الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة ابي بكر بلقايد –تلمسان-

Université Aboubakr belkaid –Tlemcen-

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunication

Spécialité :Réseau et **S**ystème de **T**élécommunication

Par :MAHI Abdelhakim

Sujet

Détection de visage par l'algorithme de boosting

Soutenue publiquement le 28 Juin 2018 devant le jury :

M. S.M MERIAH	professeur à l'université de Tlemcen	Président
M. M.BOUSAHLA	MCB à l'université de Tlemcen	Examinateur
M. F. DERRAZ	MCB à l'université de Tlemcen	Encadreur
M. A.H BOUACHA	MCA à l'université de Tlemcen	Encadreur

Année Universaire 2017/2018

Dédicace

Je dédie ce mémoire
A mes chers parents ma mère et mon père
Pour leur patience , leur soutien et leurs encouragements .
A mon frère et mes sœurs .
A mes amis et mes camarades .
Sans oublié tout les professeurs que ce soit du primaire , du moyen , du secondaire ou de l'enseignement supérieur .

Remerciements

Je tiens à remercier toutes les personnes qui ont contribuées de près ou de loin à la réalisation de ce mémoire.

Tout d'abord, je veux adresser mes remerciements à mon directeur de mémoire F. DERRAZ Maitre de conférences à l'université Abou-Bekr Belkaid de Tlemcen, pour sa grande disponibilité et ses précieux conseils tout au long de la rédaction de ce mémoire.

Je remercie également A. BOUACHA Maitre de conférences à l'université Abou-Bekr Belkaid de Tlemcen qui a bien voulu participer à mon encadrement .

Je remercie également S.M MERIAH Professeur à l'université Abou-Bekr Belkaid de Tlemcen. Je suis heureux pour ta présence dans mon jury .

A M. BOUSAHLA Maitres de conférences à l'université Abou-Bekr Belkaid c'est un grand honneur de vous voir siéger dans mon jury.

Je tiens enfin à remercier tout particulièrement ma famille qui m'a accordé la liberté d'action et la patience nécessaires pour réaliser ce travail ainsi que toutes les personnes qui m'ont soutenue.

Résumé

L'objectif de ce mémoire est de présenter un système de détection de visage d'une, ou plusieurs, personnes se déplaçant dans une foule.
Pour y parvenir, j'ai spréparé un schéma d'algorithme de détection de visage à l'aide des algorithmes de Boosting. j'ai comparé les algorithmes les plus pertinents dans la littérature et j'ai choisi l'algorithme de Boosting adaptive pour ses qualités de classification et son forte discrimination des visages. Un apprentissage de l'algorithme de Boosting est nécessaire pour construire un classifieur fort à partir d'un ensemble de classifieur faible.
Par la suite j'ai associé l'algorithme de Boosting a un filtre de poursuite, j'ai choisi le filtre particulaire, afin de poursuivre chaque visage de son apparition jusqu'à sa disparition. Le filtre particulaire est l'une des meilleures solutions pour tracer les trajectoires et prédire la position des visages dans une scène. L'algorithme proposé pour cette tache, comprends quatre composants : l'apprentissage, la détection, la poursuite et la prédiction.

Mots clés : Viola Jones, AdaBoost, image intégrale, classifieurs en cascade, détection de visage, OpenCV.

Abstract

The aim of this thesis is to present a face detection system for one or several people moving in a crowd.
To achieve this, I have prepared a facial detection algorithm scheme using boosting algorithms. I have compared the most pertinent algorithms in the literature and I have chosen the adaptive boosting algorithm for its classification qualities and its strong discrimination of the faces. Knowing the boosting algorithm is necessary to build a strong classifier from a weak classifier set. Subsequently, I combined the boosting algorithm with a tracking filter, I have choose the particle filter, to track each face from its appearance until it disappears. The particulate filter is one of the best solutions for drawing the trajectories and predicting the position of the faces in a scene. The algorithm proposed for this task includes four components : learning, detection, tracking and prediction.

Keywords : Viola Jones, AdaBoost, integral image, cascade classifiers, Face Detection, OpenCV.

خلاصة

الغرض من هذه المذكرة هو تقديم نظام للكشف عن الوجه لشخص او أكثر ينتقل بين حشد من الناس . لتحقيق ذلك , اعددت خطة خوارزمية للكشف عن الوجه باستخدام خوارزميات بوستينغ . لقد قارنت اكثر الخوارزميات ذات الصلة في الادبيات و اخترت خوارزمية التعزيز التكيفية لخصائص تصنيفها و التمييز القوي للوجه . ان تعلم خوارزمية التعزيز ضروري لبناء مصنف قوي من مجموعة من التصنيفات الضعيفة .
بعد ذلك , ربطت خوارزمية بوستينغ مع فلتر تتبع , واخترت مرشح الجسيمات , لمتباعة كل وجه من مظهره حتى اختفاءه مرشح الجسيمات هو واحد من افضل الحلول لتتبع المسارات واكتشاف مكان الوجوه في المشهد . تتضمن الخوارزمية لهذه المهمة أربعة مكونات: التعلم و الكشف و التتبع و الاكتشاف .

الكلمات الرئيسية : فيولا و جونز ,ادا بوست , الصورة الكاملة , المصنفات المتتالية , كشف الوجه , اوبن سيفي .

Table des matières

.

Table des figures

Introduction Générale:

La reconnaissance du visage est un domaine très actif dans la Vision par Ordinateur et dans la Biométrie. Elle a été étudiée vigoureusement il y a déjà 25 ans et sa finalité est de produire des applications de sécurité, des applications en robotique, d'interface homme-machine, applications pour les appareils photo numériques, jeux et divertissement.

La reconnaissance du visage implique généralement deux étapes:

· ***La détection de visage*** qui consiste à chercher et à détecter un ou plusieurs visages en parcourant une image numérique ou une vidéo.

· ***La reconnaissance de visage*** qui consiste à comparer le visage détecté à celui se trouvant dans la base de données de visages reconnus afin de savoir quel est ce visage reconnu ou à qui appartient-il.

De façon plus indirecte, la détection de visage est la première étape vers des applications plus évoluées, qui nécessitent la localisation du visage.

La détection de visage est un domaine de la vision par ordinateur consistant à détecter un visage humain dans une image numérique ou unvidéo. C'est un cas spécifique de détection d'objet, où l'on cherche à détecter la présence et la localisation précise d'un ou plusieurs visages dans une image. C'est l'un des domaines de la vision par ordinateur parmi les plus étudiés avec de très nombreuses publications et de conférences spécialisées. La forte activité de recherche en détection de visage a également permis de faire émerger des méthodes génériques de détection d'objet.

La détection de visage a de très nombreuses applications directes en vidéo-surveillance, biométrie , robotique, commande d'interface homme- machine photographie, indexation d'images et de vidéos, recherche d'images par le contenu, etc. Elle permet également de faciliter l'automatisation complète d'autres processus comme la reconnaissance de visage ou la reconnaissance d'expressions faciales.

Parmi les applications directes, la plus connue est sa présence dans de nombreux appareils photo numérique, où elle sert à effectuer la mise au point automatique sur les visages. C'est également une technique importante pour les interfaces homme-machine évoluées, afin de permettre une interaction plus naturelle entre un humain et un ordinateur.

La détection de visage est aussi utilisée en indexation d'images et recherche d'information, où elle peut être utilisée pour rechercher des images contenant des personnes, associer automatiquement un visage à un nom dans une page web, identifier les principales personnes dans une vidéo par clustering.

La détection de visage peut aussi servir à déterminer l'attention d'un utilisateur, par exemple face à un écran dans l'espace public, qui peut également, une fois le visage détecté, déterminer le sexe et l'âge de la personne afin de proposer des publicités ciblées. Cela peut également servir à savoir si une personne est bien présente devant une télévision allumée, et dans le cas contraire mettre l'appareil en veille ou réduire la luminosité afin d'économiser de l'énergie.

La détection de visage est un sujet difficile, notamment dû à la grande variabilité d'apparence des visages dans des conditions sans contraintes:

· Variabilité intrinsèque des visages humains (couleur, taille, forme)

· Présence ou absence de caractéristiques particulières(cheveux,moustache, barbe, lunettes..)

· Expressions faciales modifiant la géométrie du visage

· Occultation par d'autres objets ou d'autres visages

· Orientation et pose (de face, de profil)

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.

Dans mon travail, je propose une approche précise, robuste et rapide de détection de visage dans les plans-séquences pour ça j'ai choisi l'algorithme de Boosting adaptive qui devenu une référence pour la détection de visage , Par la suite j'ai associé l'algorithme de Boosting a un filtre de poursuite, j'ai choisi le filtre particulaire, afin de poursuivre chaque visage de son apparition jusqu'à sa disparition .

Figure 1.1: Détection des visages dans une image

Chapitre 1

Méthodes de détection du visage

1. Introducion :

La détection de visage dans une image est un traitement essentiel avant la phase de reconnaissance. En effet le processus de reconnaissance de visages ne pourra jamais devenir intégralement automatique s'il n'a pas été précédé par une étape de détection efficace. Le traitemet consiste à rechercher dans une image la position des visages et de les extraire sous la forme d'un ensemble d'imagettes dans le but de faciliter leur traitement ultérieur. Un visage est considéré correctement détecté si la taille d'imagette extraite ne dépasse pas 20% de la taille réelle de la région faciale et qu'elle contient essentiellement les yeux,le nez et la bouche [1, 2]

2. Méthodes de détection :

Une classification des méthodes de localisation faciale a été proposée par Yang et al[2]. Les méthodes sont divisés en quatre catégories :

- Approche basée sur la reconnaissance
- Approche basée sur les caractéristiques invariants
- Approche basée sur l' appariement de gabarits
- Approche basée sur l'apparence

a. Approche basée sur la reconnaissance :

Ces méthodes se basent sur la connaissance des différents éléments qui constituent un visage et des relations qui existent entre eux. Ainsi, les positions relatives de différents éléments clés tels que la bouche, le nez et les yeux sont mesurées pour servir ensuite à la classification 'visage' , 'nonvisage' par Chiang et al. [3]. Le problème dans ce type de méthode est qu'il est difficile de bien définir de manière unique un visage.

b. Approche basée sur les caractéristiques invariants :

Ces approches utilisent les éléments invariants aux variations d'illumination, d'orientation ou d'expression tels que la texture ou la signature de couleur de la peau pour la détection

c. Approche basée sur l'appariement de gabarits :

Des modèles caractéristiques d'un visage entier ou de sous-partie de visage (bouche, oeil, nez) sont crées. La localisation se fait ensuite sur base de la corrélation de ces modèles avec les candidats [4].

d. Approche basée sur l'apparence :

Ces méthodes utilisent le même principe que présenté au point précédent mais se basent sur des modèles appris à partir d'un ensemble d'essai. Ces méthodes pré- sentent l'avantage de s'executer trés rapidement mais demandent un long temps d'entraînement. Les méthodes appartenant à cette catégorie ont montré de bons résultats par rapport aux trois autres types de méthodes [5]. On peut citer parmi celles-ci, la méthode basée sur les réseaux de neurones de Rowley et al. [6], la méthode de Schneiderman et Kanade [7] basée sur un classifieur de Bayes naîf ainsi que le fameux algorithme de Viola et Jones [8] fonctionnant en temps réel, et ce dernier sera détaillé ci-dessous.

3 . La méthode de Viola et Jones :

Une avancée majeure dans le domaine a été réalisée par les chercheurs Paul Viola et Michael Jones en 2001 [8]. Ces derniers ont proposé une méthode basée sur l'apparence.

La méthode de Viola et Jones est une méthode de détection d'objet dans une image numérique, proposée par les chercheurs Paul Viola et Michael Jones en 2001. Elle fait partie des toutes premières méthodes capables de détecter efficacement et en temps réel des objets dans une image. Inventée à l'origine pour détecter des visages, elle peut également être utilisée pour détecter d'autres types d'objets comme des voitures ou des avions. La méthode de Viola et Jones est l'une des méthodes les plus connues et les plus utilisées en particulier pour la détection de visages et la détection de personnes.

En tant que procédé d' apprentissage supervisé, la méthode de Viola et Jones nécessite de quelques centaines à plusieurs milliers d'exemples de l'objet que l'on souhaite détecter pour entraîner un classifieur. Une fois son apprentissage réalisé ce classifieur est utilisé pour détecter la présence éventuelle de l'objet dans une image en parcourant celle-ci de manière exhaustive, à toutes les positions et dans toutes les tailles possibles.

Cette méthode bénéficie d'une implémentation sous licence BSD dans Open CV qui est comme nous l'avons vu, une librairie très utilisée en vision par ordinateur .

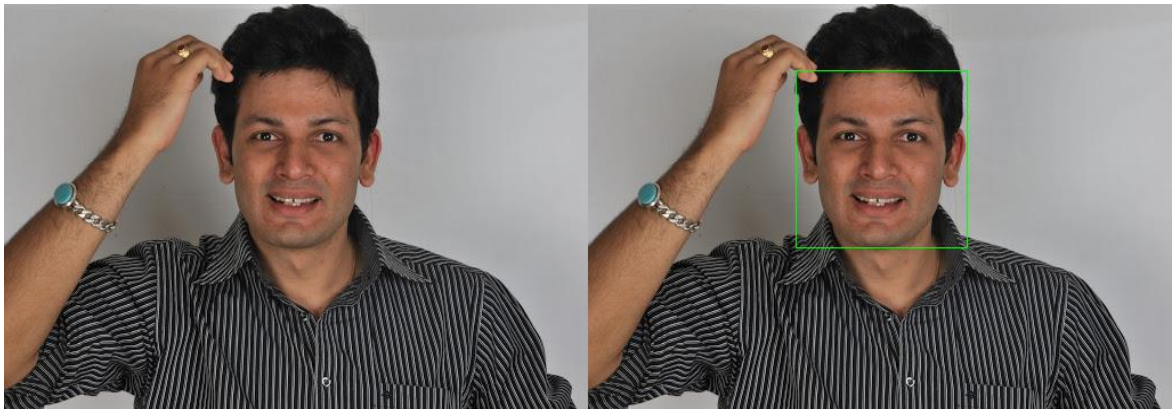

Figure 1.2: exemple de détection du visage avec la méthode de Viola et Jones

3.1 Éléments de la méthode :

La méthode de Viola et Jones est une approche basée apparence [9], qui consiste à parcourir l'ensemble de l'image en calculant un certain nombre de caractéristiques dans des zones rectangulaires qui se chevauchent. Elle a la particularité d'utiliser des caractéristiques très simples mais très nombreuses. Une première innovation de la méthode est l'introduction des images intégrales, qui permettent le calcul rapide de ces caractéristiques. Une deuxième innovation importante est la sélection de ces caractéristiques par boosting, en interprétant les caractéristiques comme des classifieurs. Enfin, la méthode propose une architecture pour combiner les classifieurs boostés en un processus en cascade, ce qui apporte un net gain en temps de détection.

La méthode, en tant que méthode d' apprentissage supervisé, est divisée en deux étapes : une étape d'apprentissage du classifieur basé sur un grand nombre d'exemples positifs (c'est-à-dire les objets d'intérêt, par exemple des visages) et d'exemples négatifs, et une phase de détection par application de ce classifieur à des images inconnues.

a) Caractéristiques :

Plutôt que de travailler directement sur les valeurs de pixels, et pour être à la fois plus efficace et plus rapide, Viola et Jones proposent d'utiliser des caractéristiques, c'est à dire une représentation synthétique et informative, calculée à partir des valeurs des pixels. Viola et Jones définissent des caractéristiques très simples, les caractéristiques pseudo-Haar[10], qui sont calculées par la différence des sommes de pixels de deux ou plusieurs zones rectangulaires adjacentes. La figure ci-contre donne des exemples des caractéristiques proposées par Viola et Jones à 2, 3 ou 4 rectangles, dans lesquelles la somme de pixels sombres est soustraite de la somme des pixels blancs. Leur nom vient de leur similitude avec lesondelettes de Haar, précédemment proposées comme caractéristiques par Papageorgiou [11] et dont se sont inspirés Viola et Jones[10].

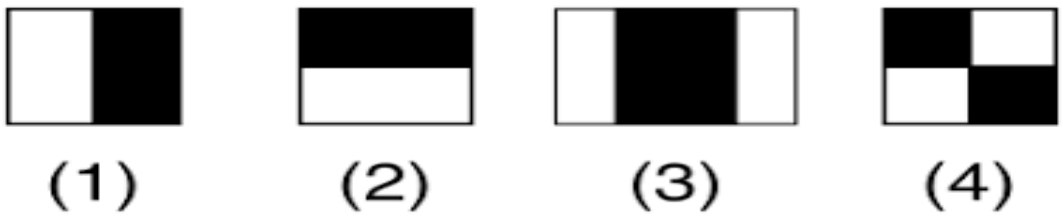

Figure 1.3 : Descripteurs de primitive de Haar dans une fenêtre : 2-,3-,4-rectangles détecteurs.

Pour calculer rapidement et efficacement ces caractéristiques sur une image, les auteurs proposent également une nouvelle méthode, qu'ils appellent « image intégrale ». C'est une représentation sous la forme d'une image, de même taille que l'image d'origine, qui en chacun de ses points contient la somme des pixels situés au-dessus de lui et à sa gauche. Plus formellement, l'image intégrale *ii* au point (*x*,*y*) est définie à partir de l'image *i* par :[10]

$$ii\,(\mathrm{x},\mathrm{y}) = \sum_{x' \le x, y' \ge y} i(x', y') \qquad \textbf{(1.1)}$$

A cause de cette représentation, une caractéristique formée de deux zones rectangulaires peut être calculée en seulement 6 accès à l'image intégrale, et donc en un temps constant quelle que soit la taille de la caractéristique.

b) Calcul :

Les caractéristiques sont calculées à toutes les positions et à toutes les échelles dans une fenêtre de détection de petite taille, typiquement de 24 × 24 pixels [10] ou de 20 × 15 pixels [12]. Un très grand nombre de caractéristiques par fenêtre est ainsi généré, Viola et Jones donnant l'exemple d'une fenêtre de taille 24 × 24 qui génère environ 160 000 caractéristiques.

En phase de détection, l'ensemble de l'image est parcouru en déplaçant la fenêtre de détection d'un certain pas dans le sens horizontal et vertical (ce pas valant 1 pixel dans l'algorithme original). Les changements d'échelles se font en modifiant successivement la taille de la fenêtre de détection . Viola et Jones utilisent un facteur multiplicatif de 1,25, jusqu'à ce que la fenêtre couvre la totalité de l'image.

Finalement, et afin d'être plus robuste aux variations d'illumination, les fenêtres sont normalisées par la variance.

La conséquence de ces choix techniques, notamment le recours aux images intégrales, est un gain notable en efficacité, les caractéristiques étant évaluées très rapidement quelle que soit la taille de la fenêtre.

c) Sélection de caractéristiques par boosting :

Le deuxième élément clé de la méthode de Viola et Jones est l'utilisation d'une méthode de boosting afin de sélectionner les meilleures caractéristiques. Le boosting est un principe qui consiste à construire un classifieur « fort » à partir d'une combinaison pondérée de classifieurs « faibles », c'est-à-dire donnant en moyenne une réponse meilleure qu'un tirage aléatoire. Viola et Jones adaptent ce principe en assimilant une caractéristique à un classifieur faible, en construisant un classifieur faible qui n'utilise qu'une seule caractéristique. L'apprentissage du classifieur faible consiste alors à trouver la valeur seuil de la caractéristique qui permet de mieux séparer les exemples positifs des exemples négatifs. Le classifieur se réduit alors à un couple (caractéristique, seuil).

L'algorithme de boosting utilisé est en pratique une version modifiée d' AdaBoost, qui est utilisée à la fois pour la sélection et pour l'apprentissage d'un classifieur « fort ». Les classifieurs faibles utilisés sont souvent des arbres de décision. Un cas remarquable, fréquemment rencontré, est celui de l'arbre de profondeur 1, qui réduit l'opération de classification à un simple seuillage.

L'algorithme est de type itératif, à nombre d'itérations déterminé. À chaque itération l'algorithme sélectionne une caractéristique qui sera ajoutée à la liste des caractéristiques sélectionnées aux itérations précédentes et le tout va contribuer à la construction du classifieur fort final. Cette sélection se fait en entraînant un classifieur faible pour toutes les caractéristiques et en élisant celle de ces dernières qui génère l'erreur la plus faible sur tout l'ensemble d'apprentissage. L'algorithme tient également à jour une distribution de probabilité sur l'ensemble d'apprentissage, réévaluée à chaque itération en fonction des

résultats de classification. En particulier, plus de poids est attribué aux exemples difficiles à classer, c'est à dire ceux dont l'erreur est élevée. Le classifieur « fort » final construit par AdaBoost est composé de la somme pondérée des classifieurs sélectionnés.

Plus formellement, on considère un ensemble de n images $(x_1, \ldots, x_n)$ et leurs étiquettes associées $(y_1, \ldots, y_n)$, qui sont telles que $y_i = 0$ si l'image x_i est un exemple négatif et $y_i = 1$ si x_i est un exemple de l'objet à détecter. L'algorithme de boosting est constitué d'un nombre T d'itérations, et pour chaque itération t et chaque caractéristique j, on construit un classifieur faible h_j , le but est d'obtenir un classifieur h qui prédise exactement les étiquettes pour chaque échantillon, c'est-à-dire $y_i = h(x_i)$ $\forall i \in \{1 \ldots n\}$. En pratique, le classifieur n'est pas parfait et l'erreur engendrée par ce classifieur est donnée par :

$$\epsilon_j = \sum_{i=1}^{n} w_i \, |h_j(x_i) - y_i| \qquad \textbf{(1.2)}$$

les w_i étant les poids associés à chaque exemple et mis à jour à chaque itération en fonction de l'erreur obtenue à l'itération précédente. On sélectionne alors à l'itération t le classifieur h_t présentant l'erreur la plus faible : $\epsilon_t = \min_j(\epsilon_j)$.

Le classifieur fort final $h(x)$ est construit par seuillage de la somme pondérée des classifieurs faibles sélectionnés :

$$h(x) = \begin{cases} 1 & \text{si } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{sinon} \end{cases} \qquad \textbf{(1.3)}$$

Les α_t sont des coefficients calculés à partir de l'erreur ϵ_t.

d) Cascade de classifieurs :

La méthode de Viola et Jones est basée sur une approche par recherche exhaustive sur l'ensemble de l'image, qui teste la présence de l'objet dans une fenêtre à toutes les positions et à plusieurs échelles. Cette approche est cependant extrêmement coûteuse en calcul. L'une des idées-clés de la méthode pour réduire ce coût réside dans l'organisation de l'algorithme de détection en une cascade de classifieurs. Appliqués séquentiellement, ces classifieurs prennent une décision d'acceptation << la fenêtre contient l'objet et l'exemple est alors passé au classifieur suivant >>, ou de rejet<< la fenêtre ne contient pas l'objet et dans ce cas l'exemple est définitivement écarté >>. L'idée est que l'immense majorité des fenêtres testées étant négatives (c.-à-d. ne contiennent pas l'objet), il est avantageux de pouvoir les rejeter avec le moins possible de calculs. Ici, les classifieurs les plus simples, donc les plus rapides, sont situés au début de la cascade, et rejettent très rapidement la grande majorité des exemples négatifs [10] .Cette structure en cascade peut également s'interpréter comme un arbre de décision dégénéré, puisque chaque noeud ne comporte qu'une seule branche.

En pratique, la cascade est constituée d'une succession d'étages, chacune étant formée d'un classifieur fort appris par AdaBoost. L'apprentissage du classifieur de l'étage *n* est réalisé avec les exemples qui ont passé l'étage *n* - 1 ; ce classifieur doit donc faire face à un problème plus difficile : plus on monte dans les étages, plus les classifieurs sont complexes [10] .

Le choix du nombre *K* d'étages est fixé par l'utilisateur ; dans leur méthode originale, Viola et Jones utilisent *K* = 32 étages. L'utilisateur doit également spécifier le taux de détection minimal d_i et le taux de fausse alarme maximal f_i à atteindre pour l'étage *i*. Le taux de détection de la cascade est alors donné par :

$$D = \prod_{i=1}^{K} d_i \qquad \textbf{(1.4)}$$

et le taux de fausse alarme par :

$$F = \prod_{i=1}^{K} f_i \qquad \textbf{(1.5)}$$

En pratique, les taux d_i et f_i sont les mêmes pour tous les étages. Indirectement, ces taux déterminent également le nombre de caractéristiques utilisées par les classifieurs forts à chaque étage : les itérations d'Adaboost continuent jusqu'à ce que le taux de fausse alarme cible soit atteint. Des caractéristiques/classifieurs faibles sont ajoutés jusqu'à ce que les taux cibles soient atteints, avant de passer ensuite à l'étage suivant.

Pour atteindre des taux de détection et de fausse alarme corrects en fin de cascade, il est nécessaire que les classifieurs forts de chaque étage aient un bon taux de détection ; ils peuvent par contre avoir un taux de fausses alarmes élevé. Si l'on prend l'exemple d'une cascade de 32 étages, pour obtenir une performance finale $D = 0.9$ et $F = 10^{-6}$, chaque classifieur fort doit atteindre $d_i = 0{,}997$. Chaque étage ajouté diminue donc non seulement le nombre de fausses alarmes, mais aussi le taux de détection.

Plusieurs chercheurs font remarquer que cette idée de filtrer rapidement les exemples négatifs les plus simples n'est pas nouvelle [13 ,14] . Elle existe dans d'autres méthodes sous forme d' heuristiques, comme la détection de la couleur chair [15,16] ou une étape de pré-classification [16] .

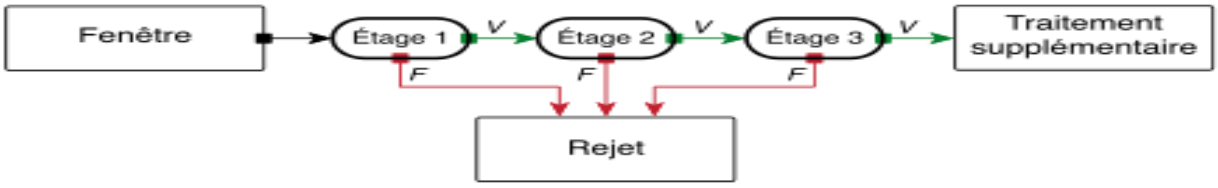

Figure 1.4 : Illustration de l'architecture de la cascade : les fenêtres sont traitées séquentiellement par les classifieurs, et rejetées immédiatement si la réponse est négative (F).

3.2 Étapes clés de la méthode :

a) Apprentissage :

L'apprentissage est réalisé sur un très large ensemble d'images positives (c'est-à- dire contenant l'objet) et négatives (ne contenant pas l'objet). Plusieurs milliers d'exemples sont en général nécessaires. Cet apprentissage comprend :

1. Le calcul des caractéristiques pseudo-Haar sur les exemples positifs et négatifs ;

2. L'entraînement de la cascade : à chaque étage de la cascade, un classifieur fort est entraîné par AdaBoost. Il est construit par ajouts successifs de classifieurs faibles entraînés sur une seule caractéristique, jusqu'à l'obtention de performances conformes aux taux de détection et de fausse alarme souhaités pour l'étage.

b) Détection :

La détection s'applique sur une image de test dans laquelle on souhaite déceler la présence et la localisation d'un objet.

En voici les étapes :

- parcours de l'ensemble de l'image à toutes les positions et échelles, avec une fenêtre de taille 24 × 24 pixels, et application de la cascade à chaque sous-fenêtre, en commençant par le premier étage :
  - calcul des caractéristiques pseudo-Haar utilisées par le classifieur de l'étage courant,
  - puis calcul de la réponse du classifieur,
  - passage ensuite à l'étage supérieur si la réponse est positive, à la sous-fenêtre suivante sinon,
  - et enfin l'exemple est déclaré positif si tous les étages répondent positivement ;
- fusion des détections multiples : l'objet peut en effet générer plusieurs détections, à différentes positions et échelles ; cette dernière étape fusionne les détections qui se chevauchent pour ne retourner qu'un seul résultat.

3.3 Limitations et extensions de la méthode :

De très nombreuses améliorations ont été proposées par la suite, visant à améliorer le paramétrage de la méthode, ou à en combler un certain nombre de limitations.

L'une des premières améliorations est apportée par Lienhart et Maydt en 2002[17]. Ils proposent d'étendre l'ensemble de caractéristiques pseudo-Haar utilisé de 4 à 14 caractéristiques. De même, ils introduisent des caractéristiques « de biais » (tournées de 45°), ainsi qu'une méthode pour les calculer basée sur une extension des images intégrales[17].

D'autres types de caractéristiques ont également été utilisés en remplacement des caractéristiques de Haar : les histogrammes de gradients orientés[18], les motifs binaires locaux ou la covariance de région[19]. Les chercheurs ont également proposé d'utiliser des variantes de l'algorithme de boosting, notamment RealBoost, qui produit un indice de confiance à valeurs réelles, en plus de la classification. Plusieurs travaux ont ainsi montré la supériorité de RealBoost sur AdaBoost dans le cadre de l'algorithme de Viola et Jones.

Viola et Jones étendent en 2003 leur système à la détection de piétons dans des vidéos, en incluant une information de mouvement en plus de l'information d'apparence [12] .

Une des limitations de la méthode est son manque de robustesse à la rotation, et sa difficulté à apprendre plusieurs vues d'un même objet. En particulier, il est difficile d'obtenir un classifieur capable de détecter à la fois des visages de face et de profil. Viola et Jones ont proposé une amélioration qui permet de corriger ce défaut [20], qui consiste à apprendre une cascade dédiée à chaque orientation ou vue, et à utiliser lors de la détection un arbre de décision pour sélectionner la bonne cascade à appliquer. Plusieurs autres améliorations ont été proposées par la suite pour apporter une solution à ce problème.

Une autre limitation importante de la méthode de Viola et Jones concerne le temps d'apprentissage de la cascade, compris généralement entre plusieurs jours et plusieurs semaines de calcul, ce qui limite sévèrement les possibilités de tests et de choix des paramètres[21].

Un des problèmes majeurs de la méthode proposée par Viola et Jones est qu'il n'existe pas de méthode optimale pour choisir les différents paramètres régissant l'algorithme : le nombre d'étages, leur ordre ou les taux de détection et de fausses alarmes pour chaque étage doivent être choisis par essais et erreurs. Plusieurs méthodes sont proposées pour déterminer certains de ces seuils de manière automatique.

Un reproche également fait à la méthode concerne la perte d'information subie au passage d'un étage à l'autre de la cascade, perte due à l'effet couperet des décisions d'acceptation ou de rejet prises à chaque étage. Certains chercheurs proposent la solution de garder l'information contenue dans la somme pondérée des classifieurs faibles, par exemple le « boosting chain » de Xiao[22] . Une modification plus radicale de structure est proposée par Bourdev et sa notion de cascade souple, qui consiste essentiellement à supprimer le concept d'étages, en formant un seul classifieur fort, donc une seule somme, et en permettant de prendre une décision à chaque évaluation de classifieur faible et de s'affranchir de la contrainte des taux de détection et de fausses alarmes cibles[14].

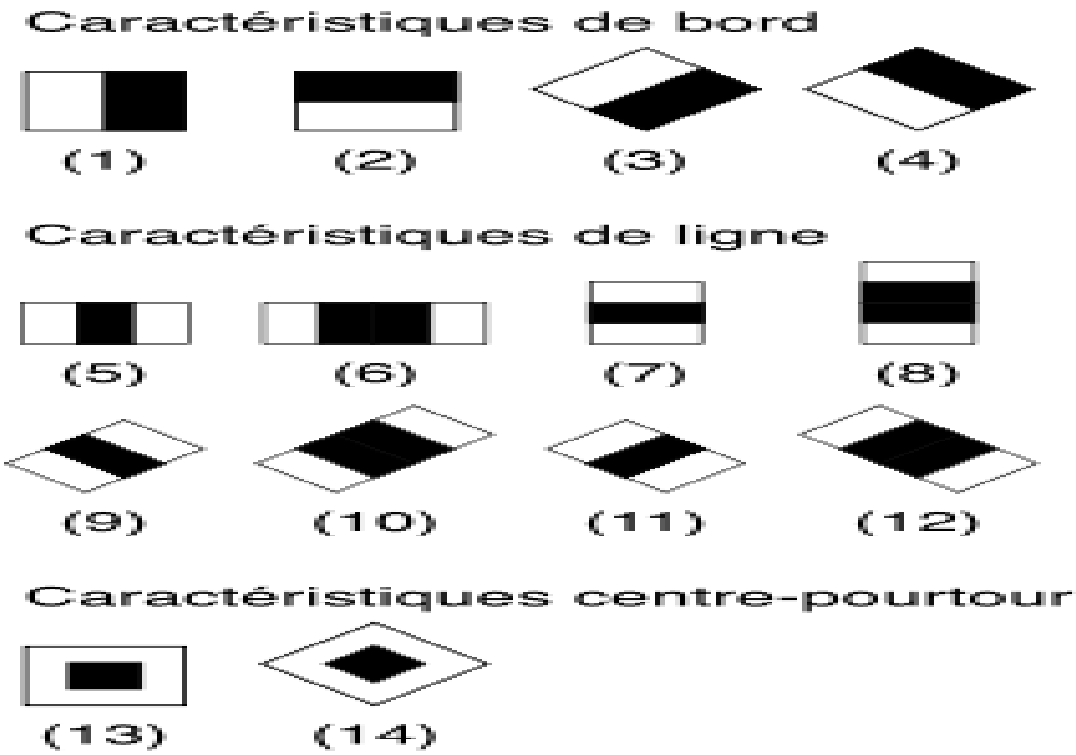

Figure 1.5 : L'extension des* caractéristiques pseudo-Haar *proposée par Lienhart.

4 . Conclusion :

Dans ce chapitre j'ai éxpliqué les différents méthodes de détection de visage en générale ensuite j'ai détaillé la méthode de viola et jones.Ces derniers ont proposé une méthode basée sur l'apparence. cette méthode est capable de détecter efficacement et en temps réel des objets dans une image. La méthode de Viola et Jones est l'une des méthodes les plus connues et les plus utilisées en particulier pour la détection de visages et la détection de personnes.

Chapitre 2

Les algorithmes de boosting

1.Introduction :

La détection de visage peut ètre définie comme : étant donné 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 , je propose une approche par l'algorithme robuste et rapide basé sur la densité d'images, boosting, qui combine des descripteurs simples pour un classifieur fort.

Le boosting est un domaine de l'apprentissage automatique (branche de l'intelligence artificielle). C'est un principe qui regroupe de nombreux algorithmes qui s'appuient sur des ensembles de classifieurs binaires .La notion de boosting était proposée en 1995 par Freund[23].En 2001 viola et jones ont appliqué cette algorithme dans la détection de visages pour la première fois.

Dans ce qui suit , je vais passer en revu les différents méthodes de boosting .

2. principe :

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 :

$$\mathrm{H(x)}= \left(\sum_1^T \alpha_t H_t(x_i)\right) \qquad \textbf{(2.1)}$$

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 [24].

3. Les algorithmes de boosting :

3.1 Algorithme AdaBoost :

Après leur travail distinct sur les algorithmes de boosting, Freund et Schapire a proposé l'algorithme adaptatif (AdaBoost) [25], [26], [27]. L'idée clé derrière AdaBoost est d'utiliser des versions pondérées des mêmes données d'entraînement au lieu de sous-échantillons aléatoires de ceux-ci. Le même kit d'apprentissage est utilisé à plusieurs reprises et, pour cela la raison, il n'a pas besoin d'être très grand, contrairement aux méthodes de renforcement antérieures.

L'algorithme AdaBoost est maintenant une méthode bien connue et profondément étudiée pour construire un ensemble de classifieurs avec de très bonnes performances [28]. L'algorithme apprend un ensemble de classificateurs, en utilisant un Wi, afin de produire le classificateur final de la forme :

$$H(x)= \text{sign}\ \left(\sum_{i=1}^{M} \alpha_i H_i(x)\right) \qquad \textbf{(2.2)}$$

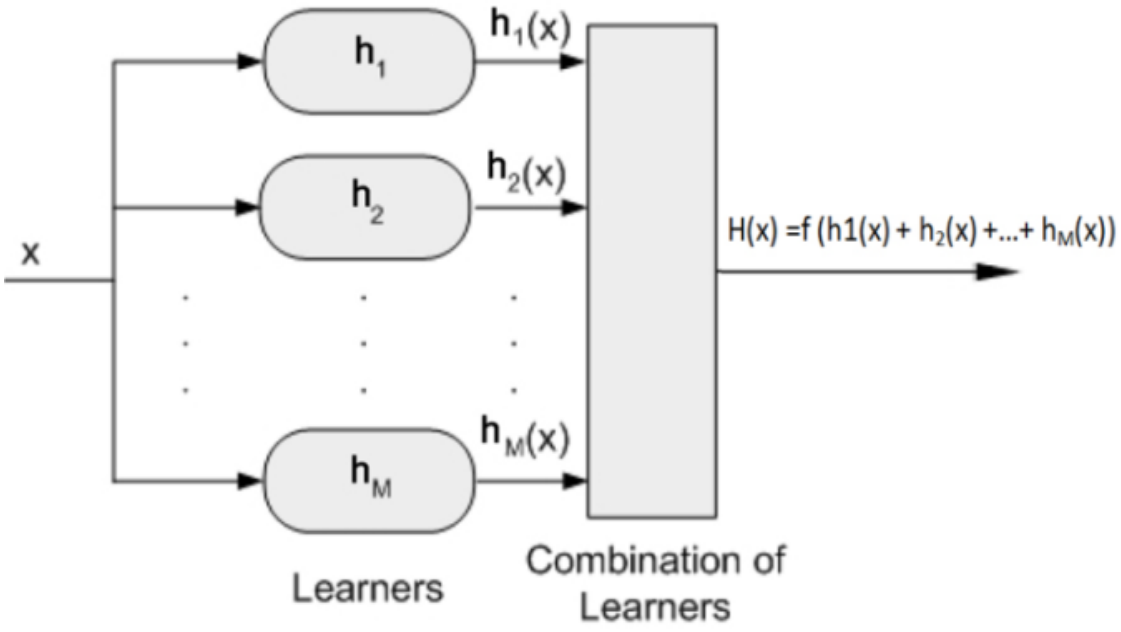

Figure 2.1:Le concept d'ensemble de classifieurs. Les sorties des apprenants faibles $h_m(x)$ avec m ∈{1,……..,M}sont combinés pour produire la sortie de l'ensemble des classifieurs donné par H(x).

Algorithme 1 : Algorithme Adaboost

Input: Dataset Z =$\{z_1, z_2, \ldots..., z_n\}$ with $z_i=\{x_i, y_i\}$,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.

Output : A classifier H : $\mathbf{Z} \rightarrow$ {-1 ,+1}.

1: Initialize the weights $w_i^{(1)}$= 1/N, i∈ {1……...,N} , and set m = 1 .

2: for m ≤ M do

3 : Run weak learner on Z, using weights $w_i^{(m)}$, yielding classifier
H_m :**Z** → {-1 ,+1}

4: $Computeerr_m = \sum_{i=1}^{N} w_i^{(m)} h\ (-y_i H_m(x_i))$, the weighted error of H_m.

5 : Compute $\alpha_m = \frac{1}{2}\log(\frac{1-err_m}{err_m})$.

6: For each sample i =1. ,……,N, update the weight :
$v_i^{(m)} = w_i^{(m)} \exp\ (-\alpha_m y_i H_m(x_i))$.

7 :Renormalize the weights: compute $S_m = \sum_{j=1}^{N} v_j$ and , for i =1. ,……,N, $w_i^{(m+1)} = v_i^{(m)} \Big/ S_m$

8: Increment the iteration counter: m← $m + 1$

9: end while

10: Final classifier :H(x)= sign $\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right)$.

3.2Algorithme Real AdaBoost :

Real Adaboost 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.

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 :

Algorithme 2 : Algorithme Real Adaboost

Input: Dataset Z =$\{z_1, z_2, \ldots\ldots, z_n\}$ with $z_i=\{x_i, y_i\}$,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.
Output : A classifier H : **Z** $\rightarrow$ {-1 ,+1}.
1: Initialize the weights w_i =1/N, i$\in$ {1……...,N}.
2: **for** m = 1 to M **do**
3: Fit the class probability estimate P_m(x)=P_w(y = 1 |x), using w_i.
4 :set H_m= $\frac{1}{2}$log((1-P_m(x))P_m(x))$\in \boldsymbol{R}$
5: Update the weights: $w_i \leftarrow w_i$ exp (-$y_i H_m(x_i)$)
6: Renormalize to weights.
7: **end for**
8: Final classifier :H(x)= sign $(\sum_{i=1}^{M} \alpha_i H_i(x))$.

3.3 Algorithme Logit Boost :

Les fondateurs de l'approche , Freund et Schapire en 1995 , ont toujours essayé d'améliorer cet algorithme , ce qui explique l'apparition de plusieurs variants de cette méthode qui optimisent différemment la pondération wi. On s'intéresse ici a l'algorithme Logit Boost qui répond au mieux au problème de classification basé sur l'utilisation de la règle de Bayes par le calcul de la probabilitè a posteriori P(y=$\frac{j}{x}$) ou j représente la classe a laquelle une obsérvation x appartient.

Les algorithmes « Boosting » sont considérés comme des procédures d'estimation pour la conception d'un modèle de régression logistique additive.

$$H(x) = (\sum_{1}^{T} \alpha_t H_t(x_i) \qquad \textbf{(2.3)}$$

Algorithme 3 : Algorithme Logit Boost

Input: Dataset Z ={z_1,z_2, ……,z_n} with z_i={x_i,y_i } ,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.
Output : A classifier H : **Z** → {-1 ,+1}.
1: Initialize the weights w_i =1 / N, i∈ {1……...,N}.
2: **for** m = 1 to M and while $H_m \neq 0$ **do**
3: Compute the working response $z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1-p(x_i))}$and weights w_i=$p(x_i)(1 - p(x_i))$
4 :Fit$H_m(x)$by a weighted least-squares of $z_i to x_i$, with weightsw_i.
5: Set H(x) = H(x) + $\frac{1}{2}H_m(x)$ and p(x) = $\frac{\exp(H(x))}{\exp(H(x))+ \exp(-H(x))}$
6: **end for**
7: Final classifier :H(x)= sign $\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right)$.

3.4 Algorithme Gentle AdaBoost :

Gentle AdaBoost est considérée comme autant une extension du Real adaboost . Cette approche est plus performante que Real adaboost étant donné sa stabilité et sa robustesse quand il s'agit des données bruitées est aux aberrants . Gentle AdaBoost minimise la fonction exponentielle de perte d'Adaboost en utilisant les étapes de newton .

Algorithme 4 : Algorithme Gentle AdaBoost

Input: Dataset Z ={z_1,z_2, ……,z_n} with z_i={x_i,y_i } ,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.
Output : A classifier H : **Z** → {-1 ,+1}.
1: Initialize the weights w_i =1 / N, i∈ {1……...,N}.
2: **for** m = 1 to M **do**
3: Train $H_m(x)$ by weighted least-squares of y_i to x_i , with weights w_i .
4 : Update $H(x) \leftarrow H(x) + H_m(x)$.
5 : Update $w_i \leftarrow w_i$ exp (-$y_i H_m(x_i)$) and renormalize to $\sum_i w_i = 1$.
6: **end for**
7: Final classifier :H(x)= sign $\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right)$.

3.5 Algorithme Modest AdaBoost :

Modest AdaBoost a été inspire par Vezhnevets et Al en 2005 .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 adaboost .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 .

Algorithme 5 : Algorithme Modest AdaBoost

Input: Dataset Z ={z_1,z_2, ...,z_n} with z_i={x_i,y_i } ,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.
Output : A classifier H : **Z** → {-1 ,+1}.
1: Initialize the weights w_i =1 / N, i∈ {1……...,N}.
2: **for** m = 1 to M and while $H_m \neq 0$ **do**
3: Train $H_m(x)$ by weighted least-squares of y_i to x_i , with weights w_i .
4 : Compute "inverted" distribution $\overline{w_i}$ = (1 - w_i) and renormalize to $\sum_i \overline{w_i} = 1$
5 : Compute $P_m{}^{+1}$ = P_w (y = +1, H_m (x)) , $\bar{p}_m{}^{+1}$ = $P_{\bar{w}}$(y = +1, H_m(x)).
6 : Compute $P_m{}^{-1}$ = P_w (y = - 1, H_m (x)) , $\bar{p}_m{}^{-1}$ = $P_{\bar{w}}$(y = - 1, H_m(x)).
7 : Set H_m (x) = ($P_m{}^{+1}(1 - P_m{}^{+1}) - P_m{}^{-1}$ ($1 - P_m{}^{-1}$)) .
8 : Update$w_i \leftarrow w_i$ exp (-$y_i H_m$ (x_i)) and renormalize to $\sum_i w_i = 1$.
9: **end for**
10: Final classifier :H(x)= sign $\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right)$.

3.6 : Algorithme Float Boost :

La variante Float Boost [29,30] est composée des étapes suivantes: 1-initialisation; Inclusion 2-forward; 3-exclusion conditionnelle; 4 sorties. Toutes ces étapes, avec l'exception de l'étape 3, sont similaires à celles d'AdaBoost et d'autres variantes discuté jusqu'à présent. La nouveauté ici est l'étape d'exclusion conditionnelle, dans laquelle leclassifieur faible le moins significatif est retiré de l'ensemble des classifieurs, sous réserve de condition que la suppression entraîne une erreur inférieure à un certain seuil.

Algorithme 6 : Algorithme Float Boost

Input: Dataset Z =$\{z_1, z_2, \ldots\ldots, z_n\}$ with z_i=$\{x_i, y_i\}$,where $x_i \in \boldsymbol{Z}$and $y_i \in$ {-1,+1};
M, the maximum number of classifiers.
N examples N = a + b; a examples havey_i =+ 1 and b examples havey_i = - 1.
J(H_M), the cost function and the maximum acceptable cost J^*

Output : A classifier H : **Z** → {-1 ,+1}.

1: { 1- Initialization stage 1. }
2 : Initialize the weights $w_i{}^0$= 1 / 2a , for those examples with y_i = +1.
3 : Initialize the weights $w_i{}^0$= 1 / 2b , for those examples with y_i = - 1.
4 : $J_m{}^{min}$= J* m = { 1,……,M_{max} } .
5 :M =0 , $\boldsymbol{H_0} = \{\}$.
6 : { 2 - Forward inclusion stage.}
7: M ← M + 1.
8 : Learn H_m(x)and $\boldsymbol{\alpha}_M$.
9 : Update : $w_i{}^{(M)} \leftarrow w_i{}^{(M-1)}$ exp (-$y_i H_M(x_i)$) and renormalize to $\sum_i w_i = 1$.
10 : $\mathbf{H}_M = \mathbf{H}_{M-1} \cup \{H_M\}$.
11 : **if** $J_M{}^{min} > J(H_M)$ **then**
12 : $J_M{}^{min} = J(H_M)$.
13 : **end if .**
14 : { 3 - Conditional exclusion stage .}
15 : h' =arg $min_{\boldsymbol{h} \in \mathbf{H}_M}$ J (H_M –h).
16 : **if** J (H_M – h') < $J_{M-1}{}^{min}$**then**.
17 :$\mathbf{H}_{M-1} = \mathbf{H}_M$ - h' .
18 :$J_{M-1}{}^{min}$ = J (H_M – h') .
19 : M ← M - 1.
20 : **if** h'= h'_m then
21 :Recalculate $w_i{}^j$ and h_j for j={m',……,M}.
22 :Goto line 15.
23 : **else**
24 :**if** M =M_{max} or J ($\mathbf{H}_M$) < J * **then**
25 : Goto line 32.
26 :**else**
27 :Goto line 7.
28 :**end if**
29 :**end if**
30 :**end if**
31 : {4 - Output stage.}
32: Final classifier : H(x)= sign $\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right)$.

4.Conclusion:

Dans ce chapitre j'ai éxpliqué le principe de boosting qui consiste à construire un classifieur fort à partir d'une combinaison pondérée de classifieurs faibles . ce principe a une grande utilité dans la méthode de Viola et Jones pour sélectionner les meilleures caractéristiques .enfin j'ai montré les différents algorithmes de ce principe .

Chapitre 3

Application

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.

2. Vue d'ensemble du système

J'utilise une caméra statique placée à une hauteur suffisante telle que. Les têtes des piétons sont visibles. Pour détecter les têtes, j'utilise la technique Viola et Jones. Le processus de formation est oline et crée un classificateur 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 de 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 camera
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 scene

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

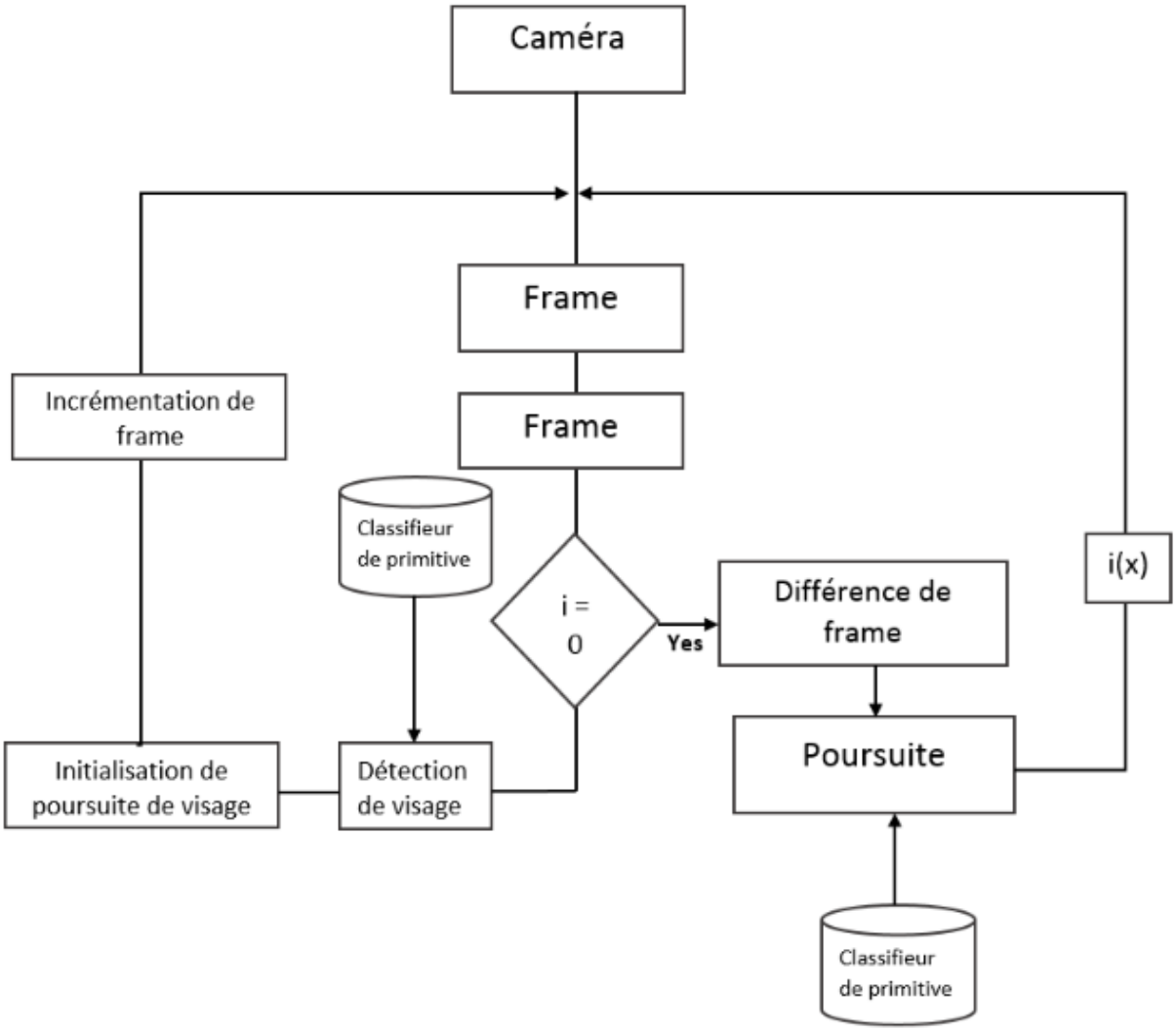

Figure 3.1 – Schéma d'apprentissage

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 and, 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. Les Etapes d'apprentissage :

— Collect des images d'apprentissage positives et negatives

— noter les images positives l 'aide de *objectmarker* or *ImageClipper* tools

— Creation d'un fichier vecteur *.vec* basé sur les images notées positives à l'aide *./createsamples*

— Entrainement du classifieur using ./haartraining

— Executer 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 seillé 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 crée 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 utlisé par la suite pour l'entrainement du classifieur par Boosting. Dans la figure ci-dessous, nous présentons un exemple des images 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 crée à l'aide de l'utilitaires : *Objectmarker*.

Les images positives sont sauvegardés dans le répertoire *../training/positive/rawdata* .ut you positive images L'utilitaire *objectmaker* est sauvegardé dans le répertoire *../training/positive* avec un fichier contenant les annotations des images positives. L'annotation du visage sur une image positive s'effectue par l'execution de l'utilitaire *./objectmaker*. Dans la figure ci-dessous nous proposons un exemple avec deux fenetre sur les terminaux :1 une fenetre avec le nom de l'image et 2 l'image positive. Le visage est selectionné sur l'image positive à l'aide de la souris. La souris est positionnée sur le coin supérieur et puis un deuxieme point est séléctionné au bas du visage. Cette procedure est réalisée pour toutes les images positives. A la fin nous aurons un fichier texte *info.txt* avec le contenu suivant :

*rawdata/image*0010.*bmp*134127424
*rawdata/image*0011.*bmp*335257039409580921204045 36
*rawdata/image*0012.*bmp*21024909045689982

Le premier chiffre donne une information sur le nombre visage existant sur chaque image ainsi que leurs positions. Pour l'image *image*0010.*bmp*, le nombre 1 veut dire qu'il y a un visage situé entre $x = 34$, $y = 12$, $width = 74$, $height = 24$.

Figure 3.2 – Exemple des image négatives

Figure 3.3 – Annotation d'une image positive

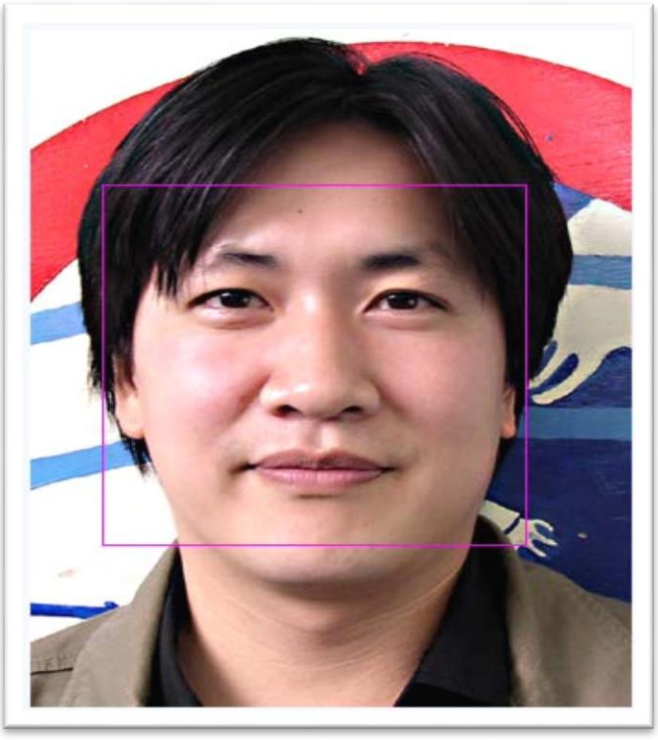

Figure 3.4 – Crop d'un visage sur une image positive

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−num*200−*w*24−*h*24
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'execution de l'apprentissage:
*haartraining − datacascades − vecvector/vector.vec − bgnegative/bg.txt − npos*200 *− nneg*200 *− nstages*15 *− mem*1024 *− modeALL − w*24 *− h*24*˘nonsym*
— data cascades Chemin pour enregistrer le classifieurs 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 _ no. en format *bmp*
— -nneg 200 Nombre des images négatives (patches) _ npos
— nstages 15 Le nombre de fois que l'apprentissage est executer
— mem 1024 la taille mémoire assignée en MB
— mode ALL pour fixer la fenêtre de largeur *w*24 et *h*24 Sample size
— nonsym Uniquement lorsque les visages ne sont pas bien allignés.
Les dimensions *˘W* et *˘H* utiliser dans *harrtraining.sh* dans identiqus à ceux utilisés pour la creation des images des visages *sample−creation.sh ./Harrtraining* collecte des nouveaux ensembles d'image négative à chaque étape de l'apprentissage, l'utilisation de l'option *˘nneg* limite la taille de ses ensemble d'images. Lorsque le script *./haartraning.sh* est excuté nous obtenons quelques information 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 dependant ST.THR FA : Fause Alarme EXP. ERR : Erreur de classifieur fort

```
Parent node: 0

*** 1 cluster ***
POS: 200 200 1.000000
NEG: 200 0.243605
BACKGROUND PROCESSING TIME: 0.01
Precalculation time: 8.09
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.915344| 1.000000| 1.000000| 0.067500|
+----+----+-+---------+---------+---------+---------+
|   2|100%|+|-1.761648| 1.000000| 1.000000| 0.050000|
+----+----+-+---------+---------+---------+---------+
|   3|100%|-|-1.040223| 1.000000| 0.325000| 0.027500|
+----+----+-+---------+---------+---------+---------+
Stage training time: 4.79
Number of used features: 3

Parent node: 0
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+---+
|  0|  1|
+---+---+
```

Figure 3.5 – Classification correcte des images positives

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

— Le nombre de primitive est important
— Fause détection augmente
— Le temps de calcul devient important

Etape 6. Creating the XML File : A la fin de l'apprentissage, dans le repértoire *../training/cascades/* un catalogue noté de 0 jusqu'a "N-1", avec N le nombre des

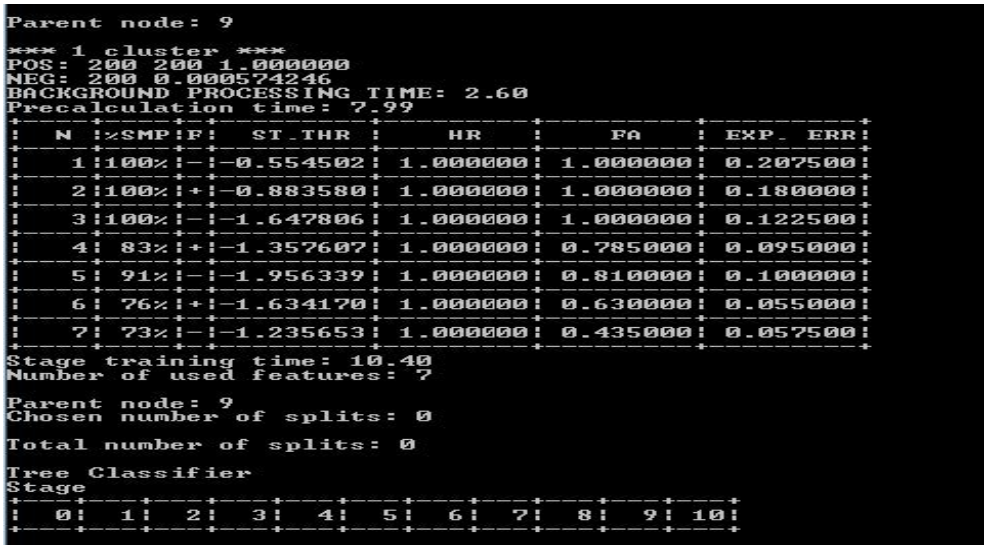

Figure 3.6 – Classification avec des erreurs des images positives

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 crées (classifieurs faibles) dans un seul classieur 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 : *./haarconv* data myfacedetector.xml 24x24. Le fichier *myfacedetecor.xml* est le fichier de sortie et 24x24 reprensente les dimension de visage sur une image (W et H).

4.Détection de visage :

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

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

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 représenté 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 estimés à partir d'un modèle de mouvement probabiliste, puis rééchantillonne les particules selon leur poids. La première La distribution du filtre est basée sur l'emplacement du visage la première fois qu'il est détecté. Voici les differentes étapes :

1. Pronostiquez : prédire distribution sur la position du visage à un instant donné.

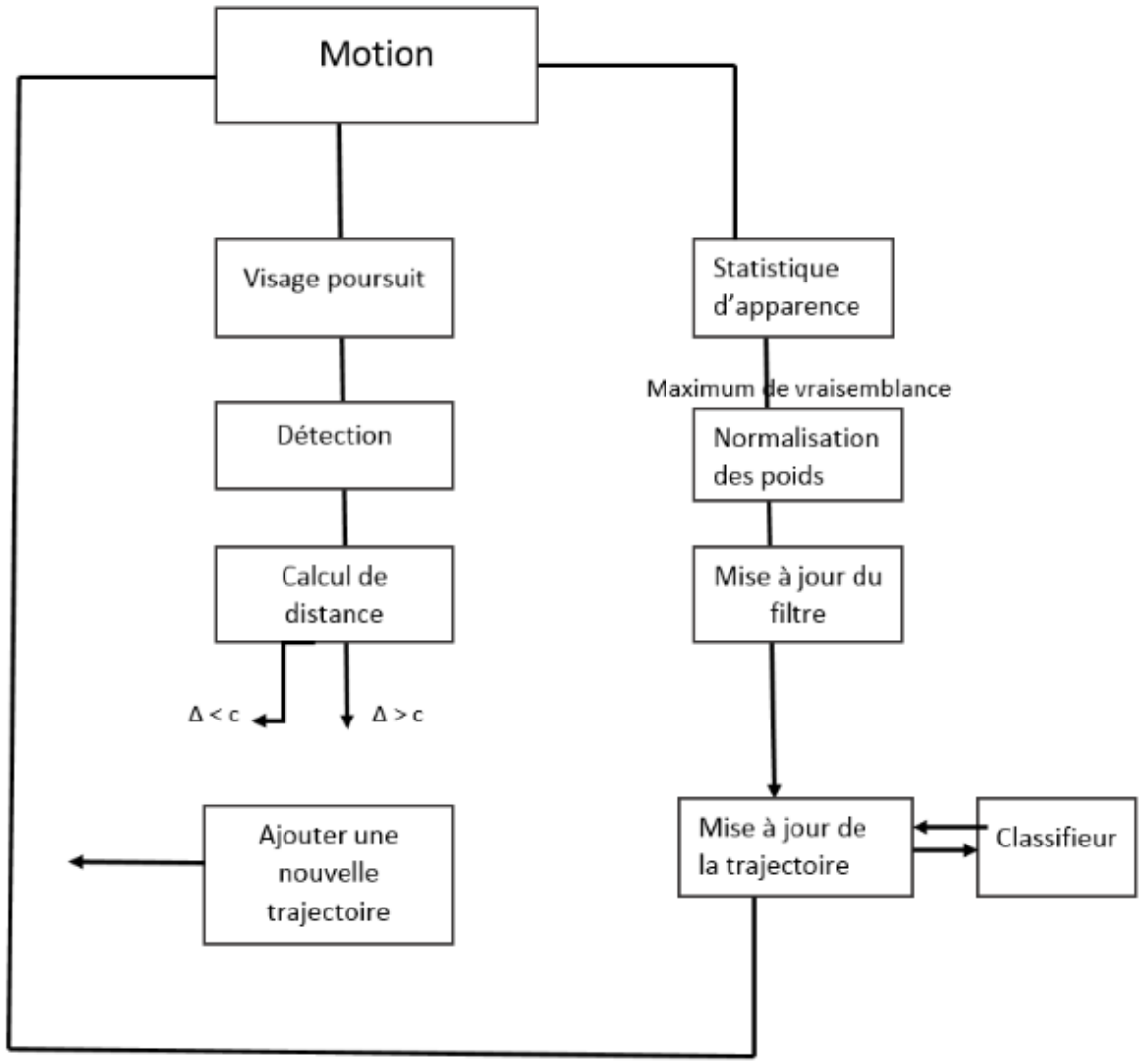

Figure 3.7 – Algorithme de poursuite de visage dans une foule

2. Mesurer : pour chaque particule propagée k, mesurons le poids du filtre à l'aide d'un modèle d'apparence basé sur l'histogramme couleur normalisé.
3. Rééchantillonnage : je 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 .

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 0*et*100% .
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 0*et*100%.
Dans OpenCV, la valeur H est normalisée en 0–180 ; les valeurs S et V sont normalisées en 0–255 (la Figure3.8).

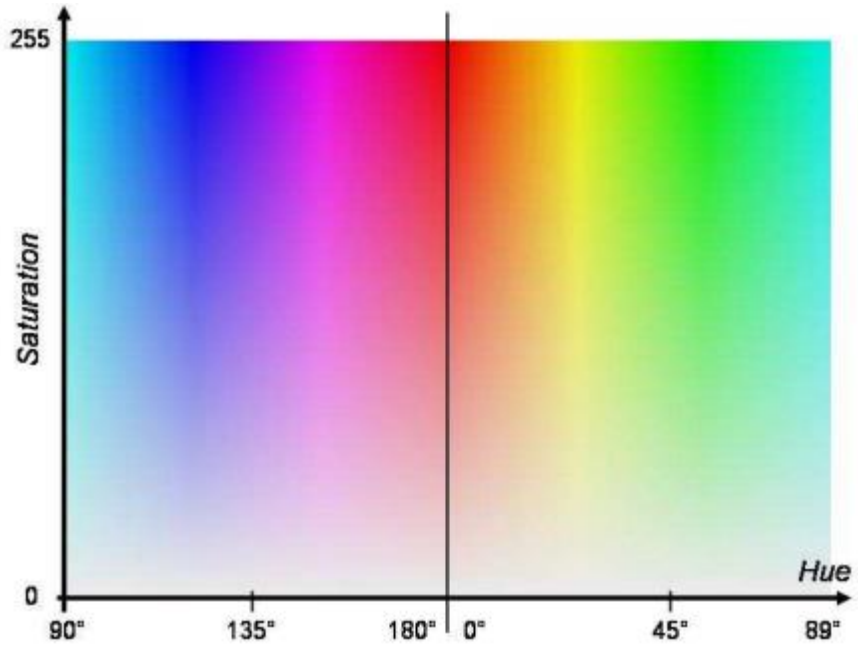

Figure 3.8 – Espace de couleur HSV

Le modèle HSV est une transformation non-linéaire de l'espace de couleur RGB. Il est défini dans OpenCV par la transformation ci-dessous :
$V = max(R, G, B)$
$S = (V - min(R, G, B)) \_ 255/V \text{ if } V \,!= 0, 0 \text{otherwise } (G - B) \_ 60/S, \text{ if } V = R$
$H = 180 + (B - R) \_ 60/S, \text{ if } V = G240 + (R - G) \_ 60/S, \text{ if } V = B \text{if} H < 0 \text{then } H = H + 360$
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 .

En considérant que la valeur de la " brillance" dela 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 960 composants pour l'apprentissage avec SVM. La Figure3.9 donne un exemple de l'histogramme H-S d'un visage segmenté.

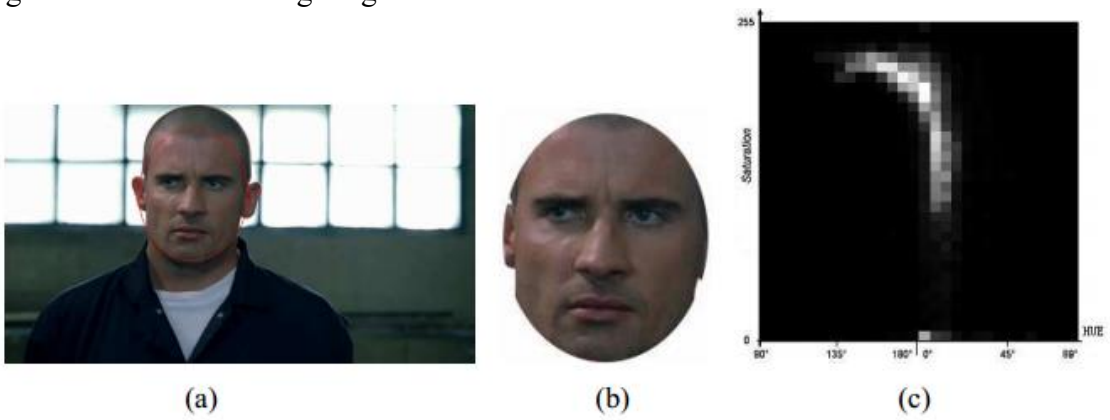

Figure 3.9 – Histogramme H-S (a) une frame (b) visage segmenté (c) Histogramme H-S

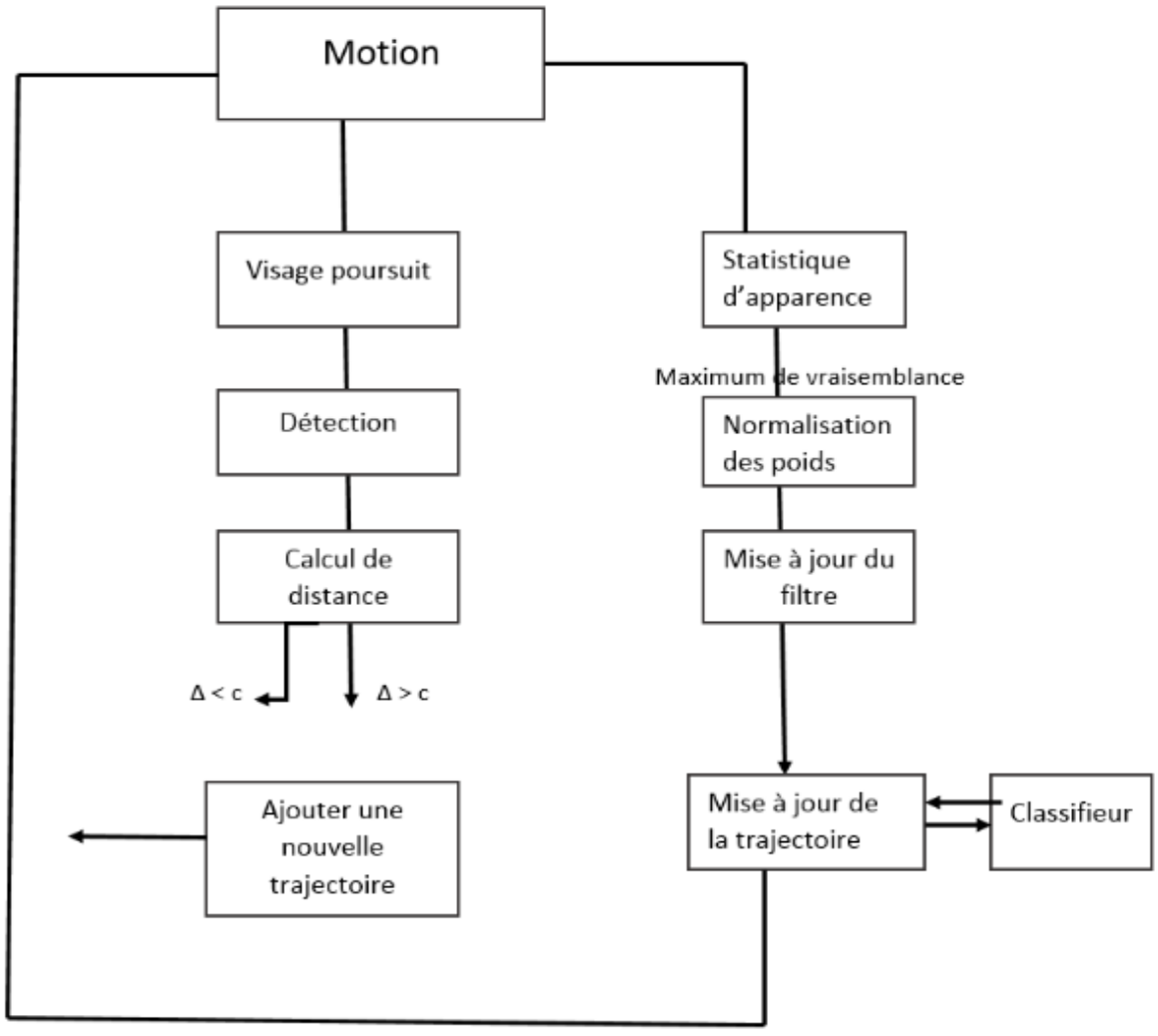

Figure 3.10 – Schéma du système de détection de visage dans une foule

6. Conclusion :

Dans notre travail d'extraction, nous avons extrait des caractéristiques de la couleur de l'espace HS et des caractéristiques de la texture basée sur les ondelettes de Gabor. 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.

Les fonctions noyaux seront étudiées pour la mesure de la similarité entre ensembles de vecteurs caratcteristiques à partir des tubes spatio-temporels. Nous pouvons envisager de chercher des fonctions noyaux qui sont sensibles à certaines conditions et d'autres fonctions noyaux qui sont sensibles à d'autres conditions. Une combinaison de ces fonctions noyaux pourra nous permettre d'obtenir un résultat plus robuste et plus pertinent.

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 incombent 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 j'ai 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 j'ai détaillé dans le chapitre 3 mes travaux de détection et l'approche pour améliorer le résultat obtenu en ajoutant un module de prétraitement.

Bien que ma 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, j'ai pu aborder la tâche de reconnaissance. Mon apport 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. je passe 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]Boudjellal. S. « Détection et identification de personne par méthode biomé- trique». Mémoire de Magister : Electronique-Télédétection. Tizi-ouzou : Université Mouloud Mammeri, 95p.

[2] Ming-Hsuan Yang, David J. Kriegman et Narendra Ahuja. Detecting faces images : « A survey». Dans IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.24(1), p 34-58, 2002.

[3]Cheng-Chin, Wen-Kai Tai, Mau-Tsuen Yang, Yi-Timg Huang, and chi-Janng Huang. « A novel method for detecting lips, eyes and faces in real time». Real-Time Imaging, 9(4) : 277-287, 2003.

[4] M.Van Wambeke, « Reconnaissance et suivi de visages et implémentation en robotique temps-réel» . Master Ingénieur Civil en Génie Biomédical. Belgique. Université catholique de Lovain,, 93p,2010.

[5] Wenlong Zheng and Suchendra M. Bhandarkar. « Face detection and haking using a boosted adaptative particle filter». Journal of visual communication and Imog Representation, 20(1) : 9-27, 2009.

[6] H. A. Rowley, S. Baluja, and T. Kanade. « Neural Network-based face detection». IEEE Transachen on Pattern Analysis and Machine Intelligence, 20(1) : 23-38, 1998.

[7] H. Schneidermand and T. Kanade. « Probabilistic modeling of local appearance and spacial relation ships for object recognition». Computer Vision and Pattern Recognition, IEEE Computer Society Conférence on 0 :45, 1998.

[8] Paul Viola and Michael Jones. «Robust real-time object detection» . In Second international work shop on statistical and computation atheories of vision, Vancouver, Canada, July 13 2001.

[9] Szeliski (2010) p. 653.

[10] Paul Viola et Michael Jones, « Robust Real-time Object Detection », *IJCV*, 2001.

[11]C. Papageorgiou, M. Oren et T. Poggio, « *A General Framework for ObjectDetection* », International Conference on Computer Vision, 1998.

[12] P. Viola, M. Jones et D. Snow, « *Detecting Pedestrians using Patterns of Motion and Appearance* », IJCV, vol. 63, n° 2, p. 153-161, 2005.

[13] H. Schneiderman, « Feature-centric Evaluation for Efficient Cascaded Object Detection », *IEEE CVPR*, 2004.

[14] L. Bourdev et J. Brandt, « Robust Object Detection via Soft Cascade », *CVPR*, p. 236-243 , 2005.

[15]H. Schneiderman et T. Kanade, « Object detection using the statistics ofparts », *IJCV*, 2002.

[16] R. Féraud, O. Bernier, J. Viallet et M. Collobert, « A fast and accurate face detector based on neural networks », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, nº 1, janvier 2001.

[17]Rainer Lienhart et Jochen Maydt, « An Extended Set of Haar-like Features for Rapid Object Detection », *IEEE ICIP*,2002 .

[18] Qiang Zhu, Shai Avidan, Mei C. Yeh et Kwang T. Cheng, « Fast Human Detection Using a Cascade of Histograms of Oriented Gradients », *IEEE CVPR*, p. 1491-1498,2006 .

[19] O. Tuzel, F. Porikli et P. Meer, « Human Detection via Classification on Riemannian Manifolds », *IEEE CVPR*, p. 1–8,2007 .

[20]M. Jones et P. Viola, « Fast Multi-view Face Detection », *IEEE CVPR*,2003 .

[21] C. Zhang et P. Viola, « Multiple instance pruning for learning efficient Cascade detectors », *Neural Information Processing Systems*,2007 .

[22]R. Xiao, L. Zhu et H.-J. Zhang, « Boosting Chain Learning for ObjectDetection », *ICCV*, p. 709-715,2003 .

[23]Y. Freund. «Boosting a weak learning algorithm by majority », Information and Computation. 1995.

[24]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.

[25] Y. Freund and R. Schapire. « A decision-theoretic generalization of on-line learning and an application to boosting». In European Conference on Computational Learning Theory – EuroCOLT. Springer, 1994.

[26] Y. Freund and R. Schapire. « Experiments with a new boosting algorithm». In Thirteenth International Conference on Machine Learning, p. 148–156, Bari, Italy, 1996.

[27] Y. Freund and R. Schapire. « A decision-theoretic generalization of on-line learning and an application to boosting». Journal of Computer and System Sciences, 55(1): p 119–139, 1997.

[28] T. Hastie, R. Tibshirani, and J. Friedman. « The Elements of Statistical Learning. Springer», 2nd edition, New York, NY, 2001.

[29] S. Li and A. Jain. « Handbook of Face Recognition. Springer», New York, NY, 2005.