en 0–180 ; les valeurs S et V sont normalisées en 0–255 (la Figure3.8).

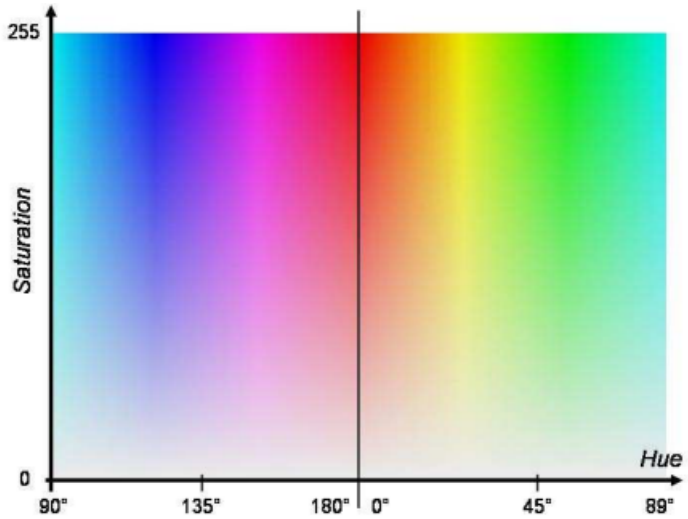

FIGURE 3.8 – Espace de couleur HSV

Le modèle HSV est une transformation non-linéaire de l'espace de couleur RGB. Il est défini dans OpenCV par la transformation ci-dessous :

$$\begin{cases} V = max(R,G,B) \\ S = (V - min(R,G,B)) * 255/V if\ V! = 0, 0otherwise\ (G-B) * 60/S, if\ V = R \\ H = 180 + (B-R) * 60/S, if\ V = G240 + (R-G) * 60/S, if\ V = BifH < 0then\ H = H + 360 \end{cases}$$

Histogramme HS

L'espace de couleur HSV sépare les informations de couleurs en teinte, saturation et valeur. Ces informations peuvent être utilisées pour la reconnaissance de visage [21].
En considérant que la valeur de la " brillance" de la couleur est influencée par la condition extérieure, nous n'extrayons que les valeurs de H et S. Un histogramme 2D de la distribution de H-S est calculé pour chaque visage comme les caractéristiques couleurs. L'histogramme H-S est défini en 30 bins à l'axe H et 32 bins à l'axe S. Il peut être considéré comme un vecteur de 960 composants pour l'apprentissage avec SVM. La Figure3.9 donne un exemple de l'histogramme H-S d'un visage segmenté.

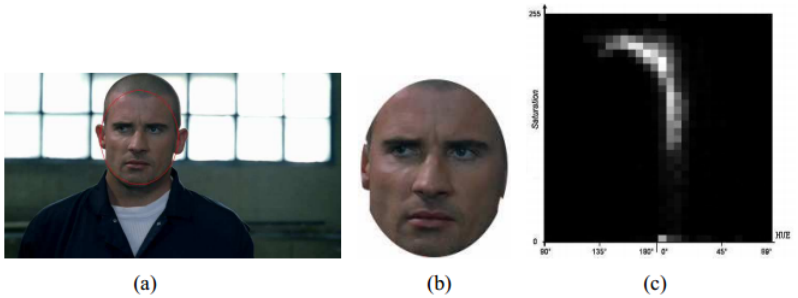

FIGURE 3.9 – Histogramme H-S (a) une frame (b) visage segmenté (c) Histogramme H-S

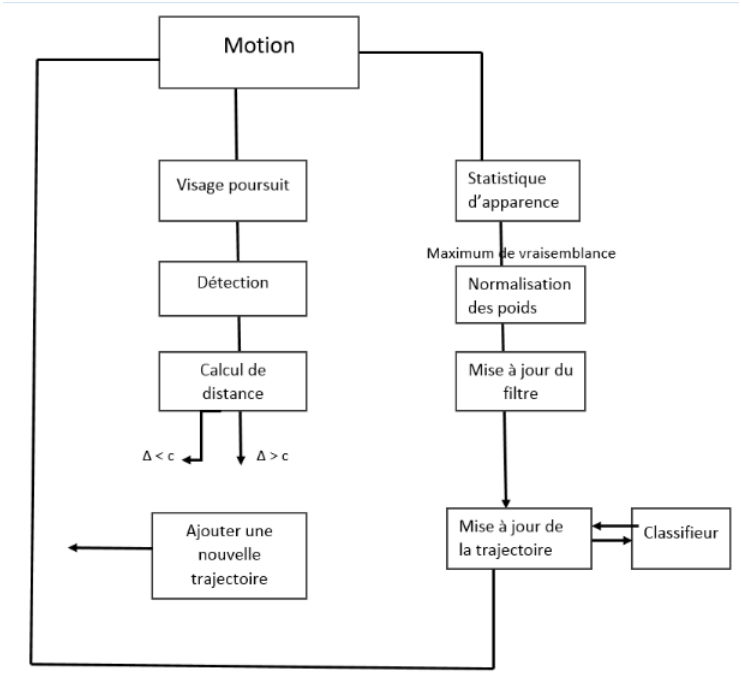

FIGURE 3.10 – Schéma du système de détection de visage dans une foule

3.6 Conclusion

Dans notre travail d'extraction, nous avons extrait des caractéristiques de la couleur de l'espace HS . Une évaluation sur ces caractéristiques nous montre qu'une seule caractéristique ne peut pas être capable de distinguer les visages des acteurs. La solution est de combiner plusieurs caractéristiques.
Nous avons exposé les résultats obtenus lors du test de notre système implémenté de classification des expressions faciales.
Ces résultats étaient bons, mais nous avons besoin d'une plus large base de données pour parvenir à réaliser un système puissant et précis .

Conclusion Générale

La detection de visage dans la foule demeure un problème complexe et non parfaitement résolu, malgré tous les travaux réalisés au cours des dernières années. Plusieurs problèmes incomblent à cette tâche de détection de visage et chacun d'eux est non trivial. De nombreuses conditions réelles affectent la performance d'un système.
A travers ce projet nous avons mis en oeuvre une approche de détection de visage, et pour aboutir à ce but, il fallait au préalable aborder le problème d'apprentissage par algorithme de boosting .
Pour cela nous avons détaillé dans le chapitre 3 nos travaux de détection et notre approche pour améliorer le résultat obtenu en ajoutant un module de prétraitement. Bien que notre méthode d'amélioration ait montré de bons résultats, au niveau de l'interpolation de visages non détectés par la librairie OpenCV, elle élimine parfois de vrais visages.
Après la phase de détection, nous avons pu aborder la tâche de reconnaissance. Notre rapport dans cette tâche délicate, est d'utiliser la notion des points d'intérêt pour reconstruire un modèle de visage.

Ce projet ne manque pas de perspectives : pour la tâche de détection, et à partir des visages détectés par la librairie OpenCV, il est intéressant de trouver d'autres méthodes d'élimination des fausses alarmes et de détecter en contre-partie les visages oubliés par la méthode de boosting. Nous proposons d'utiliser des approches heuristiques, pour prévoir si une telle détection correspond à un visage ou non, en tenant compte des positions des autres visages.

Bibliographie

[1] Ognjen Arandjelovic and Andrew Zisserman. Automatic face recognition for film character retrieval in feature-length films. In *Computer Vision and Pattern Recognition, CVPR. IEEE Computer Society Conference on*, volume 1, pages 860–867. IEEE, 2005.

[2] Emna Bahri and Mondher Maddouri. Une nouvelle approche du boosting face aux données bruitées. In *et gestion des connaissances : EGC*, 2008.

[3] Charles Bouveyron. *Modélisation et classification des données de grande dimension : application à l'analyse d'images.* PhD thesis, Université Joseph-Fourier-Grenoble I, 2006.

[4] Elena Casiraghi, Raffaella Lanzarotti, and Giuseppe Lipori. A face detection system based on color and support vector machines. In *Italian Workshop on Neural Nets*, pages 113–120. Springer, 2003.

[5] Mark Culp, Kjell Johnson, and George Michailidis. ada : An r package for stochastic boosting. *Journal of Statistical Software*, 17(2) :9, 2006.

[6] Y. Freund. *Boosting a weak learning algorithm by majority, Information and Computation.* 1995.

[7] Y Freund. Iriimms wit ha new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[8] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780) :1612, 1999.

[9] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[10] S. Z. Li G. Guo and C. Kapluk. Face recognition by support vector machines. In *Image and Vision Computing*, pages 631–638. Springer, 2001.

[11] Robert Godin, Guy Mineau, Rokia Missaoui, and Hafedh Mili. Méthodes de classification conceptuelle basées sur les treillis de galois et applications. *Revue d'intelligence artificielle*, 9(2) :105–137, 1995.

[12] Vanessa Gómez-Verdejo, Manuel Ortega-Moral, Jerónimo Arenas-García, and Aníbal R Figueiras-Vidal. Boosting by weighting critical and erroneous samples. *Neurocomputing*, 69(7) :679–685, 2006.

[13] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Joint Pattern Recognition Symposium*, pages 297–304. Springer, 2003.

[14] Loris Nanni and Alessandra Lumini. Regionboost learning for 2d+ 3d based face recognition. *Pattern Recognition Letters*, 28(15) :2063–2070, 2007.

[15] Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines : an application to face detection. In *Computer vision and pattern recognition. Proceedings. IEEE computer society conference on*, pages 130–136. IEEE, 1997.

[16] M. Jones P. Viola. Rapid object detection using a boosted cascade of simple features. In *Conference On Computer Vision And Pattern Recognition*, 2001.

[17] M. Jones P. Viola. Fast multi-view face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[18] J. Maydt R. Lienhart. An extended set of haar-like features for rapid object detection. In *IEEE ICIP*, volume 1.

[19] V. Pisarevsky R. Lienhart, A. Kuranov. Empirical analysis of detection cascades of boosted classifiers for rapidobject detection. Technical report, May 2002.

[20] François Savard. Réseaux de neurones à relaxation entraînés par critère d'autoencodeur débruitant. 2012.

[21] David Saxe and Richard Foulds. Toward robust skin identification in video images. In *Automatic Face and Gesture Recognition, Proceedings of the Second International Conference on*, pages 379–384. IEEE, 1996.

[22] Marc Sebban and Henri-Maxime Suchier. Etude sur l'amélioration du boosting : Réduction de l'erreur et accélération de la convergence. *Journal électronique d'intelligence artificielle (submitted, 2003)*, 2004.

[23] Alexander Vezhnevets and Vladimir Vezhnevets. Modest adaboost-teaching adaboost to generalize better. In *Graphicon*, volume 12, pages 987–997, 2005.

[24] Paul A Viola and Michael J Jones. Detecting pedestrians using patterns of motion and appearance in videos, 2007. US Patent.

[25] LECUN YANN. *Modèles connexionnistes de l'apprentissage*. PhD thesis, These de Doctorat, Universite Paris 6, 1987.

[26] Shuji ZHAO. Apprentissage et recherche par le contenu visuel de catégories sémantiques d'objets vidéo. Master's thesis, Juillet 2007.

[27] Yeh M-C Cheng K-W Zhu Q., Avidan S. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1491–1498, 2006.