République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Mémoire de Magister

Présenté à l'Université de Guelma
Faculté des Sciences et de l'Ingénierie

Département de : **Génie électrique**
Spécialité : **Signaux et Images**

Présenté par : **M^r^ OUDJANI BRAHIM**

Détection de Visages par Méthode Hybride AdaBoost et Couleur de Peau

Sous la Direction du : **Pr. Tebbikh Hicham**

Membres de jury:

Présidente :	**Dr**	**KECHIDA Sihem**	**Université de Guelma**
Rapporteur :	**Pr**	**TEBBIKH Hicham**	**Université de Guelma**
Examinateur :	**Dr**	**BOUKROUCHE Abdelhani**	**Université de Guelma**
Examinateur :	**Dr**	**ROUAÏNIA Mounira**	**Université de Skikda**
Invité d'honneur :	**Dr**	**BENCHERIET Chemesse ennahar**	**Université de Guelma**

Remerciements

Tout d’abord, nous tenons à remercier Allah, le tous puissant qui a éclairé notre chemin, et la patience qu’il m’adonné pour réaliser ce travail.

En guise de reconnaissance et de gratitude, mes sincères remerciements et ma profonde reconnaissance sont adressés à toutes les personnes, qui ont aidé à l’accomplissement de ce mémoire, notamment :

Monsieur TEBBIKH Hicham, professeur à l'Université de Guelma, directeur de Laboratoire d’Automatique & Informatique de Guelma« **LAIG** », président du conseil scientifique de la faculté des sciences et de l’ingénierie, pour m’avoir fait l’honneur par son encadrement, pour son très grande gentillesse, et pour le soutien et l’aide qu’il m’a apportée lors de mon travail dans le laboratoire « **LAIG** ». Soyez assuré de ma profonde gratitude, Sincères remerciements.

Madame Bencheriet Chemesse ennehar, Docteur Chargée de cours au département d'Informatique de l’université de Guelma, recevez mes remerciements sincères pour vos conseils éclairés, pour avoir suivi la réalisation de ce travail, et pour votre aide précieuse dans l’élaboration de ce document. Chaleureux remerciement.

Monsieur BOUKROUCHE Abdelhani, Docteur d’état, maître de conférence au département de Génie électrique de l’université de Guelma, qui nous a guidés nous tous dans notre carrière de Magister, pour ses conseils et sa disponibilité, et enfin pour accepter de participer à notre jury avec tant de gentillesse.

Dr KECHIDA Sihem, Maître de conférence à l’université de Guelma, pour l’intérêt qu’elle accordé à ce travail en acceptant de le juger et de présider le jury. Hommage respectueux.

Dr ROUAÏNIA Mounira, Maître de conférence à l’université de Skikda, pour m’avoir honoré de siéger au jury et de juger ce travail, et qu’elle trouve ici l’expression de notre gratitude.

Je doit aussi remercie Monsieur MOUSSAOUI Abdelkrim, Docteur maître de conférence à l’université de Guelma, chef département des Sciences et Techniques, de m’avoir même encouragé et conseillé, et qui avec efficacité m’a facilité des choses dans mes projets.

Je tiens également à remercier mes parents, pour tout l’embarras que je leur ai causé au long de mon chemin, je les remercie pour leur confiance aveugle mais récompensée, et qu’il voient dans ce travail et dans le reste de mon chemin l’amour et la fierté. Je vous aime. Je remercie aussi mes sœurs et frères, qu’ils m’ont épaulé et aimé dans les moments difficiles. Un grand merci à tous les membres de **LAIG,** à mes amis et mes collègues, merci pour tous les bons moments passés ensemble.

Résumé

Viola et Jones **[22]** ont présenté un nouveau et efficace algorithme de détection de visage basé sur des caractéristiques simples entraînés par l'algorithme AdaBoost. Deux contributions principales dans leur travail est l'utilisation des caractéristiques rectangulaires et l'introduction d'une nouvelle représentation d'image appelée "l'image intégrale" qui permet aux caractéristiques rectangulaires utilisées par le détecteur d'être calculées très rapidement. Ce travail essaye de replier leurs résultats mais en l'absence de l'architecture de cascade utilisée dans leur approche.

Pour notre détecteur de visage humain, nous nous sommes intéressés à l'hybridation de méthodes existantes dans la littérature. Pour cela, nous avons utilisé les filtres de rectangle (Filtre de Haar) entraînés par Adaboost et l'information couleur de peau, ce qui permet au détecteur de scanner uniquement les régions susceptibles d'être des régions de peau. Cela a permit de réduire le temps de calcul et aussi dans certains cas les fausses détections. Pour cela, des images de l'espace de couleur RGB peuvent être converties en RGB normalisé, YCbCr, HSV, HSI, YUV,…etc.

En se basant sur ce mécanisme, nous définissons 6 caractéristiques de rectangle d'un ensemble de 49554 caractéristiques et nous choisissons le meilleur espace de couleur ainsi la méthode pour la segmentation.

Notre algorithme de détection de visage est implémenté sur Matlab et mis en application sur une série d'images de test à différents degrés de complexités.

Mots clés : détection de visage, segmentation de peau, AdaBoost, filtres de rectangle.

Abstract:

Viola and Jones **[22]** presented a new and effective algorithm of face detection based on simple characteristics trained by the AdaBoost algorithm. Two principal contributions in their work is the use of the rectangular characteristics and the introduction of a new representation of image called "the integral image" which allows the rectangular characteristics used by the detector to be calculated very quickly. This work tries to fold up their results but in the absence of the architecture of cascade used in their approach. For our human face detector we precisely interested in test to make a hybridization of existing methods in the literature. For that, we used the filters of rectangle (Filter of Haar) learned by Adaboost and skin color information, which allows the detector scanning only the areas likely to be areas of skin. That is to reduce the computing time and also in certain cases of false detections. For that the images of space color RGB can be converted into normalized RGB, YCbCr, HSV, YUV….etc.

basing on this mechanism, we define 6 characteristics of rectangle of a set of 49554 characteristics, we use the algorithm of boosting for the training, and the best space of color and the method for the segmentation are selected.

Our algorithm of face detection is implemented on Matlab and applied on a series of the test images with various degrees of complexities

Keys words: face detection, skin segmentation, AdaBoost, filters of rectangle.

ملخص:

فيولا و جون**[22]**, قدما خوارزمية جديدة و فعالة لكشف الوجوه, معتمدة على وصاف (خصائص) بسيطة مدربة بواسطة خوارزمية أدابوست. خاصيتين أساسيتين في عملهما, هما استعمال وصاف مربعة و إدخال تمثيل جديد للصورة اسمه الصورة الأصلية, الذي يسمح الوصاف المربعة المستعملة من طرف الكاشف بأن تحسب بسرعة (في ظرف وجيز). في عملنا نحاول التقرب من النتائج المحصلة من طرف كاشف فيولا و جون لكن مع غياب بنية التسلسل الموجودة في عملهما.

في كاشفنا للوجوه, نهتم بمحاولة المزج بين الطرائق الموجودة في حاليا, ولهذا استعملنا الوصاف المربعة مدربة بواسطة أدابوست, و معلومة لون البشرة, التي تسمح لكاشفنا بالمسح فقط على المواقع التي يمكن أن تكون بشرة

نقترح الوصاف المربعة المدربة بواسطة أدابوست, و كذلك معلومة لون البشرة لتقليص زمن المعالجة و كذلك في بعض الأحيان الكشف الخاطئ, الأمر الذي يسمح للكاشف بالمسح فقط على المناطق المحتمل أن تكون مناطق لون بشرة, لأجل هذا, الصور في مجال اللون أر جي بي يمكن أن تحول إلى مجالات أخرى كلأر جي بي المسوى, الأش أس في...الخ.

اعتمادا على هذه الميكانيزمية, نعرف 6 أشكال للوصاف مربعة من ما مجموعه 49554 واصف, نستعمل خوارزمية أدابوست للتعليم,ونختار أفضل مجال للألوان وكذلك أفضل طريقة للبحث عن لون البشرة.

الخوارزمية المستعملة لكشف الوجوه برمجت على الماتلاب, ووضعت حيز التطبيق على سلسلة من الصور على مختلف درجات التعقيد.

الكلمات الدالة: الكشف عن الوجوه, شدف لون البشرة, أدابوست, الوصاف المربعة.

Sommaire

Sommaire

Introduction

Introduction :

Le sujet de la reconnaissance et de détection de visage humain est très ancien, en raison de son importance pratique et théorique, il reste toujours un centre important de recherche. Cela, est motivé par la multiplicité et la variété des champs d'application (haute sécurité, télésurveillance, contrôle d'accès…).

Les chercheurs ont montré que l'humain utilise pour reconnaître un visage ces différentes caractéristiques qui varient entre la géométrie, la texture et les couleurs des différentes régions du visage : les yeux, la bouche, le nez, le franc, les joues et le montant. Grâce à cette remarque, plusieurs études ont été développées afin de savoir s'il était possible de modéliser d'une manière informatique ce comportement.

Cependant, pour assurer de bons résultats dans la phase de la reconnaissance, le problème de la détection de visage est une procédure cruciale.

Les premiers systèmes de détection de visages ont été développés dans les années 70, ils sont efficaces dans peu d'applications, par exemple l'identification de photographie de passeport. Au début des années 90, plusieurs techniques ont été établies avec le progrès dans le codage de vidéo et la nécessité de l'identification de visage. Ces dernières années, différentes approches ont été développées pour résoudre le problème de détection de visages dans différents environnements et conditions.

Dans notre travail, nous voulons résoudre ce problème en utilisant une méthode basée sur l'information de couleur et les filtres de rectangle entraînés par Adaboost qui est l'un des meilleurs algorithmes d'apprentissage développé dans la dernière décennie. Notre travail se compose donc en deux parties ; une concerne le choix de la méthode de la segmentation de peau et l'espace de couleur adéquat et l'autre a pour objet d'essayer d'implémenter l'approche de Boosting de viola et jones sur les zones de peau préalablement détectées. Le principe de cette approche consiste à générer des classifieurs et à leur affecter une pondération. On construit ensuite ce que l'on appelle un classifieur agrégé (fort), en assimilant la prédiction de chaque classifieur à un vote, la classe prédite par le classifieur agrégé sera la classe majoritaire. L'idée maîtresse des méthodes est de construire des classifieurs

différents, en perturbant l'échantillon d'apprentissage, de manière à éparpiller l'erreur pour la rendre minoritaire.

Notre travail est réparti comme suit :

Le premier chapitre fournit un état de l’art assez complet de l'historique et des développements courants des techniques de détection de visages.

Le deuxième chapitre présente une introduction à l’algorithme AdaBoost, ces versions et sa modification au contexte de la détection de visage.

Dans le troisième chapitre on a abordé les techniques de la détection de peau humaine à savoir celle basée sur la création d’un modèle de peau et celle basée sur un simple seuillage suivi d’une étude comparative détaillée sur la segmentation de couleur chair dans différents espaces de couleurs.

L’implémentation de l’algorithme AdaBoost sur les zones peau détectées, les résultats obtenus ainsi que leur interprétations font l’objet du chapitre quatre.

Chapitre I

1.1. Introduction

Pour construire un système automatisé qui analyse l'information contenue dans les images de visage, des algorithmes efficaces et robustes de détection de visages sont exigés. En effet, vue l'importance de la détection de visage pour n'importe qu'elle système d'analyse de visage et dans le but d'identifier toutes les régions d'image qui contiennent un visage indépendamment de la position, de l'orientation ou de l'éclairage, plusieurs recherches ont été faites. En particulier, de nombreuses techniques ont été développées pour détecter des visages dans des images fixes, les plus importantes feront l'objet d'une illustration détaillée par catégorie.

1.2. Système de détection de visages

L'analyse de visage devenue un enjeu important de la vision. Modéliser la vision a toujours été un défi en informatique et en intelligence artificielle. Il apparaît qu'étudier le cerveau et la vision humaine peut faire progresser en vision par ordinateur. N'importe q'elle étape dans le processus de l'analyse de visages doit être précéder par l'étape de détection de visages.

Tout processus automatique de détection de visages doit prendre en compte plusieurs facteurs qui contribuent à la complexité de sa tâche, car le visage est une entité dynamique qui change constamment sous l'influence de plusieurs facteurs, et par la suit la démarche générale adoptée pour réaliser un tel système :

- Monde physique : Visages Humains.
- Codage: Acquisition d'images (Caméra,…etc), il consiste aussi à la digitalisation de l'image, et il comporte un risque de bruit.
- Prétraitement: Les étapes de prétraitement consistent à sélectionner l'information nécessaire à l'application. Cette sélection passe souvent par l'élimination du bruit dû aux conditions d'acquisition, par la normalisation des données, aussi bien que par la suppression de la redondance.
- Analyse : (appelée aussi indexation, représentation, modélisation ou extraction de caractéristiques), il faut extraire de l'image les informations qui seront

sauvegardées en mémoire pour être utilisées plus tard dans la phase Décision. Le choix de ces informations utiles revient à établir un modèle pour le visage, elles doivent être discriminantes et non redondantes.

- Apprentissage (si le système est basé sur l'apprentissage): consiste à mémoriser les représentations calculées dans la phase Analyse.
- Décision : Mesure de similarité.

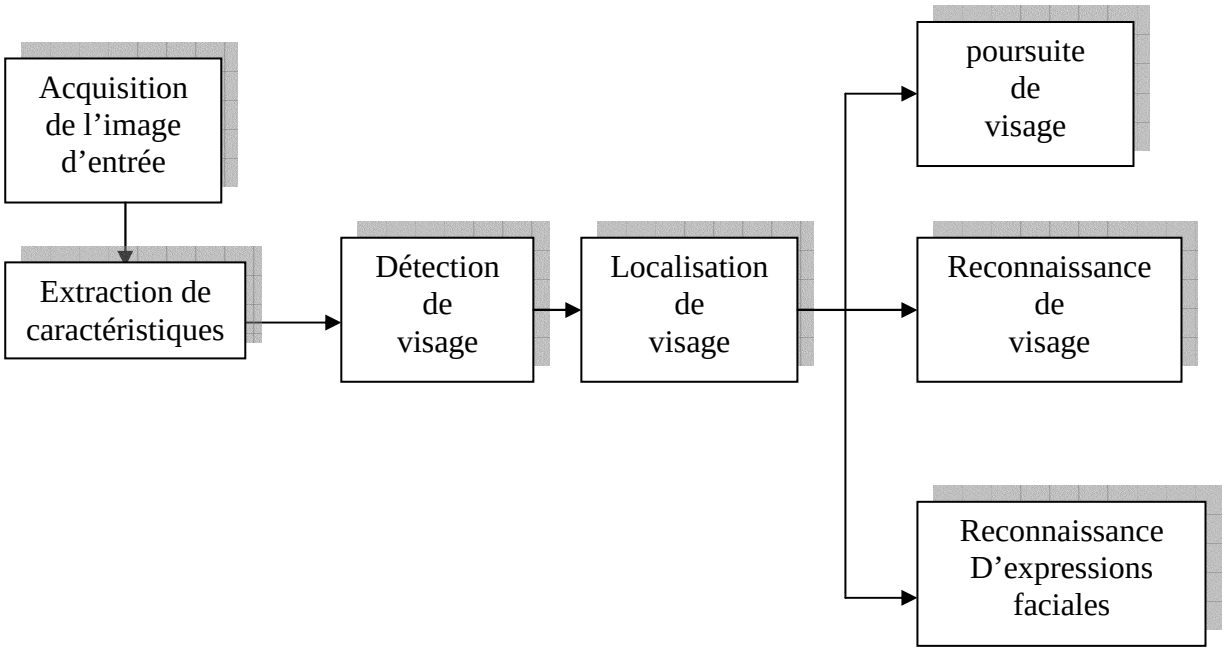


Figure 1.1 : Principales étapes d'analyse de visages

1.3. Difficultés de la détection de visages

La détection automatique de visage fait face à beaucoup de difficultés en raison de la très grande variabilité de la forme à détecter (image d'un visage quelconque, d'orientation et de taille quelconque avec un éclairement quelconque), les états d'expression faciales, d'occlusion et d'éclairage changent également l'aspect global des visages. En fait, il y a pas mal de défis en détection de visage dus aux conditions rigoureuses pour la classification des formes et les techniques d'apprentissage.

Les facteurs qui affectent la performance d'un système dans la détection de visage sont : l'échelle, la pose, l'illumination, l'expression faciale, l'occlusion.

1.3.1. L’échelle

Dans une image donnée, un groupe de visages peut apparaître dans différentes échelles (figure 1.2).

L'échelle (ou la taille) d'un visage peut être traitée par un processus simple de remise à l’échelle.

Figure 1.2 : visages à différentes échelles

1.3.2. Pose

Les performances d’un système de détection de visages chutent d’une manière significative quand les variations de pose sont présentes. Les poses variables se produisent d'un changement de point de vue ou quand la tête est tournée dans les trois-dimensions. Donc certaines caractéristiques faciales telles que l’oeil ou le nez peuvent devenir partiellement ou complètement occlus.

Figure 1.3 : visages à différentes poses

1.3.3. Occlusion

L'occlusion est une autre issue confrontée à la détection de visages dans la pratique. Les lunettes, les écharpes et les barbes tous changent l'aspect d'un visage. La figure 1.4 montre des exemples de visages sous différents déguisements.

Figure 1.4 : visages occlus

1.3.4. Expression faciale

L'expression faciale peut modifier d'une manière significative la géométrie d'un visage et donc influencer la décision (les résultats) d'un système de détection de visages (figure 1.5).

Figure 1.5 : visage à différentes expressions faciales

1.3.5. Illumination

Un même visage, avec la même expression faciale, et vu du même point de vue semble différent à cause des changements de l'éclairage. En fait, les changements provoqués par des différences de l'illumination sont souvent plus grands que les différences qui existent entre les individus, figure 1.6.

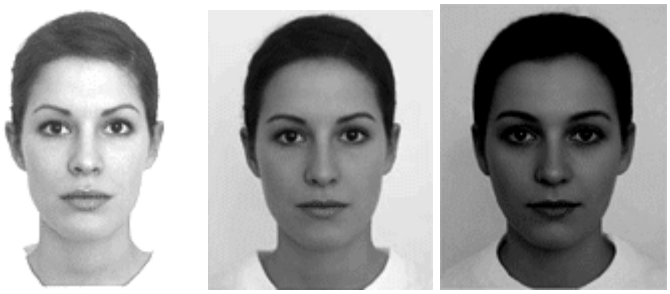


Figure 1.6 : visage sous différentes conditions d'illumination

1.4. Célèbres travaux de détection de visages

La détection de visage, quelle que soit la technique utilisée, se base toujours sur une capture d'image (vidéo par exemple) ou sur une photo. Les caractéristiques du visage sont ensuite extraites selon la méthode choisie, puis enregistrées dans une base de données.

Les travaux effectués sur la détection de visages se sont répartis en quatre approches théoriques.

- Approches basées sur les connaissances acquises
- Approches basées sur les caractéristiques invariables
- Approches basées sur la mise en correspondance « template matching »
- Approches basées sur l'apprentissage (ou l'aspect)

Certains chercheurs utilisent des techniques basées sur les connaissances acquises :

En 1994, Yang et Huang **[1]** ont proposé une hiérarchie de méthode basée sur les connaissances acquises pour détecter les visages. Dans cette méthode des règles dérivées à partir des connaissances ont été utilisés, comment un visage typique est constitué et quels sont les facteurs qui constituent celui-ci, habituellement, ces règles capturent les rapports entre les caractéristiques faciales, et la présence de la bouche, des yeux,…etc. Les positions relatives des différentes composantes du visage sont étudiées après avoir été détectées. L'avantage dans cette méthode est l'absence de n'importe quel apprentissage, le système est lié seulement par des règles qui ont été soigneusement définies. Le système de Yang et Huang s'est composé de trois niveaux. D'abord, une image est divisée en cellules de tailles égales. Au niveau 1, les règles qui décrivent quelques contraintes communes sur les modèles humains de

visage (template) ont été utilisées pour choisir les candidats de visage, alors dans ce niveau la partie centrale du visage a quatre cellules avec un niveau de gris fondamentalement uniforme. Les candidats de visage du niveau 1 sont passés sur le niveau 2 dans lequel la résolution de chaque candidat de visage est doublée, la taille de chaque cellule est réduite par la moitié et le template se compose maintenant de 8*8 cellules carrées.

Pour le niveau 2 un ensemble de règles de détection de visage ont été utilisés, et les candidats restants sont passés au niveau 3. Dans le niveau 3, l'égalisation d'histogramme locale, le multi-binarisation, et le suivi bi-directionnel de contour ont été utilisés. Puis les caractéristiques de contours des yeux et de la bouche ont été extraites. Si les caractéristiques extraites ressemble bien aux caractéristiques des yeux et de la bouche, le système l'identifierait comme modèle de visage.

Kotropoulous et Pitas **[2]** ont présenté une règle basée sur la méthode de localisation qui est similaire au travail de Yang et Huang. D'abord, les caractéristiques faciales sont localisées avec la méthode de projection qui est utilisée avec succès pour localiser les contours d'un visage. Soit I(x,y) la valeur d'intensité d'une image de taille m*n dans la position (x,y), les projections verticales et horizontales de l'image sont définies par:

$$HI(x) = \sum_{y=1}^{n} I(x, y) \text{ et } VI(y) = \sum_{x=1}^{m} I(x, y)$$

D'abord le profile horizontale d'une image d'entrée est obtenu, puis les deux minimums locaux, déterminés en détectant les changements brusques en HI, correspondent au côté gauche et droit de la tête. De même, le profil vertical est obtenu et les minimums locaux sont déterminés pour les endroits des lèvres, de bouche, du bout de nez, et des yeux. Ces caractéristiques détectées constituent un visage candidat

D'autres ont utilisés des techniques basées sur les caractéristiques invariables même lorsque la pose, le point de vue, ou les conditions d'éclairage changent:

Parmi ces techniques, celle qui détectent les caractéristiques faciales tels que les yeux, les narines, les sourcils, les lèvres, les oreilles, le nez, la bouche, etc..., et elles impliquent alors la présence d'un visage.

Graf et autres **[3]** ont développé en 1995 une méthode pour localiser les caractéristiques faciales et les visages dans des images en niveau de gris. Après le filtrage à bande passante, des opérations morphologiques sont appliqués pour augmenter les régions avec l'intensité élevée qui ont une certaine forme (par exemple, les yeux).

L'histogramme de l'image traitée montre typiquement une crête proéminente. Basé sur la valeur de la crête et sa largeur, les valeurs des seuils adaptatives sont choisies afin de produire deux images binaires. Des composants connexes sont identifiés dans les deux images binaires pour identifier les régions des caractéristiques faciales candidates. Des combinaisons de chaque régions sont alors évaluées avec des classificateurs pour déterminer si et où un visage est présent. Leur méthode a été testée avec des images tête-épaule de 40 individus et avec cinq séquences vidéo où chaque séquence se compose de 100 à 200 armatures.

Leung et autres **[4]** ont développé une méthode probabiliste pour localiser un visage dans une scène encombrée qui se base sur la détection des caractéristiques locales et la comparaison de graphique aléatoire. Ils ont formulé le problème de localisation de visage comme problème de recherche de certaines caractéristiques faciales qui sont plus susceptibles d'êtres un modèle de visage. Cinq caractéristiques (deux yeux, deux narines, et une jonction de nez/lèvre) sont utilisés pour décrire un visage typique. Pour n'importe quelle paire de caractéristiques faciales du même type (par exemple, paire de l'oeil gauche et l'oeil droit), leur distance relative est calculée. Pour un ensemble d'images, les distances sont modelées par une distribution gaussienne.

Une autre caractéristique efficace utilise la couleur de peau humaine pour la détection de visage lorsque l’image initiale est en couleur. Bien que la couleur de peau diffère parmi les individus, des études ont montré que la variabilité de la couleur de la peau tenait plus de la différence d’intensité plutôt que de la chromaticité. On peut utiliser plusieurs types d’espaces de couleurs (RGB, RGB normalisé, HSV, YcrCb, YIQ, YES, CIE XYZ, CIE LUV,…etc) :

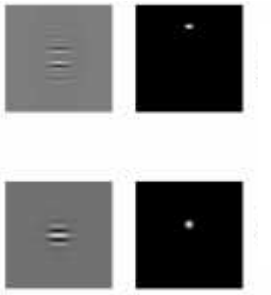
D.Chai et K. N. Ngan **[5]**, ont proposé une méthode pour la segmentation automatique du visage d'une personne à partir d'une image donnée qui se compose d'une vue de tête et épaules de la personne et d'une scène de fond complexe. La méthode implique un algorithme rapide, fiable, et efficace qui exploit les caractéristiques spatiales de la distribution de la couleur de peau humaine. Une carte universelle de couleur de peau est dérivée et utilisée sur le composant de chrominance de l'image d'entrée pour détecter les Pixels avec l'aspect de couleur de peau. Puis, basé sur la distribution spatiale des Pixels détectés de couleur de peau et de leurs valeurs correspondantes de luminance, l'algorithme utilise un ensemble de procédés de régularisation récentes pour renforcer les régions des Pixels de couleur de peau qui sont plus susceptibles d'appartenir aux régions faciales et pour éliminer ceux qui ne le sont pas.

McKenna et autres **[6]**, ont présenté un modèle adaptatif de mélange de couleur pour suivi des visages dans des conditions d'illumination variables. Au lieu de compter sur un modèle de couleur de peau basé sur la constance de couleur, ils ont utilisé un modèle stochastique pour estimer la distribution de la couleur d'un objet en temps réel et s'adapter aux changements des conditions d'éclairage et de visionnement. Les résultats préliminaires prouvent que leur système peut suivre des visages dans une marge des conditions d'illumination. Cependant, cette méthode ne peut pas être appliquée pour détecter des visages dans une image simple.

Le visage peut être aussi identifié à partir de sa texture en se basant sur des vecteurs qui décrivent au mieux les caractéristiques de la texture. Les approches les plus utilisées dans l'analyse de la texture sont les matrices de co-occurrences, les filtres de Gabor et les ondelettes.

Lin-Lin Huang, Shimizu et Kobatake **[7]**, proposent une méthode de détection de visage en utilisant les caractéristiques de filtre de Gabor. En considérant les caractéristiques désirables du filtre de Gabor telle que les sélectivités de localité spatial et l'orientation, ils conçoivent quatre filtres pour extraire les caractéristiques faciales à partir de l'image locale. Le vecteur de caractéristiques basé sur des filtres de Gabor est utilisé comme entrée du classificateur, qui est un réseau de neurone

polynomial (PNN) sur un sous-espace réduit de caractéristiques appris par l'analyse de composant principal (PCA). L'efficacité de la méthode proposée est démontrée par les résultats expérimentaux obtenus sur un grand nombre d'images et la comparaison avec les méthodes actuelles.



(a) **(b)**

Figure 1.7: (a) filtres de Gabor en domaine spatial, (b) domaine fréquentiel

Jie Chen, Shiguang Shan et autres **[8]**, Ont proposés une méthode de détection de visage basée sur les caractéristiques de Gabor, et propose un détecteur de visages hiérarchique combinant le rendement élevé des caractéristiques de Harr et l'excellente puissance distinctive des caractéristiques de Gabor.

D'autres chercheurs ont proposé de nombreuses techniques qui combinent plusieurs caractéristiques globales et locales pour localiser ou détecter des visages :
Yachida et autres **[9]** ont présenté en 1995 une méthode pour détecter des visages dans des images de couleur en utilisant la théorie floue. Ils avaient utilisé deux modèles flous pour décrire la distribution de couleur de peau et de cheveux dans l'espace de couleur de CIE XYZ. Cinq modèles de forme principaux (un frontale et quatre vues de côté) sont utilisés pour extraire l'aspect des visages dans les images. Chaque modèle de forme est un modèle 2D qui se compose de $m \times n$ cellules carrées où chaque cellule peut contenir plusieurs Pixel. Deux propriétés sont assignées à chaque cellule, la proportion de peau et la proportion de cheveux, qui indique les taux de la région de peau (ou de la région de cheveux). Dans une image de test, chaque Pixel est classifié comme cheveux, visage, cheveux-visage, et cheveux-fond basé sur les modèles de distribution. Les modèles principaux de forme sont alors comparés aux régions de peau et de cheveux extraits dans une image de test. S'ils

sont semblables, la région détectée va bien à un visage candidat. Pour la vérification, les caractéristiques d'oeil-sourcil et de nez-bouche sont extraites à partir d'un visage candidat en utilisant les contours horizontaux.

Sobottka et Pitas **[10]**, ont proposés une méthode de la localisation de visages et l'extraction des caractéristiques faciales en utilisant la forme et la couleur. D'abord, la segmentation de couleur dans l'espace HSV est effectuée pour localiser des régions de couleur de peau humaine. Dans cette étape, des composants connectés sont déterminés en appliquant l'algorithme de région croissant à une haute résolution de l'image segmentée. La recherche des visages dans une image peut être effectuée en détectant des objets avec la forme elliptique. L'avantage ici est que l'information de région est plus robuste contre le bruit et le changement de l'illumination. Les composants connectés qui sont bien rapprochés par une ellipse sont choisis comme des visages candidats. Enfin, ces candidats sont vérifiés en recherchant les caractéristiques faciales à l'intérieur des composants connectés.

Les techniques basées sur la mise en correspondance « template matching » ont été utilisées par **[11]** et **[12]**.

Luhong Liang, Haizhou Ai et Guangyou Xu **[11]**, ont proposé un système de détection de visage basé directement sur la technique de template. Un visage moyen est produit à partir de l'ensemble des photos de visages qui sont considérées comme des patterns de visage. D'après leur point de vue, les yeux jouent un rôle très important dans la détection de visages humains. Donc, ils utilisent un pattern des yeux et un pattern de visage indépendamment. Tout d'abord, des candidats de visage sont recherchés dans toutes les positions de l'image. Puis, un réseau de neurones qui est construit lors de la procédure de mise en correspondance est utilisé pour éviter les fausses détections.

Kim, Kang, Shin and Park **[12]**, présentent une combinaison d'un ellipse et la template matching, ils construisent un template de visage d'une image de contour horizontale. Après la construction de template un détecteur de contour appelé SUSAN est appliqué pour trouver les régions ou le template existe. La construction du template est un processus rapide. L'étape finale du détecteur est d'assortir une

ellipse aux régions d'intérêt identifiées, la longueur et la largeur de l'ellipse sont les deux produits du template de visage qui a été défini et utilisé pour tracer les régions d'intérêt de l'étape antérieure.

D’autres ont proposés des techniques basées sur l’apprentissage qui devraient capturer la variabilité représentative de l'aspect facial de la base d’apprentissage :
L’une de ces techniques se nomme "Eigenface", où l’image est décomposée en une série d’images teintées de nuances de gris.
Kohonen **[13]** a proposé en 1989 un exemple récent de l'utilisation des vecteurs propres dans l'identification de visages, dans lequel un réseau de neurone simple est démontré pour exécuter la reconnaissance de visages pour des images alignées et normalisées. Le réseau de neurone calcule une description de visages par l'approximation des vecteurs propres de la matrice de l'autocorrélation des images. Ces vecteurs propres sont connus sous le nom d'eigenfaces.

Les réseaux de neurones ont été appliqués avec succès dans les techniques basées sur l’apprentissage dans différents problèmes de reconnaissance de forme.
Agui et autres **[14]** ont proposé en 1992 une méthode en utilisant les réseaux de neurones hiérarchiques. La première étape se compose de deux sous-réseaux parallèles dans lesquels les entrées sont des valeurs d'intensité d'une image originale et des valeurs d'intensité d'image filtrée à l'aide d'un filtre de Sobel 3×3. Les entrées du réseau de la deuxième étape comprennent les sorties des sous-réseaux et les valeurs caractéristiques extraites telles que l'écart type des valeurs de Pixel dans le modèle d'entrée, le taux du nombre de Pixel blancs de tout le nombre des Pixels binarisé dans une fenêtre, et les moments géométriques. Une valeur de sortie à la deuxième étape indique la présence d'un visage dans la région d'entrée. Les résultats expérimentaux prouvent que cette méthode peut détecter des visages si tous les visages dans les images de test ont la même taille.
Rowley, Baluja et Kanade **[15]** ont proposé une méthode de détection de visage basée sur les réseaux de neurones. Ils ont formé les réseaux de neurones qui

pourraient distinguer entre le visage et les non-visages sur un grand ensemble de modèles d'image de visages et de non-visages.
Chaque réseau de neurones a renvoyé des points entre -1 et 1. Les points près de 1 ont indiqué un visage tandis que les points près de -1 pour les non-visages.
Une pyramide des images a été construite pour détecter des visages dans différentes échelles.

Une autre méthode des techniques de l'apprentissage appelée modèle de distribution. Sung et Poggio **[16]** ont présenté en 1998 un système de la détection de visages basé sur un modèle de distribution. Ce système décrit comment la distribution des patterns de visages peut être appris à partir des exemples de visages et de non visages. Leur système se compose d'un modèle de distribution des patterns de visages et de non-visages, et un classificateur qualifié.
À chaque endroit de l'image, un vecteur de caractéristique de différence est calculé entre le pattern local de l'image et le modèle de distribution. Le classificateur qui est en fait un réseau de neurones (MLP) détermine, en se basant sur les mesures de vecteur de caractéristique de différence, si un visage humain existe à l'endroit courant de l'image ou non.

Un ensemble de techniques d'apprentissage supervisé appelés SVMs (Support Vector Machine) destinées à résoudre des problèmes de discrimination et de régression est utilisé par plusieurs chercheurs pour le problème de la détection de visage.
Haizhou, Lihang Ying et Guangyou Xu **[17]**, proposent un système de détection de visages qui se compose de 2 étages de SVM : Tout d'abord, un SVM linéaire qui filtre des candidats de visage de toutes les fenêtres observées de l'image d'entrée. En suite, un SVM non linéaire prends des décisions finales si un candidat de visage est vraiment un visage ou non.
La fenêtre observée est normalisée à la taille 20x20. Des candidats de visage sont détectés en balayant cette fenêtre sur tous les endroits de l'image. Pour que le

système puisse découvrir des visages à différente taille, leur système applique le traitement des images de pyramide.

En fin, nous arrivons à la technique qui nous intéresse dans notre travail basée sur l'apprentissage. La partie majeur des attentions a été tournée vers l'approche de Boosting de Viola et Jones **[18]**, cette approche donne un très bon résultat en termes d'exactitude et vitesse, et bien adaptée aux applications en temps réel. Le boosting est une méthode mise au point pour des problèmes de classification, mais qui peut aussi être utilisée pour des problèmes de régression. Elle est de plus en plus utilisée en classification d'images, en particulier pour la phase d'apprentissage. Le boosting consiste à combiner les résultats obtenus par plusieurs classifieurs (faibles) pour obtenir un classifieur plus efficace (fort). L'algorithme de boosting le plus populaire est l'algorithme appelé AdaBoost, mis au point en 1997 par Freund et Shapire **[19]**.
Viola et jones **[18]** ont proposés un système de détection de visages en temps réel par AdaBoost c'est un nouveau et efficace algorithme de détection de visages, leur approche est basée sur les images intégrales et une structure de cascade ainsi que les filtres rectangulaires entraînés par AdaBoost. Ces filtres utilisés sont généralement présentés pour l'analyse de la texture. Cet ensemble de filtres a été évalué une première fois dans **[18,20,21,22]**, puis une seconde fois par Lienhart **[23]**. Le nouvel ensemble, qui est très utilisé aujourd'hui, est présenté sur la figure suivante:

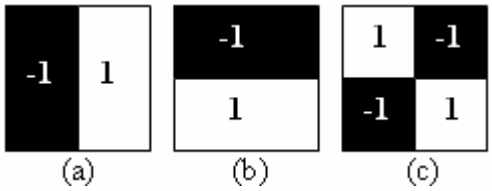


Figure 1.8 : Filtres de Haar
(a) filtre vertical. (b) filtre horizontal. (c) filtre diagonal.

Lienhart **[23]** a présenté et empiriquement analysé une autre extensions de l'approche de Viola et Jones. Il a crée un nouveau ensemble de caractéristiques orientées. Ces nouveaux caractéristiques enrichissent d'une manière significative les

caractéristiques simples de viola et jones, et peuvent également être calculés efficacement avec ces nouveaux caractéristiques orientées.

Beaucoup de chercheurs qui ont été attiré par cette méthode comme Shachar Kaufman et al **[24]**, leur projet est basé sur l'algorithme de Viola et Jones, et les améliorations apportées par Linheart et al, fournissent une solution fiable au problème de détection, pour les visages dans une scène et les yeux dans une image de visage. Cette solution réalise non seulement des résultats comparables à ceux réalisée par les meilleurs systèmes connus, mais en termes de taux de fausse alarme elle surpasse ces systèmes. Un système complet utilise le détecteur de visages de Viola et Jones , deux détecteurs d'oeil de Viola et Jones, un détecteur d'oeil basé sur SVM, et la reconnaissance de visages multi classes également basé sur SVM sont mis en application et combinés dans un système complet fournissant un mécanisme de fonctionnement pour le contrôle d'accès pour la porte de VISL (Vision and Image Sciences Laboratory) mise en application et donner un temps de traitement et une exactitude impressionnants.

D'autres chercheurs **[25]**, **[26]** ont utilisé l'Adaboost avec les histogrammes en détection de visage ou les filtres de Gabor pour la reconnaissance ou la vérification de visage.

1.5. Conclusion:

Dans une large gamme des approches générales, plusieurs aspects semblent être particulièrement efficaces dans les systèmes de détection de visage.

Les systèmes basés sur les connaissances acquises sont restrictifs, et ils sont liés par les règles dont ils sont l'origine.

Les Approches des caractéristiques invariantes sont bonnes mais se fondent sur l'invariance qui est souvent difficile de satisfaire dans toutes les conditions possibles.

Les Approches basées sur la mise en correspondance sont adéquates mais elles manquent de bonne représentation des composants non-objet des objets et sont limités par l'exactitude du template qui a été créé.

Les Approches basées sur l’aspect montrent que la plupart des promesses avec l'induction des techniques de la machine d’apprentissage, garantis des réalisations plus robustes.

L’approche de Viola et Jones semble avoir la meilleure application en utilisant l’architecture de cascade et l’image intégrale qui est absolument principale pour une application rapide des caractéristiques rectangulaires ou en d’autres termes les ondelettes de Haar.

Profitant des avantages de cette nouvelle technique qui rapporte de bons taux de détection et qui est fortement inspirée pour la détection en temps réel, en emphasant également sur deux contributions principales de bases de cette technique : la nouvelle représentation d'image et le choix des caractéristiques, on a essayé d’implémenter la technique en l’absence de la structure cascade et en utilisant l’information de couleur de peau pour limiter l’espace de recherche des visages. On a donc à disposition deux techniques différentes: technique basée sur l’apprentissage et technique basée sur les caractéristiques invariantes.

Chapitre II

2.1. Introduction

Les méthodes d'agrégation à base de votes sont devenues un enjeu important en classification. Des méthodes performantes de boosting telles que le Bagging **[27]** ou l'algorithme Adaboost **[28]** ont suscité un vif intérêt dans la communauté apprentissage. Elles présentent en effet d'excellentes performances en pratique et leur simplicité autorise des justifications théoriques séduisantes.

Le principe des méthodes à base de vote est très intuitif. Il consiste à générer des classifieurs et à leur affecter une pondération. On construit ensuite ce que l'on appelle un classifieur agrégé (fort), en assimilant la prédiction de chaque classifieur à un vote, la classe prédite par le classifieur agrégé sera la classe majoritaire. L'idée maîtresse des méthodes est de construire des classifieurs différents, en perturbant l'échantillon d'apprentissage, de manière à éparpiller l'erreur pour la rendre minoritaire.

2.2. Problématique

Considérons un algorithme A dans un ensemble de classifieurs H. On se pose le problème suivant:

Construire avec une complexité algorithmique "raisonnable" des combinaisons linéaires convexes des classifieurs de H pertinentes.

L'algorithme A dit "algorithme basique" ou "algorithme faible" sera à la base de la construction de telles combinaisons. Les principes d'induction visant à réduire l'erreur empirique ne sont donc pas fondés dans ce cas. D'autres principes d'induction doivent être mis en oeuvre utilisant d'autres justifications que les théorèmes classiques de convergence uniforme. Il y a deux stratégies différentes pour résoudre ce problème.

La première consiste dans un premier temps à générer indépendamment T classifieurs en perturbant l'échantillon d'apprentissage et dans un second temps à les pondérer afin d'optimiser un critère donné. Le BAGGING (Boostrap AGGregatING) est certainement l'exemple le plus connu.

La seconde, est une procédure itérative, désignée par l'acronyme ARCING (Adaptative Reweighting CombinING) par Breiman **[29]**. A l'étape t, le choix du

classifieur dépend des $t-1$ étapes précédentes, on met à jour une pondération α^t sur les exemples en fonction des $t-1$ étapes précédentes (on choisit initialement la pondération uniforme).

Le classifieur h_t construit est pondéré en fonction de tous les classifieurs déjà construits.

Breiman **[29]** distingue deux types d'algorithmes ARCING : les "algorithmes normalisés" ou les poids sont normalisés à chaque itération comme Adaboost, et les "algorithmes non-normalisés" ou les poids sont normalisés au final.

2.3. Idées de base du boosting

Les idées de base du Boosting sont présentées pour un problème de classification binaire pour rester dans le contexte de la détection avec la classe de visage et la classe de non visage.

Le but de la classification binaire est de trouver une « règle » (hypothèse) qui, basée sur un ensemble d'observations, assigne un objet à une des deux classes.

Soit X l'espace d'entrée qui contient les objets et on dénote l'ensemble des classes possibles par Y (dans notre cas $Y = \{-1,+1\}$).

La tâche d'apprentissage peut être résumées comme suit: Il s'agit d'estimer une fonction $f : X \to Y$, utilisant les paires d'entrée/Sortie des données d'apprentissage générées au hasard indépendamment d'une distribution de probabilité inconnu $P(x, y)$.

$(x_1, y_1),...(x_n y_n) \in R^d \times \{-1,+1\}$ .....(2.1)

Telle que f va prévoir correctement les exemples (x, y) inconnus. Dans le cas où Y = (-1, +1), nous avons soi-disant un classifieur fort, et l'étiquette attribuée à une entrée x est donnée par $y = f(x)$.

La vraie performance du classifieur f est évaluée par :

$L(f) = \int \lambda(f(x), y) dP(x, y)$. .....(2.2)

Où λ est la fonction de coût choisie. Le risque *L(f)* est souvent appelé erreur de généralisation ou erreur de test, dans le sens qu'il mesure le coût à l'égard de l'exemple inconnu dans l'ensemble d'apprentissage.

Pour la classification binaire, nous utilisons souvent la fonction de coût $\lambda(f(x), y) = I(y.f(x) \leq 0)$, ou $I(E) = 1$ si l'évènement E survient, et 0 autrement.

En d'autres termes :

$$\lambda(f(x_i), y_i) = \begin{cases} 1 & si \quad x_i \; est \quad mal \quad classé \\ 0 & si \quad non \end{cases} \qquad \ldots\ldots\ldots\ldots(2.3)$$

Mal classé c-à-d : $f(x_i) \neq y_i$.

La distribution de probabilité $P(x, y)$ est inconnu, ce risque *L(f)* ne peut pas être directement minimisé. Il faut donc estimer une fonction aussi proche que possible de f_{optimal} à l'aide des informations disponibles, c'est-à-dire les exemples d'apprentissage et les propriétés de la classe des fonctions *F* à partir de laquelle *f* est choisie.

Une solution classique est d'approximer l'erreur de généralisation par le risque empirique défini comme suit:

$$\hat{L}(f) = \frac{1}{N} \sum_{n=1}^{N} \lambda(f(x_n), y_n) \ldots\ldots\ldots\ldots(2.4)$$

C'est le cas si les exemples sont uniformément distribués.

Si l'ensemble d'apprentissage est suffisamment important, nous espérons que:

$$\lim_{N \to \infty} \hat{L}(f) = L(f) \ldots\ldots\ldots\ldots(2.5)$$

Toutefois, une forte condition est nécessaire pour valider la dernière formule, l'erreur de risque $\hat{L}(f)$ doit convergé uniformément sur la classe des fonctions *F* à $L(f)$.

Mais, pour pouvoir garantir à ce que la fonction *f* obtenue en minimisant $\hat{L}(f)$ converge asymptotiquement vers le minimum de $L(f)$, il faut qu'une autre condition soit satisfaite. Intuitivement, la classe des fonctions *F* ne peut pas être trop complexe. Autrement, on peut trouver une fonction *f* qui possède une erreur de

classification sur les données d'apprentissage arbitrairement petite mais une grande erreur de généralisation $L(f)$. Ce phénomène est appelé « overfitting ». Une condition suffisante pour éviter « l'overfitting » est l'exigence que $\hat{L}(f)$ converge uniformément (à travers F) vers $L(f)$.

Dans le cas où la base de données d'apprentissage n'est pas très grande, ce qui est presque toujours le cas en réalité, les conditions précédentes ne sont pas respectées et un grand écart peut avoir lieu entre l'erreur de généralisation $L(f)$ et le risque empirique $\hat{L}(f)$. Dans ce cas, le phénomène appelé « overfitting » peut de nouveau apparaître car une faible erreur de généralisation $L(f)$ ne peut pas être obtenu en minimisant simplement l'erreur empirique $\hat{L}(f)$. La solution pour faire disparaître le phénomène appelé « l'overfitting » est de limiter la taille de la classe des fonctions F**.** Le choix de fonctions qui classifient la majorité des données d'apprentissage correctement est préférable au choix de fonctions qui classifient presque toutes les données d'apprentissage correctement.

Pour les algorithmes de Boosting, il a été montré que sous certaines conditions la complexité de la classe des fonctions F sature en augmentant le nombre d'hypothèses utilisées pour créer la fonction de classification f. On pourrait donc croire que le phénomène de « l'overfitting » n'est pas possible. Ceci est faux, spécialement lorsqu'on utilise des procédures de Boosting avec des données d'apprentissage bruitées. Lors du Boosting, il faut donc aussi être attentif à la taille de la classe des fonctions F.

Les deux méthodes de Boosting les plus connues sont AdaBoost et LogitBoost, dans les paragraphes suivants on va parler justement sur AdaBoost.

2.4. AdaBoost :

La méthode de Adaboost (Adaptive Boosting) permet de combiner plusieurs hypothèses simples pour créer une autre hypothèse plus performante.

L'algorithme Adaboost a été proposé par Freund et Schapire au début des années 90. C'est un algorithme de type ARCING qui étonne autant par sa simplicité que par

ses performances expérimentales. Il a suscité un vif enthousiasme dans la communauté apprentissage. Bien que de nombreuses tentatives d'amélioration aient été faites, l'algorithme initial Adaboost reste le plus séduisant.

L'origine de sa mise au point remonte à une question de Schapire qui cherchait à construire des algorithmes de boosting.

L'idée de base du Boosting est donc de combiner des « règles » simples (hypothèses simples) pour créer un ensemble dont la performance de chaque élément de l'ensemble est amplifié. L'ensemble composé des hypothèses est défini de la manière suivante :

Soit $h_1, h_2, ..., h_T$ un ensemble d'hypothèse simple, et considérons l'ensemble de composite de l'hypothèse :

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \quad \text{.....(2.6)}$$

avec α_t le poids qui est attribué à l'hypothèse h_t de l'ensemble. Les poids α_t et les hypothèses h_t doivent être entraînés pendant la procédure de Boosting.

En générale, il y a plusieurs approches possibles pour sélectionner les coefficients α_t et les hypothèses h_t. Lors du Boosting les coefficients α_t et les hypothèses h_t sont sélectionnés itérativement à l'aide d'exemples d'apprentissage pondérés. À chaque itération, les poids des exemples d'apprentissage sont recalculés de manière à attribuer une grande pondération aux exemples d'apprentissage mal classifiés et une faible pondération aux autres (Figure 2.1). Cette technique permet de concentrer la procédure d'apprentissage sur les exemples durs à classifier.

Le classifier final (fort) est la combinaison des tous les classifieurs faibles :

$$H(x) = sign(\alpha_1 \cdot h_1(x) + \alpha_2 \cdot h_2(x) + \alpha_3 \cdot h_3(x)) \quad \text{.....(2.7)}$$

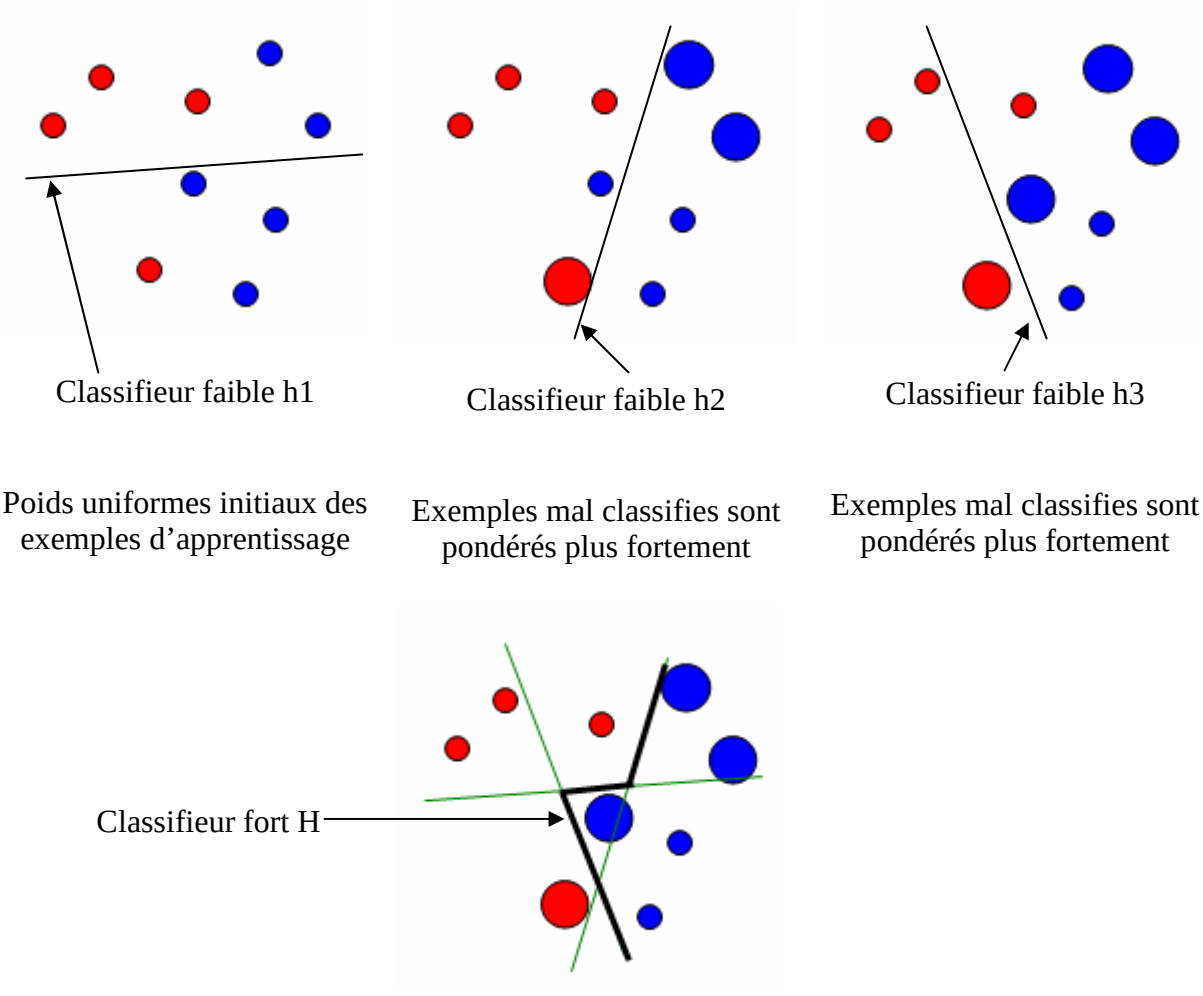


Figure 2.1: Description de AdaBoost

2.5. Versions de l'algorithme AdaBoost

On peut distinguer deux versions de l'algorithme AdaBoost dénotés par **AdaBoost.M1** et **AdaBoost.M2 [30]**. Les deux versions sont équivalentes pour des problèmes binaires de classification et diffèrent seulement dans leur manipulation des problèmes avec plus de deux classes.

2.5.1. Algorithme AdaBoost.M1

Entrée : une séquence des exemples m $((x_1, y_1),....,(x_m, y_m))$ avec les étiquettes $y_i \in Y = \{1,...k\}$.

- Construire l'algorithme de faible apprenant
- Donnée un nombre d'itérations T

- Pour tout i, initialiser $D_t(i) = \frac{1}{m}$ ……(2.8)
- Pour t= 1,2,…,T faire :
  1. appeler l'algorithme de faible apprenant, lui fournissant la distribution D_t.
  2. récupérer une hypothèse $h_t : X \to Y$.
  3. calculer l'erreur de h_t : $\varepsilon_t = \sum_{i:ht(xi)\neq yi} D_t(i)$ …….(2.9)

     Si $\varepsilon_t > \frac{1}{2}$, alors T= t-1.
  4. $\beta_t = \varepsilon_t /(1-\varepsilon_t)$ ……(2.10)
  5. mise à jour de la distribution

     $D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if \quad h_t(x_i) = y_i \\ 1 & \text{sin } on \end{cases}$ ……(2.11)

Où Z_t est le constant de normalisation (choisi de sorte que D_{t+1} doit être une distribution).

A la sortie l'hypothèse finale : $h_{fin}(x) = \arg\max_{y\in Y} \sum_{t:ht(x)=y} \log \frac{1}{\beta_t}$ …….(2.12)

L'algorithme prend comme entrée une base d'apprentissage de n exemples $S = (x_i, y_i), \ldots (x_n, y_n)$ ou x_i est un exemple tiré d'un certain espace X et représenté sous une certaine manière (typiquement, un vecteur des valeurs caractéristique), et $y_i \in Y$ est l'étiquette de classe associée à x_i.

L'algorithme de boosting a accès à un autre algorithme d'apprentissage non spécifié, appelé l'algorithme d'apprentissage faible ou apprenant faible, Cet apprenant faible se répète dans une série d'itérations, à l'itération t l'algorithme de boosting fournit le faible apprenant avec une distribution D_t sur les exemples d'apprentissage, par conséquent, le faible apprenant calcule un classifieur ou l'hypothèse h_t qui devrait bien classifier une fraction signifiante des exemples d'apprentissage, relativement à D_t. C'est-à-dire, le but du faible apprenant est de trouver une hypothèse h_t qui minimise l'erreur d'apprentissage ε_t, que cette erreur est mesurée à l'égard de la distribution D_t qui a été fournie au faible apprenant. Ce processus continue pour

des T itérations, et, enfin, l'algorithme de boosting combine les hypothèses faibles $h_1,....,h_T$ dans une hypothèse finale unique h_{fin}.

2.5.2. Algorithme AdaBoost.M2

Entrée : une séquence des exemples m $((x_1, y_1),....,(x_m, y_m))$ avec les étiquettes $y_i \in Y = \{1,...k\}$

- L'algorithme de faible apprenant
- Donnée un nombre d'itérations T
- Soit $B = \{(i, y) : i \in \{1,...m\}, y \neq y_i\}$…….(2.13)
- Pour $(i, y) \in B$, initialiser $D_t(i, y) = \frac{1}{|B|}$ ……(2.14)
- Pour t= 1,2,…,T faire
  1. appeler l'algorithme de faible apprenant, lui fournissant la distribution D_t.
  2. récupérer une hypothèse $h_t : X \times Y \to [0,1]$.
  3. calculer le pseudo-perte de

     $h_t : \varepsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$ ..(2.15)
  4. $\beta_t = \varepsilon_t /(1 - \varepsilon_t)$ …….(2.16)
  5. mise à jour la distribution

     $D_t : D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \beta_t \exp(1/2)(1 + h_t(x_i, y_i) - h_t(x_i, y))$ …….(2.17)

Où Z_t est le constant de normalisation (choisi de sorte que D_{t+1} doit être une distribution).

A la sortie l'hypothèse finale : $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) h_t(x, y)$ …….(2.18)

La deuxième version d'AdaBoost essaye de surmonter quelques difficultés liées à la première version en prolongeant la communication entre l'algorithme boosting et le faible apprenant. D'abord, le faible apprenant produit des hypothèses plus expressives dont la sortie est un vecteur dans $[0,1]^k$, plutôt qu'une étiquette

simple en Y. Intuitivement, le y^{ieme} composant de ce vecteur représente un "degré de croyance" dont l'étiquette correcte est y. Les composants avec des valeurs près de 1 ou de 0 correspondent à ces étiquettes considérées comme plausibles ou non plausibles, respectivement.

Tandis que un algorithme de faible apprenant d'une puissance plus expressive est donnée, également une condition plus complexe sur la performance des hypothèses faibles est placée.

Plutôt qu'en utilisant l'erreur habituelle de prévision, les hypothèses faibles font bien en ce qui concerne une mesure plus sophistiquée de l'erreur appelée la pseudo-perte.

À la différence de l'erreur ordinaire qui est calculée à l'égard d'une distribution sur des exemples, la pseudo-perte est calculée à l'égard d'une distribution sur l'ensemble de toutes les paires des exemples d'apprentissage et les étiquettes incorrectes.

Plus formellement, une étiquette mal est une paire (i, y) où i est l'index d'un exemple d'apprentissage et y est une étiquette incorrecte liée à l'exemple i. Soit B l'ensemble de toutes les étiquettes mal:

$B = \{(i, y) : i \in \{1,....., m\}, y \neq y_i\}$.

Une distribution d'une étiquette mal est une distribution définie sur l'ensemble B de toutes les étiquettes mal.

Dans chaque itération t de boosting, Adaboost.M2 fournit au faible apprenant une distribution D_t d'étiquette mal. Ainsi le faible apprenant calcule une hypothèse h_t de la forme :

$h_t : X \times Y \rightarrow [0,1]$

Intuitivement, le poids $D_t(i, y)$ assigné à cette étiquette mal représente l'importance de distinguer l'étiquette incorrecte y sur l'exemple x_i.

Une hypothèse faible est alors interpréter de la manière suivante :

Si $h_t(x_i, y_i) = 1$ et $h_t(x_i, y) = 0$ alors h_t a prévu correctement que les étiquettes x_i sont y_i et pas y (puisque h_t considère y_i pour être plausible et y non plausible). De même, Si $h_t(x_i, y_i) = 0$ et $h_t(x_i, y) = 1$, alors h_t a incorrectement fait la prévision

opposée. Si $h_t(x_i, y_i) = h_t(x_i, y)$, alors la prévision de h_t est prise pour être une conjecture aléatoire. (c-à-d, les valeurs de h_t dans (0.1) sont interprétées probablement). cette interprétation nous mène à définir la pseudo-perte de l'hypothèse h_t à l'égard de la distribution d'étiquette mal D_t par la formule:

$$\varepsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y)) .$$

Le but de faible apprenant est de trouver une hypothèse faible h_t avec la petite pseudo-perte.

Après la réception de h_t, la distribution d'étiquette mal est mise à jour en utilisant une règle semblable à celle utilisée dans Adaboost.M1. L'hypothèse finale h_{fin} produit pour un exemple donné x l'étiquette y qui maximise la moyenne pondérée des valeurs de l'hypothèse faible $h_t(x, y)$.

2.6. AdaBoost de Viola et Jones pour la détection de visages

Dans ce paragraphe on décrit un algorithme d'Adaboost.M1 modifié pour être mieux adapté au contexte de la détection de visage. Viola et Jones ont proposé un algorithme AdaBoost.M1 avec une architecture de cascade et lui appliqué avec succès à la détection de visages. Dans AdaBoost de Viola, le classifieur faible a été établie par une caractéristique rectangulaire, l'apprentissage par AdaBoost est adopté pour combiner ces classificateurs faibles. Par conséquent, dans un certain sens, dans AdaBoost de Viola, le classificateur faible est un peu équivalent à une caractéristique faible. Pour le problème spécifique de la détection de visages, Viola a conçu un algorithme rapide pour appliquer un nombre énorme de caractéristiques de rectangle à une petite région d'une fenêtre candidate, dont quelques unes sont alors choisies et combinées pour former un classificateur fort. De plus, la détection de visages basée sur AdaBoost a été reconnue en tant que la plus réussie pour la tâche de la détection de visages.

2.6.1. Algorithme AdaBoost :

Soient les images d'exemple $S = \{(x_1, y_1),...,(x_n, y_n)\}$, ou $Y_i = 0;1$ pour les exemples négatifs et positifs respectivement, Ou $i = 1,...,n$

- Initialiser les poids $W_{t,i} = \frac{1}{2l}; \frac{1}{2m}$ ……(2.19)

Pour $Y_i = 0;1$ respectivement, ou l et m sont des nombres des exemples négatifs et positifs respectivement.

- pour t=1,…, T :

1. normaliser les poids.

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \quad \text{……(2.20)}$$

Alors que, w_t est une distribution de probabilité.

2. pour chaque attribut (feature) ,j, former un classifieur h_j

Ce qui est limité à utiliser un seul attribut (feature).

L'erreur est évaluée à l'égard de w_t , $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$. ……..(2.21)

3. choisir le classifieur h_t, avec la plus basse erreur ε_t .

4. faire la mise à jour des pondérations:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad \text{……..(2.22)}$$

Ou $e_i = 0$ si l'exemple x_i est correctement classifié, $e_i = 1$ si non, et

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \text{…….(2.23)}$$

- Le classifieur fort final est:

$$H(x) = \begin{cases} 1 & si \quad \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & si\ non \end{cases} \text{……(2.24)}$$

Où $\alpha_t = \log \frac{1}{\beta_t}$ ……....(2.25)

Pour comprendre cet algorithme fondamental, les principales étapes sont détaillées dans le paragraphe 2.7.

2.6.2. Architecture de cascade

L'architecture de cascade est très efficace, l'image d'entrée est passée par une séquence de classifieurs. Pour qu'une image soit classifiée en tant que visage, elle doit passer par tous les classifieurs. La classification s'arrête immédiatement si un quelconque de ces étapes échoue.

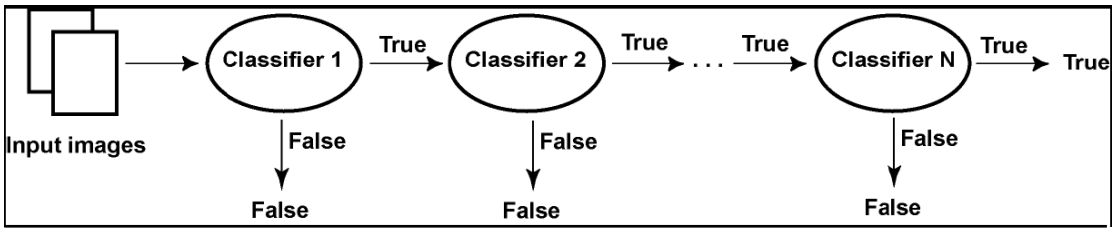


Figure 2.2 : Architecture de cascade.

La décision « fausse » stoppe le calcul et impose le détecteur de retourner « faux ». Une décision « vraie » passe l'image d'entrée au prochain classifieur dans la cascade. Si tous les classifieurs votent vraie alors l'entrée est classifiée comme un vrai exemple. Si n'importe quel classifieur vote « faux » alors le calcul se stoppe et l'entrée sera classifiée comme fausse.

La structure en cascade a trois paramètres principaux qu'il faut déterminer :

K : Le nombre total des classifieurs.

n_i : Nombre de caractéristiques pour chaque étape i.

θ_i : Seuil pour chaque étape i.

Pour une cascade, le taux total de faux positifs est le produit du taux de faux positifs de chaque étape : $F = \prod_{i=1}^{K} f_i$ ……(2.26)

Et de même pour le taux total des détections correctes : $D = \prod_{i=1}^{K} d_i$ …..(2.27)

Par exemple, si nous voulons atteindre un taux de détection de 90%, nous pouvons construire 10 étape de classifieur dans laquelle chaque étape a un taux de détection de 0,99 ($d_i = 0.99$). En effet, $0.9 \approx 0.99^{10}$. Si chacune de ces étapes rejette 70% de négatifs (c-à-d un taux de faux positifs de 30%), le taux total de faux positif est $0.30^{10} \approx 6 \cdot 10^{-6}$.

1. Entrée : Définition des cibles de l'apprentissage

f : Le taux de faux positifs maximal acceptable par étape.

d : Le taux de détection minimal acceptable par étape.

F_{traget} : Le taux de faux positifs souhaités à la fin du processus.

P : L'ensemble des exemples positifs

N : L'ensemble des exemples négatifs

2. Initialisation :

$F_0 = D_0 = 1$

$i = 0$: Le numéro de la courante étape.

3. Boucle principale:

Tant que $F_i > F_{traget}$

$i \leftarrow i + 1$

$n_i = 0$: Le nombre de caractéristiques rectangulaires.

$F_i = F_{i-1}$

- Tant que $F_i > f \cdot F_{i-1}$

  * $n_i \leftarrow n_i + 1$

  * Entraîner le classifieur avec n_i caractéristiques en utilisant AdaBoost avec P et N de la base d'apprentissage.

  * Calculer F_i et D_i pour le courant classifieur avec l'ensemble de validation (Cet ensemble est utilisé pour tester les performances de la cascade, Certaines images sont présentées à la fin de cascade et permettent d'évaluer la qualité globale du détecteur).

  *Réduire le seuil du classifieur jusqu'à ce que le taux de détection

pour le i-ème classifieur est moins de $d \cdot D_{i-1}$.

- Vider l'ensemble d'apprentissage négatif : $N \leftarrow \phi$.
- Si $F_i > F_{traget}$, évaluer la classifieur en cascade sur une série d'exemples négatifs (images des non visages) et de mettre de chaque fausses détections dans la série N.

Le taux maximum de faux positifs « f » pour une étape, le taux minimum de détection « d » pour une étape et 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 AdaBoost 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 d'exemples négatifs et on ne garde que les exemples mal classifiés pendant la dernière étape.

2.7 AdaBoost pas à pas

AdaBoost est un algorithme qui choisit un classifieur faible à chaque étape. L'algorithme reçoit comme entrée la base de données d'apprentissage $S = \{(x_1, y_1),...,(x_N, y_N)\}$ et le paramètre T qui indique combien d'hypothèses doivent être combinées au maximum pour construire l'hypothèse combinée plus puissante $H(t)$. Pour la première itération, les poids $w_{t,i} = \frac{1}{2m}; \frac{1}{2l}$ pour $Y_i = 1; -1$ respectivement.

Ou m indique le nombre des exemples positives utilisés et l le nombre des exemples négatifs.

À chaque itération t, tous les classifieurs faibles $h_t^j \in H$ sont entraînés, H est donc la classe des classifieurs faibles.

Maintenant discutons d'un classifieur faible pour l'apprentissage des classifieurs binaires. Le principe de boosting est de combiner les classificateurs faibles qui s'appellent aussi les faibles apprenants.

Chaque classifieur faible a une caractéristique rectangulaire f_j, un seuil θ_j, et une parité P_j qui indique la direction du signe d'inégalité de l'hypothèse simple suivante :

$$h_t^j(x) = \begin{cases} 1 & si \quad p_j f_j(x) < p_j \theta_j \\ -1 & si \quad non \end{cases} \quad \text{.....(2.28)}$$

La façon de choisir le seuil et la parité des classifieurs faibles h_t^j est décrite dans le chapitre 4. Une fois les classifieurs faibles h_t^j entraînés, l'erreur d'entraînement ε_t^j pour chaque classifieur faible h_t^j est calculée :

$$\varepsilon_t^j = \sum_{i=1}^{n} w_{t,i}.I(x_i, y_i) \text{.....(2.29)}$$

Avec :

$$I(x_i, y_i) = \begin{cases} 1 & pour \quad y_i \neq h_t(x_i) \\ 0 & \sin on \end{cases} \text{.....(2.30)}$$

Parmi tous les classifieurs faibles entraînés, AdaBoost choisit maintenant le meilleur. Il s'agit du classifieur qui possède l'erreur d'apprentissage minimale.

$$\begin{cases} h_t^j \rightarrow h_t \\ \varepsilon_t^j \rightarrow \varepsilon_t \end{cases} \text{ si } \varepsilon_t^j < \varepsilon_t^k \text{ pour tout } k \neq j$$

Après avoir choisi le meilleur classifieur h_t , AdaBoost calcule le poids α_t à attribuer au classifieur h_t. Le poids α_t est calculé de façon à minimiser la fonction de coût

$$G^{AB}(\alpha) = \sum_{i=1}^{n} \exp\{-y_i(\alpha h_t(x_i) + f_{t-1}(x_i))\} \text{.....(2.31)}$$

avec

$$f_{t-1}(x_i) = \sum_{r=1}^{t-1} \alpha_r h_r(x_i) \text{.....(2.32)}$$

Analytiquement le α_t peut être calculé de la manière suivante :

$\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ ……(2.33), (**[31]**, **[32]**, **[33])**.

Et celui qui a été utilisée dans notre application.

Pour finir, AdaBoost actualise la pondération des données d'apprentissage pour ce concentrer sur des exemples difficiles à classifier, pour notre application on a utilisé la fonction de pondération suivante :

$w_{t+1,i} = w_{t,i}\exp(-1\cdot(y_i\cdot\alpha_t\cdot h_i))$ ……(2.34) , (**[32]**, **[33])**.

Pour les visages $y_i = 1$, pour les non visages $y_i = -1$.

Si x_i est classé comme un visage $h_i = 1$ si non $h_i = -1$.

Les exemples classifiés correctement reçoivent donc une pondération plus petite et celles classifiées incorrectement reçoivent une pondération plus grande.

L'itération est interrompue, lorsque l'erreur d'entraînement ε_t du classifieur h_t est égale à 0 ou supérieur à 0,5. Si $\varepsilon_t = 0$, la classification est optimale et il n'est plus nécessaire d'ajouter d'autres classifieurs. Si $\varepsilon_t \geq 0.5$, alors le classifieur h_t ne satisfait plus les conditions de classifieur faible. Ceci signifie que h_t classifie moins bien qu'une sélection aléatoire, et l'hypothèse f_T ne peut donc plus être améliorée.

L'hypothèse composée f_T est la combinaison linéaire des hypothèses faibles h_t donnée par :

$f_T(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$ ……..(2.35)

L'hypothèse finale $H(x)$ est donnée par :

$H(x) = sign[f_T(x)]$ ……..(2.36)

$(+1)"Objet": \quad si \quad f_T(x) \geq \frac{1}{2}\sum_{t=1}^{T}\alpha_t$ …….(2.37)

$(-1)"Non\_objet": \quad si \quad f_T(x) < \frac{1}{2}\sum_{t=1}^{T}\alpha_t$ ……(2.38)

L’algorithme AdaBoost est un algorithme plus sophistiqué pour le boosting de la combinaison des hypothèses. Il est appelée Adaptive dans le sens que les exemples qui sont mal classifiés doivent obtenir les poids les plus élevés dans la prochaine itération, par exemple, les exemples près de la frontière de décision sont plus difficiles à classer et à cet effet obtenir des poids élevés dans l’ensemble d’entrée après la première itérations.

2.8 Conclusion

Nous avons étudié dans ce chapitre, l'algorithme Adaboost et ces versions, il se caractérise par sa simplicité et sa souplesse. Différent chercheurs ont essayé de l'adapter à des problématiques de l’analyse de visage donnant lieu à des résultats considérables au moins du point de vue expérimental.

L'objectif était donc de fournir un point de départ à l'utilisation d'Adaboost. Les performances de cet algorithme ainsi que les justifications expérimentales dont il jouit, en font un bon candidat.

Chapitre III

3.1. Introduction

La couleur de la peau humaine est un descripteur efficace utilisé pour la détection, localisation et suivi des cibles contenant la peau comme les visages et les mains. Il est souvent difficile de séparer les objets de la peau de non peau comme des objets en bois, qui peuvent ressembler à des couleurs de peau. Par conséquent, la peau est souvent combinée avec d'autres facteurs comme le mouvement, la texture et les caractéristiques de contours, mais seulement dans ce chapitre le traitement de la couleur est étudié. L'objectif est de diviser les pixels de l'image de couleur de peau et non peau.

Il existe de nombreuses possibilités d'attributs pour caractériser la couleur de peau, l'histogramme, les moments, la couleur…etc. Avec la même méthode, si on change l'espace de couleur, il peut donner des informations différentes de l'image. Ce chapitre se consacre aux méthodes de segmentation de couleur de peau les plus utilisées dans un système d'indexation et recherche des peaux humaines ainsi que sur les espaces de couleur.

Ce chapitre est une phase préliminaire pour notre travail, on doit donc choisir le meilleur espace de couleur et la meilleure technique de segmentation qui représente le mieux la couleur de peau.

3.2. Techniques de détections de peau

Terrillon et autres **[34]** ont comparé la performance des neuf espaces de chrominance dans le contexte de la détection de visage. Les espaces de couleur ont été évalués en utilisant deux métriques. La première était la déviation de carré moyenne (MSDN) qui a été calculée de l'histogramme normalisé construit pour chaque espace de couleur. La seconde métrique était le degré de chevauchement entre les distributions référées de peau et non peau appelé HIN. L'étude a conclu que la discrimination entre la classe de peau et la classe de non-peau est la plus haute dans les espaces normalisés.

Albiol et autres **[35]** ont réclamé que pour chaque espace de couleur il y a un détecteur optimum de peau avec performance comparable. Un détecteur de peau peut être trouvé avec la même performance indépendamment de l'espace de couleur

choisi, à condition qu’il y ait une transformation inversible entre les espaces de couleur comparés. Trois espaces de couleur RGB, YCbCr, et HSV ont été comparés. L’étude a conclu que les caractéristiques de fonctionnement des espaces de couleur tridimensionnels étaient identiques, et la performance de l'espace de couleur CbCr était inférieure puisque la transformation de l’espace de couleur tri-dimensionnel à l'espace de couleur bi-dimensionnel n'est pas inversible.

3.3. Perception de la couleur pour l’oeil humain

L'environnement a une forte influence sur notre vision. Notre œil ne voit pas la même couleur toujours de la même manière. Notre perception des couleurs est toujours faussée, parce que lorsque l'on regarde un objet, l'œil a tendance à mesurer, à comprendre, à évaluer sa couleur en fonction de la scène qui l'entoure. Tout est affaire de contrastes. Rappelons enfin que la couleur n'est pas une caractéristique propre d'un objet. Elle dépend de la qualité de la lumière qui l'éclaire. L'objet ne réagira pas de la même manière à la lumière du soleil et à celle diffusée par un néon.

Quand nous regardons, le cristallin de l'oeil humain projette une image optique sur la rétine. La lumière est alors transformée en signaux électriques. Ces signaux passent par le nerf optique pour arriver au cerveau. C'est alors que l'image est perçue par l'individu. La rétine comporte trois types de cellules sensibles à la couleur, les cônes. Chaque type de cônes est sensible à une des couleurs primaires (rouge, vert, bleu). La couleur est une partie importante de la perception visuelle de notre environnement. Cependant la couleur est une notion crée par l'homme, il n'y a pas de couleur dans la nature sans la vision humaine.

3.4. Perception de la couleur pour la machine

Une gamme très large de couleurs peut être visuellement assortie par un mélange des trois lumières primaires. Ceci permet à des couleurs d'être reproduites par un mélange des lumières rouges, vertes, et bleues (les couleurs primaires de l'espace de couleur additif sont représenté sur la figure 3.1, les couleurs primaires de l'espace de couleur soustractive sont représenté sur la figure 32).

a. Couleurs additives

Il s'agit d'un procédé par ajout de lumière pour reproduire les différentes couleurs. Le procédé est principalement utilisé dans les domaines du traitement électronique de l'image tels que télévision, écran d'ordinateur, vidéo et vidéo projection. Le masque perforé des écrans comporte des éléments phosphorescents qui s'éclairent en rouge, vert et bleu pour constituer l'image en couleurs. L'ensemble des éléments phosphorescents produit la sensation de couleur, par synthèse additive. L'oeil ne perçoit pas les points audiovisuels si le diamètre est inférieur au seuil de perception (inférieur à 0.3 mm).

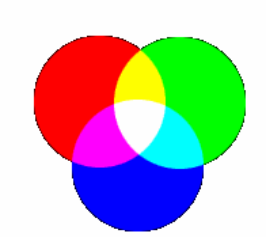


Figure 3.1: Espace additif

b. Couleurs soustractives

Cette méthode est utilisée dans l'imprimerie couleur, la photographie et les diapositives. Ce procédé repose sur un système de retrait de lumière par utilisation de filtres et de pigments. Ces procédés de reproduction des couleurs utilisent les couleurs primaires soustractives qui sont le cyan, le magenta et le jaune.

Les procédés soustractifs trichromes sont les couleurs complémentaires des couleurs de base de la synthèse additive.

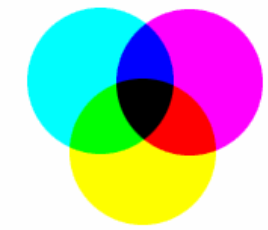


Figure 3.2: Espace soustractif

La théorie tri-chromatique décrit la manière comment les trois lumières séparées rouge, vert et bleu, peuvent assortir n'importe quelle couleur visible basée sur l'utilisation de l'oeil des trois couleurs sensitives. C'est la base sur laquelle la photographie et l'impression fonctionnent, en utilisant trois colorants différents pour reproduire la couleur dans une scène. C'est également la manière dont la plupart des espaces de couleur d'ordinateur actionnent, en utilisant trois paramètres pour définir une couleur.

Un ordinateur donc peut décrire une couleur en utilisant les paramètres exigés pour assortir une couleur. Une couleur est habituellement spécifiée en utilisant trois paramètres. Ces paramètres décrivent la position de la couleur dans l'espace de couleur étant utilisé. Ils ne nous indiquent pas que ce qu'est la couleur, cela dépend de quel espace de couleur est utilisé, cet espace de couleur décrit un environnement dans lequel des couleurs sont représentées, comparées, ou calculées. Il peut définir un environnement sur un, deux, trois ou quatre dimensions dont les composants de couleur représentent les valeurs d'intensité.

3.5. Espaces de couleur

La première étape pour détecter la peau est de choisir un espace de couleur adéquat, le choix d'un espace particulier dépend souvent de la méthode envisagée et du résultat recherché. L'espace de couleur étant souvent utilisé pour coder les images, l'utilisation d'un espace particulier par une méthode nécessite d'utiliser une transformation T permettant de passer de l'espace à l'espace désiré et une transformation inverse T^{-1} de façon à stocker l'image résultat, l’espace choisi doit produire une robustesse aux variations de l’illumination. Cela est réalisé si l'espace de couleur sépare efficacement la chrominance de la luminance.

Nous passerons brièvement sur les espaces de couleur les plus populaires et leurs propriétés.

3.5.1 Espace RGB

C'est l'espace de base, supporté nativement par la plupart des cartes vidéo. En combinant les trois primitives rouge, vert et bleu, on peut obtenir (presque) toutes

les couleurs visibles. Le système de couleur RGB est un système de couleur additif. Quand on mélange des pigments de peinture pour obtenir des couleurs précises, on filtre la lumière, c'est-à-dire qu'on en retire des composantes, ne réfléchissant que les couleurs désirées. C'est ce que l'on nomme un système soustractif. Le système RGB est additif car on ajoute de la lumière. Dosées, les trois composantes permettent de représenter un grand nombre de couleurs, mais pas toutes les couleurs visibles par l'œil. Il est l'un des espaces couleurs le plus largement répandus pour le traitement et le stockage des données d'image numérique.

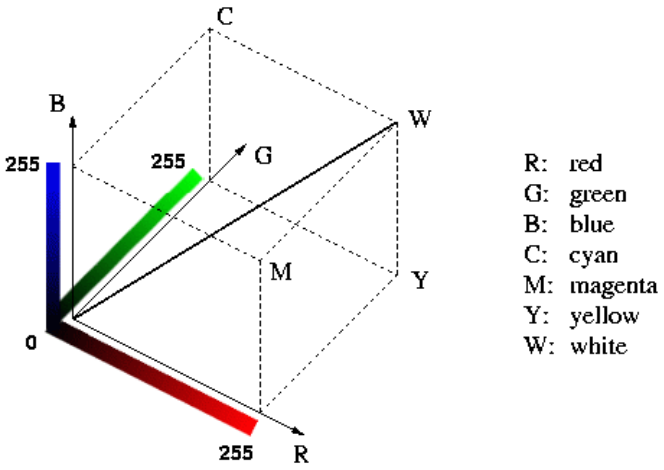


Figure 3.3: Le Cube de Couleurs RGB

3.5.2 Espace RGB normalisé

L’espace RGB Normalisé est formé indépendamment des différents niveaux d'éclairage. Le rouge, vert et bleu de l'espace RGB normalisé peut être obtenu d’après les trois composantes de l’espace RGB grâce à la formule suivante:

$$r = \frac{R}{R+G+B} \; ;$$

$$g = \frac{G}{R+G+B} \; ;$$

$$b = \frac{B}{R+G+B} \; .$$

Comme la somme des trois composantes normalisés est connue ($r+g+b=1$), le troisième composante ne tient aucune information significative et peut être omis, ce qui engendre une réduction de la dimensionnalité de l'espace. Les composantes restantes s'appellent souvent "les couleurs pures", pour la dépendance de *r* et *g* sur l'éclat de la couleur de la source RGB est diminué par la normalisation. Une propriété remarquable de cette représentation est celle pour les surfaces mates, tout en ignorant la lumière ambiante, le RGB normalisé est invariable (dans certaines prétentions) aux changements de l'orientation extérieure relativement à la source lumineuse, ceci ainsi que la simplicité de la transformation a aidé cet espace de couleur pour gagner la popularité auprès les chercheurs.

3.5.3 Espace HSV (Hue Saturation Value)

Cet espace de couleur est le plus intuitif. Les couleurs, plutôt que d'être décomposées en primitives, comme RGB, par exemple, sont représentées selon des notions évidentes: la luminosité, la teinte et la pureté.

La luminosité (value dans HS**V**) correspond à la brillance perçue de la couleur.

La teinte (*hue* dans **H**SV) correspond à notre notion de la petite école, une roue de couleur où se trouvent six couleurs primitives: rouge, jaune, vert, cyan, bleu et magenta. La teinte donne la position sur ce cercle de couleur:

La teinte est encodée comme la position, un peu comme l'heure sur un cadran, sur le cercle des couleurs.

La pureté, ou saturation (H**S**V), correspond à la distance au blanc. Au centre du cône, les couleurs sont des tons de gris; plus on s'éloigne du centre, plus les couleurs se démarquent et plus elles sont *pures*. Les couleurs sont saturées, ou maximalement pures, sur la surface du cône.

L'espace de couleur forme un cône, avec le bout pointu dans les couleurs sombres, et le sommet le noir. La logique derrière cette forme est qu'une couleur ne peut pas être en même temps très foncé et très saturée; on ne peut pas avoir un rouge éclatant et très foncé à la fois. Les couleurs les plus brillantes se retrouvent sur la base du cône.

Les espaces de couleur HSV ont été présentés quand il y avait un besoin de l'utilisateur d'indiquer des propriétés de couleur numériquement. Ils décrivent la couleur avec des valeurs intuitives. L'intuitivité des composants de l'espace de couleur et de la discrimination explicite entre la luminance et les propriétés de chrominance a rendu ces espaces de couleurs populaires dans les travaux sur la segmentation de la couleur de peau.

Plusieurs propriétés intéressantes de la teinte (Hue en anglais) ont été notées, il est invariable aux points culminants aux sources lumineuses blanches, et aussi, pour les surfaces mates, à la lumière ambiante et l'orientation de la surface relativement à la source lumineuse. Cependant, Poynton **[36]**, précise plusieurs caractéristiques indésirables de ces espace de couleur, y compris des discontinuités de Hue et le calcul du l'"éclat" (value), qui est en mal conflit avec les propriétés de la vision de couleur.

$$H = \arccos \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)(G-B))}}$$

$$S = 1 - 3\frac{\min(R,G,B)}{R+G+B}$$

$$V = \frac{1}{3}(R+G+B)$$

Une manière alternative du calcul de la teinte et de saturation en utilisant des valeurs logarithmiques a été présentée par Fleck **[37]**, où la transformation logarithmique additionnelle des valeurs de RGB est visé à réduire la dépendance de la chrominance au niveau d'illumination.

Le système de coordination polaire de l'espace Hue-saturation résultant de la nature cyclique de l'espace de couleur le rend incommode pour les modèles paramétriques de couleur de peau qui ont besoin du faisceau serré de couleurs de peau pour une meilleure performance. Une représentation différente de HS en utilisant les coordonnées cartésiennes est donnée par:

$X = S\cos H$; $Y = S\sin H$.

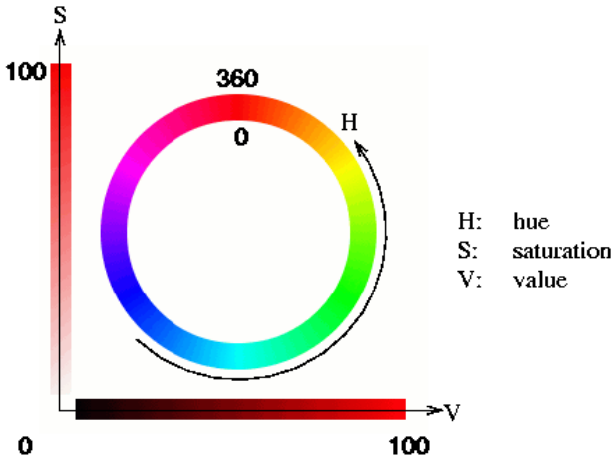


Figure 3.4: Teinte, Saturation, et Valeur

3.5.4 Espace YCbCr

L'espace de couleur YCbCr a été défini en réponse aux demandes croissantes des algorithmes numériques dans la manipulation de l'information video, et il est devenu un modèle largement répandu dans la vidéo numérique. Il appartient à la famille des espaces de couleur de la transmission de télévision, généralement utilisé par les studios de télévision et pour le travail de compression d'image.

Les algorithmes de compression JPEG et MPEG utilisent cette transformée de couleur comme étape préalable à la décimation. En concentrant l'information utile dans Y, la luminosité, on peut discrétiser les deux autres coefficients, C_r et C_b sans avoir trop de perte perceptible.

La transformée est donnée par

$$\begin{pmatrix} +0.299 & +0.587 & +0.114 \\ -0.169 & -0.441 & +0.500 \\ +0.500 & -0.418 & -0.081 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix}$$

On peut écrire :

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = B - Y$$

$$Cr = R - Y$$

La simplicité de transformation et la séparation explicite des composants de luminance et de chrominance rend cet espace de couleur attrayant pour la modélisation de couleur de peau.

3.5.5. Espace YPbPr

YPbPr est un espace de couleur utilisé dans la vidéo électronique. YPbPr est la version analogique de l'espace couleur YCbCr, les deux sont numériquement équivalente, mais YPbPr est conçu pour être utilisé dans les systèmes analogiques mais YCbCr est destiné à la vidéo numérique.

YPbPr est convertie de la RGB du signal vidéo, qui est divisé en trois composantes, Y, Pr et Pb.

Y: information de la luminance (luminosité).

Pb: la différence entre le bleu et la luminance (B - Y).

Pr: la différence entre le rouge et la luminance (R - Y).

3.5.6. Espace Ydbdr :

YDbDr est un espace de couleur utilisé dans le SÉCAM (le standard de radiodiffusion de télévision en couleur), qui est utilisé en France et dans certains pays de l'Europe de l'Est. Il est très proche de YUV et de ses espaces de couleur reliés tels que YIQ, YCbCr et YPbPr.

YDbDr est composé de trois composantes Y, Db et Dr, Y est la luminance et Db et Dr sont les composantes de chrominance.

$$\begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.450 & -0.883 & 1.333 \\ -1.333 & 1.116 & 0.217 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} Y \\ db \\ dr \end{pmatrix}$$

3.5.7. Espace XYZ

Cet espace de couleur nous vient de la CIÉ (Commission Internationale de l'Éclairage) et date des années 1930. À cette époque, on se demandait encore comment représenter les couleurs pour la télévision.

Ici aussi la composante Y représente la luminance. Les deux autres composantes, X et Z, sont des composantes auxquelles il n'est pas facile d'associer de notions intuitives.

La transformée est donnée par :

$$\begin{pmatrix} +0.431 & +0.342 & +0.178 \\ -0.222 & +0.707 & +0.071 \\ +0.020 & -0.130 & -0.939 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

3.5.8. Espace Yxy :

Yxy exprime les valeurs de l'espace XYZ en termes des coordonnées chromatiques x et y, quelque peu analogue aux coordonnées de la teinte et la saturation de l'espace HSV. Les coordonnées sont indiquées dans les formules ci-dessous, utilisé pour convertir XYZ en Yxy:

Y = Y.

x = X / (X+Y+Z).

y = Y / (X+Y+Z).

Notez que la valeur de Z est incorporée dans les nouvelles coordonnées et n'apparaît pas par elle-même. Y est toujours liée à la luminosité d'une couleur, les autres aspects de la couleur se retrouvent dans les coordonnées chromatiques x et y.

3.5.9. Espace YIQ

Les télés nord-américaine et japonaise utilisent l'espace YIQ pour encoder les images couleurs. Si le décodeur est une télé noir et blanc, seule la composante Y est utilisée. où Y représente la luminance utilisée seule par les téléviseurs noir et blanc et en synthèse additive par les téléviseurs couleur. I et Q sont respectivement les composantes chromatiques représentant les oppositions cyan-orange et magenta-bleu.

La transformée est donnée par :

$$\begin{pmatrix} +0.299 & +0.587 & +0.114 \\ +0.596 & -0.275 & -0.322 \\ +0.212 & -0.523 & +0.311 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} Y \\ I \\ Q \end{pmatrix}$$

3.5.10 Espace YUV

Cet espace a été introduit par le standard allemand PAL. Cet espace utilise à son tour des composantes R, G et B autres que celles définies par la CIE et par de standard NTSC. La formule de la luminance Y est la même du point de vue écriture que celle définies pour l'espace YIQ. Néanmoins, les valeurs Y_{YUV} sont légèrement différentes des valeurs Y_{YIQ} à cause de l'utilisation de primaires RGB différentes. Les composantes de l'espace YUV s'écrivent comme suit :

$$\begin{cases} Y = 0.299\ R + 0.587\ G + 0.114\ B, \\ U = 0.493(B - Y), \\ V = 0.877(R - Y). \end{cases}$$

De même que pour l'espace YIQ, les composantes peuvent être exprimées sous forme d'une matrice de passage donnée par la formule suivante :

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.5.11. Espace Hunter Lab :

Hunter a développé en 1948 un espace couleur uniforme appelé Hunter Lab, que l'on peut lire directement avec un colorimètre photoélectrique. Il est calculé à partir des racines carrées de XYZ.

L = 10 * sqrt(Y).

a = 17.5 * (((1.02 * X) - Y) ./ sqrt(Y)).

b = 7 * ((Y - (0.847 * Z)) ./ sqrt(Y)).

3.6. Segmentation de la couleur de peau

La peau humaine a les propriétés de couleur uniques. L'idée d'utiliser la couleur de peau dans l'étape de prétraitement de la détection de visage a été suggérée parce qu'elle est invariable contre la graduation, les rotations, la translation. Les deux défis de base décident quel espace de couleur doit être utilisé, et comment modeler la distribution de couleur de peau. Des méthodes de détection de peau sont classées par catégorie dans deux catégories, technique de seuillage simples et la modélisation de couleur de peau à partir d'une base choisie.

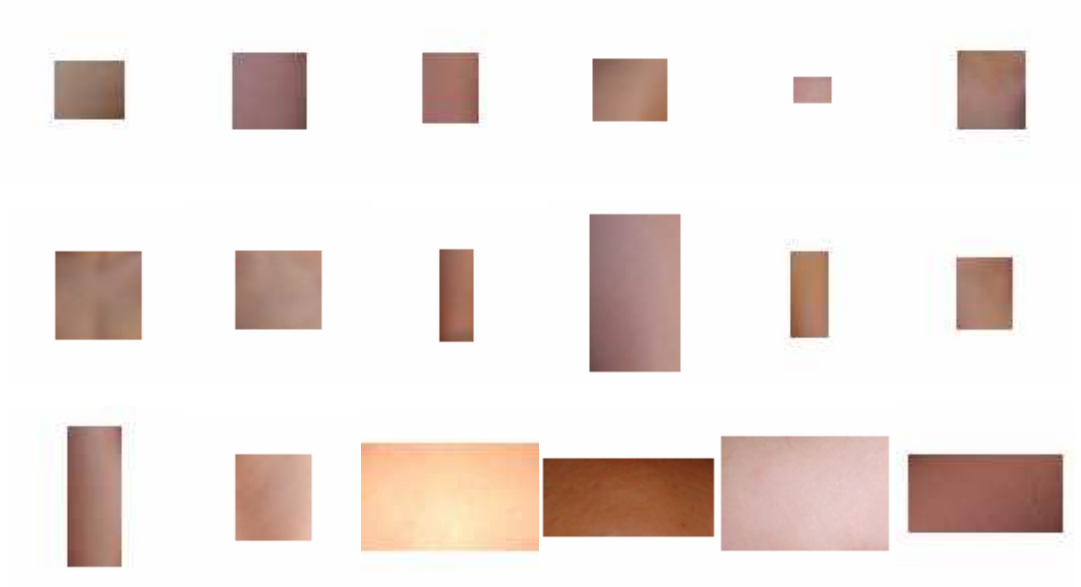


Figure 3.5 : Quelques échantillons de la base de peau

3.7. Techniques de segmentation de la couleur de peau

La couleur de peau peut être segmentée de deux manière : par seuillage ou par modèle de peau, figure 3.6.

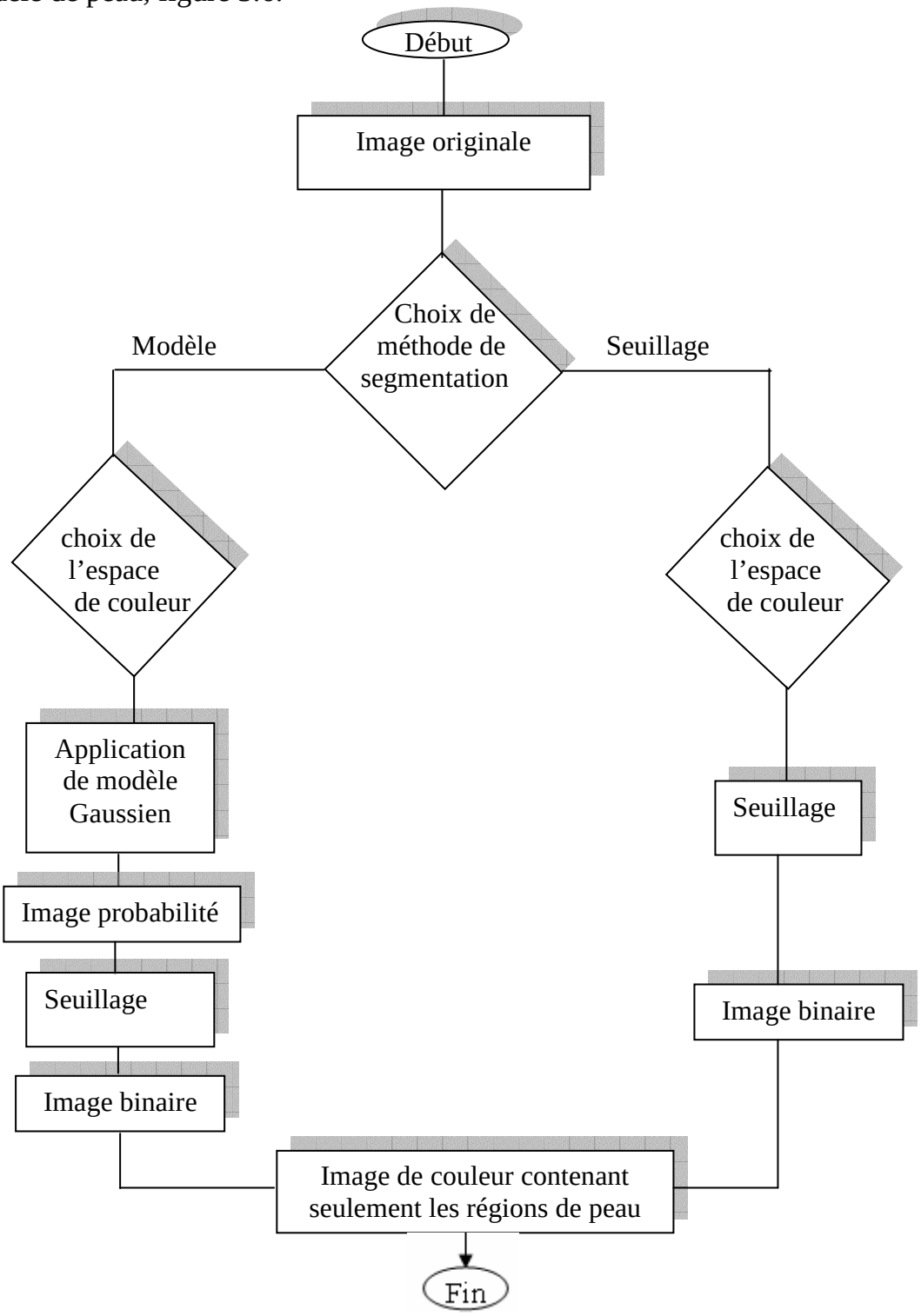


Figure 3.6 : Etapes de la détection de peau

3.7.1 Segmentation de peau par seuillage

La détection de peau par seuillage consiste à classifier chaque Pixel indépendamment de ses voisins. Un détecteur de couleur de peau peut être défini en termes de la relation entre la chrominance et les composants d'intensité, la teinte, les composants de saturation, ou les composants rouges, verts et bleus. Ceci change selon l'espace de couleur utilisé.

La méthode sépare le couleur de peau de la couleur non peau, cette méthode propose un ensemble de seuils fixés de peau dans un espace de couleur donné, quelques espaces de couleur permettent de rechercher des Pixel de couleur de peau dans l'espace chromatique 2D, réduisant la dépendance à légard de la variation d'éclairage, d'autres, tel que l'espace RGB, adressent le problème d'éclairage en présentant différentes règles selon les conditions d'illumination (jour uniforme, ou le flash).

Travaillant dans les différents espaces de couleur, nous avons mis en application onze algorithmes différents analysés en ce chapitre. Ils sont nommés par l'espace de couleur adopté : YCbCr, HSV1, HSV2, RGB, RGBn, YUV, Ydbdr, Ypbpr, Hlab, YIQ, Yxy.

3.7.1.1. YCbCr

Chai et Ngan **[38]** développent un algorithme qui exploite les caractéristiques spatiales de distribution de couleur humaine de peau. Une carte couleur de peau est dérivée et utilisée sur les composants de chrominance de l'image d'entrée pour détecter les Pixels qui semblent être peau. L'algorithme utilise alors un ensemble de procédés de régularisation pour renforcer ces régions des Pixel de couleur de peau qui peuvent probablement appartenir aux régions faciales. Nous utilisons seulement leur étape de segmentation de couleur. Travaillant dans l'espace de YCbCr, les auteurs constatent que les gammes de Cb et Cr les plus représentatives pour la carte de référence de couleur de peau étaient :

$77 \le Cb \le 127$ et $133 \le Cr \le 173$

3.7.1.2. RGB :

Kovac et Al **[39]**, travaillent dans l'espace de couleur de RGB, ils classifient la couleur de peau par des règles heuristiques qui tiennent compte deux conditions différentes : jour uniforme et flash ou illumination latérale.

Illumination uniforme du jour :

$R > 95,\ G > 40,\ B > 20$

$\max\{R,G,B\} - \min\{R,G,B\} < 15$

$|R - G| > 15,\ R > G,\ R > B$

Le flash ou l'illumination latérale du jour :

$R > 220,\ G > 210,\ B > 170$

$|R - G| \le 15,\ B < R,\ B < G$

3.7.1.3. HSV

Tsekeridou et Pitas **[40]** travaillent dans l'espace de couleur HSV et choisissent les Pixels ayant les couleurs de la peau en plaçant les seuils suivants :

Pour le HSV1 (Transformation par la fonction de Matlab « rgb2hsv ») :

$V \ge 40,$

$0.2 < S < 0.6,$

$0^\circ < H < 25^\circ$ ou $335^\circ < H < 360^\circ$

La gamme choisie de H limite des couleurs rougeâtres de la segmentation et la gamme de saturation choisie assure l'exclusion de couleurs rouges pures et rouge très foncé, tous les deux sont causés par de petites variations dans le conditionnement de l'éclairage. Le seuil sur V est présenté pour jeter les couleurs foncées.

Pour le HSV2 (Transformation par l'application des valeurs de « HSV » décrites au paragraphe 3.5.3) et après plusieurs essais sur cette espace de couleur on a abouti à fixer les seuils suivants :

$0.10 < S < 30$
$0.03 < H < 0.07$
$V > 0.4$

3.7.1.4. RGBn

Gomez et Morales **[41]** utilisent une approche d'induction constructive pour déterminer la carte de peau. Commençant par les trois composants de RGB dans une forme normalisée et un ensemble simple d'opérateurs arithmétiques, les auteurs produisent un modèle pour la détection de peau. Parmi les différentes règles de combinaison présentées par les auteurs, nous avons choisi celui avec la précision et le taux de succès les plus élevés

$$\frac{r}{g} > 1.185;$$

$$\frac{r \cdot b}{(r+g+b)^2} > 0.107;$$

$$\frac{r \cdot g}{(r+g+b)^2} > 0.112.$$

Ou r, g, b sont les coordonnées normalisés obtenus par :

$$r = \frac{R}{R+G+B};$$

$$g = \frac{G}{R+G+B};$$

$$b = \frac{B}{R+G+B}.$$

N’ayant pas trouvé dans la littérature scientifique, des seuils de segmentation de la couleur de peau pour les espaces de couleurs restants nous avons donc proposé les nôtres obtenus après plusieurs tests.

3.7.1.5. YUV

Les seuils sont fixés comme suit :

$80 < Y < 180$
$20 < U < 100$
$100 < V < 160$

3.7.1.6. Ydbdr

Les seuils sont fixés comme suit :

125<Y<200

105< db<240

180< dr<240

3.6.1.7. Ypbpr

Les seuils sont fixés comme suit :

120<Y<236

33<pb<107

75<pr<130

3.7.1.8. Yiq

Les seuils sont fixés comme suit :

110<Y<242

80<i<153

53<q<122

3.7.1.9. Yxy

Les seuils sont fixés comme suit :

115<Y<240

x=1, et y=1.

3.7.1.10. Hunter lab

Les seuils sont fixés comme suit :

$0 \leq a < 0.55$ et b > 0.99.

3.7.2. Segmentation de peau par modèle de peau

L'inspiration pour utiliser l'analyse de couleur de peau pour la classification initiale d'une image dans des régions probables de visages et de non-visages provient d'un certain nombre de caractéristiques simples mais puissantes de couleur de peau.

La détection de peau peut utiliser des modèles paramétriques ou non paramétriques de distribution de peau. En cas de modélisation paramétrique, un modèle statistique prédéfini est choisi pour modéliser la distribution de peau.

Dans le cas de modélisation non paramétrique, une analyse d'histogramme des données d'apprentissage suit une fonction de probabilité de peau décrivant la distribution de la couleur de peau.

On a utilisé l'approche paramétrique pour se faire on suit les étapes suivants :

1- modifier l'image originale pour les différents espaces dans laquelle nous avons l'intention de détecter des peaux.

2- créer un modèle de peau pour chaque espace de couleur. Pour créer le modèle de peau, il était nécessaire d'utiliser plusieurs images avec des personnes de couleurs de peau variables. Nous avons utilisé 200 images différentes des personnes avec différentes couleurs de peau.

3- De ces images nous avons extrait des imagettes ne contenant que des sections peau qui sont par la suite représentée dans l'espace de couleur choisit.

4- Chacun de ces échantillons était alors filtré par un filtre passe bas afin de réduire le bruit dans chaque image. La réponse impulsionnelle du filtre passe-bas est donnée par:

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$$

5- Après les moyennes r et b et la covariance C sont déterminés dans chaque espace de couleur utilisé:

$moyenne : m = E\{x\}$; Où $x = (r, b)^T$ est le vecteur transposé de coordination.

r est la moyenne de la première coordonnée représentative, et b la deuxième.

$\text{cov}\,ariance : C = E\{(x - m)(x - m)^T\}$

6- En utilisant les valeurs de la moyenne et de covariance, le modèle de couleur de peau peut être adapté dans un modèle gaussien par $N(m, c)$

Maintenant on a le modèle de peau et on peut commencer le procédé de détection de peau.

7- Après création du modèle de peau, on détermine les régions qui sont les plus

susceptibles

d’êtres des régions de peau dans l'image originale. En utilisant le modèle de distribution de peau, nous utilisons alors l'algorithme suivant pour déterminer les régions probables de peau :

$$P(r,b) = \exp\left[-0.5(x-m)^T C^{-1}(x-m)\right]$$

r, b sont par notation les moyennes des coordonnées les plus représentatives approprié à chaque espace de couleur.

C est la covariance.

Cet algorithme détermine la probabilité qu'un Pixel étant une région de peau en se basant sur le modèle de peau déjà crée.

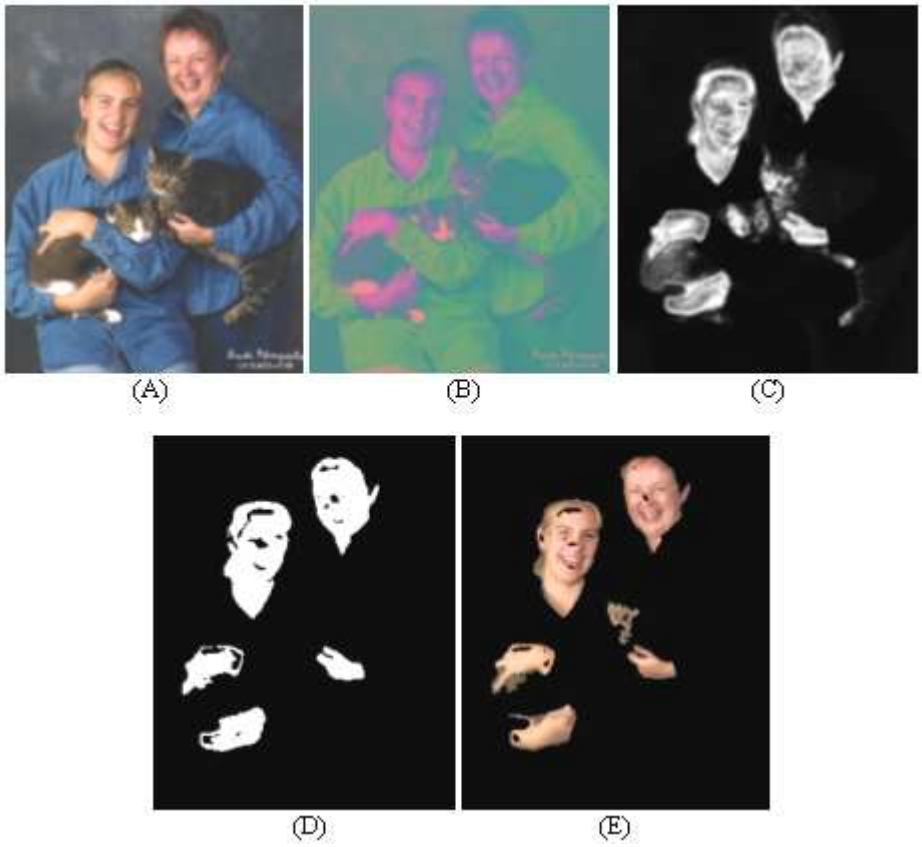
8- Après la détermination de la probabilité de chaque Pixel de l'image, nous obtenons une image en niveau de gris donnant la probabilité basée sur les niveaux de gris.

9- Après avoir obtenu une image de probabilité de peau en niveau de gris, le seuillage de l’image dans une image binaire est alors nécessaire. Puisque les couleurs de peau changent entre chaque personne, un processus adaptatif de seuil est nécessaire.

Si la valeur de seuil est trop basse, la quantité des régions de peau segmentées augmente. Basant sur le fait qu'une valeur de seuil diminue et qu'elle est trop basse, l'augmentation du nombre des régions de peau saute brusquement. Une fois que la valeur de seuil optimale est déterminée, l'image en niveau de gris est alors convertie en binaire.

(A) : Image originale.
(B) : Image convertie à l'espace Hlab.
(C) : Image probabilité.
(D) : Image binaire.
(E) : Image de peau.

Figure 3.7 : Etapes de détection de la peau par Modèle avec l'espace couleur Hunter lab



(A) : Image originale.
(B) : Image convertie à l'espace Ycbcr.
(C) : Image probabilité.
(D) : Image binaire.
(E) : Image de peau.

Figure 3.8 : Etapes de détection de la peau par Modèle avec l'espace couleur YCbCr

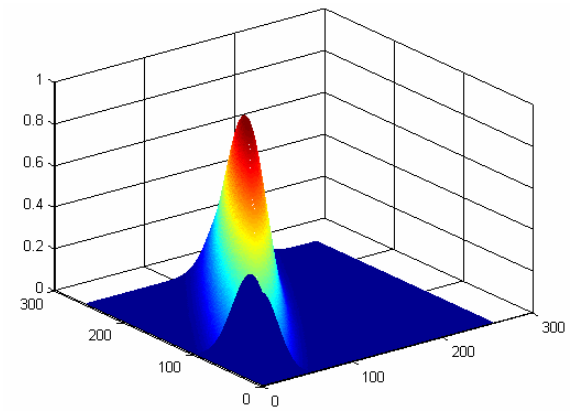


Figure 3.9: distribution de la couleur de peau pour l'espace RGB

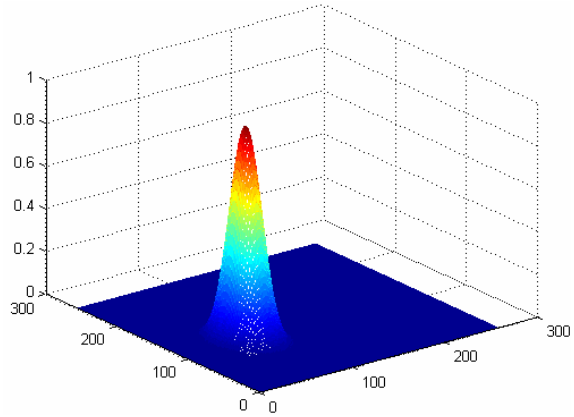


Figure 3.10: distribution de la couleur de peau pour l'espace RGBn

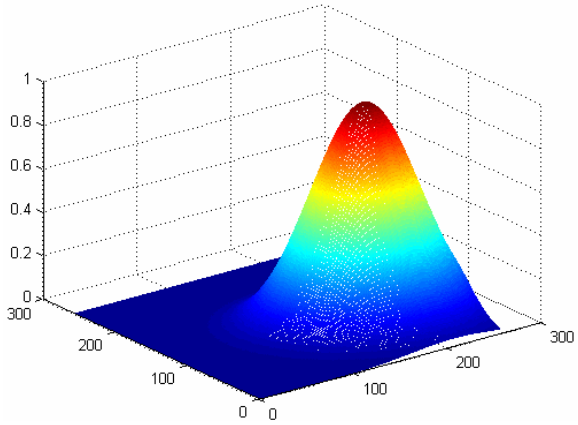


Figure 3.11: distribution de la couleur de peau pour l'espace HSV

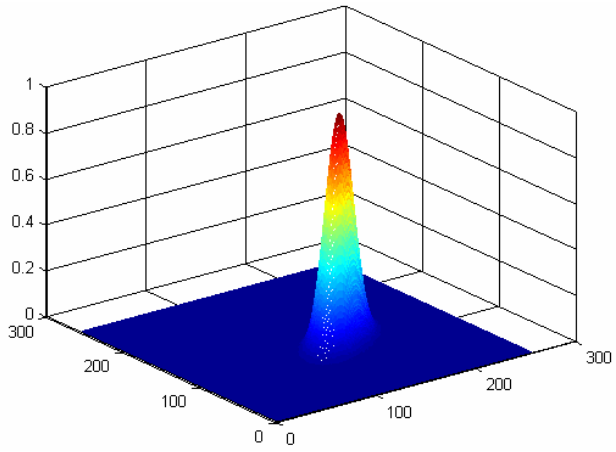


Figure 3.12: distribution de la couleur de peau pour l'espace YCbCr

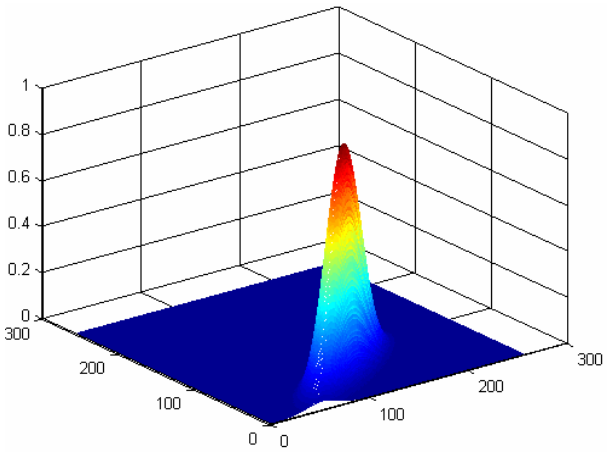


Figure 3.13: distribution de la couleur de peau pour l'espace YUV

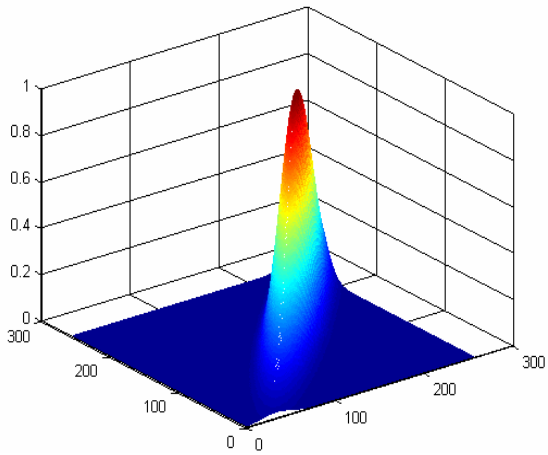


Figure 3.14: distribution de la couleur de peau pour l'espace Ydbd

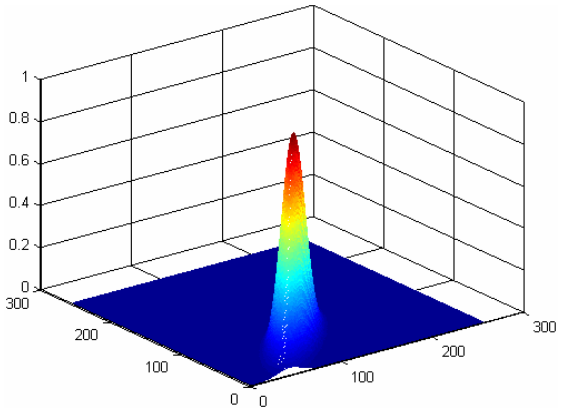


Figure 3.15: distribution de la couleur de peau pour l'espace Ypbpr

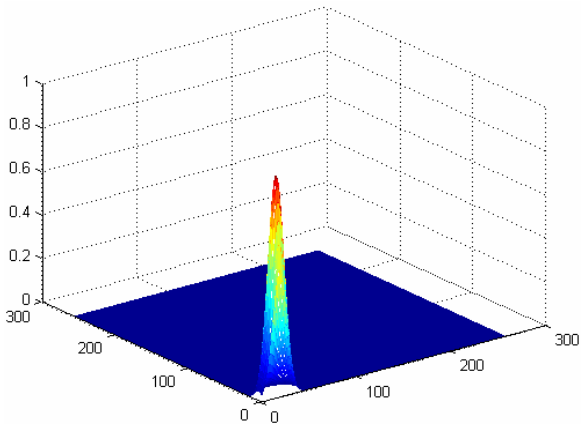


Figure 3.16: distribution de la couleur de peau pour l'espace HLab

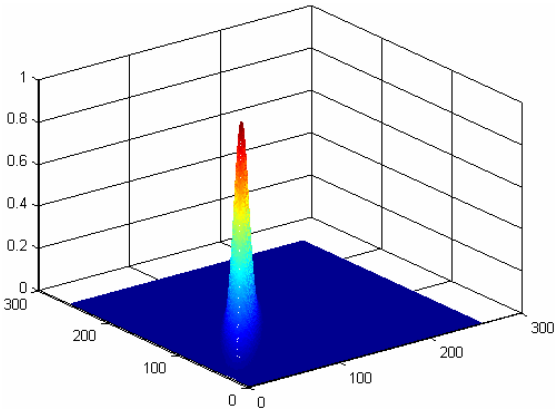


Figure 3.17: distribution de la couleur de peau pour l'espace Yiq

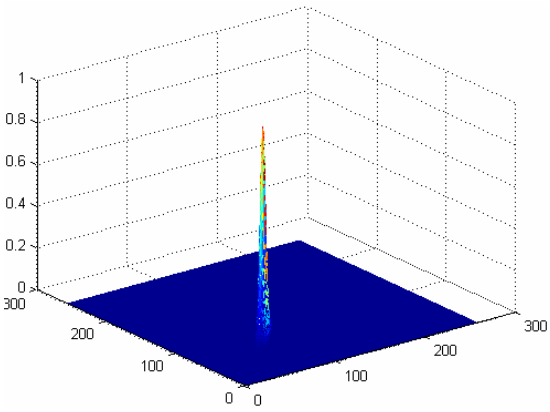


Figure 3.18: distribution de la couleur de peau pour l'espace Yxy

3.8. Résultats et discussion:

Deux méthodes ont été appliqués sur un ensemble de test de 40 images pour détecter la peau humaine, Dans notre expériences, on a utiliser dix espaces de couleur différents : RGB, HSV, RGBn, YCbCr, YUV, Ydbdr, Ypbpr, YIQ, Yxy et Hunter lab.

Sur cette série de test on a obtenu les résultats dans les tableaux ci-dessous (3.1 et 3.2). De par ces tests, nous avons testé l'influence de l'espace couleur sur deux méthodes de segmentation souvent utilisées dans la littérature (par seuillage et par modèle). La décision de savoir si un pixel est peau ou non peau se fait seulement avec les valeurs de couleur de ce pixel seul.

$$\text{Taux de détections correctes} = \frac{\text{Nombre de région de peau détectées}}{\text{Nombre de région de peau réels}}$$

$$\text{Taux de fausses détections} = \frac{\text{Nombre de fausses détections}}{\text{Nombre de détections}}$$

	RGB	RGBn	HSV1	HSV2	YCbCr	YUV	Ydbdr	Ypbpr	HLab	Yiq	Yxy
Taux de détection correcte	97.94 %	93.19 %	89.27 %	91.7 %	92.07 %	90.8 %	90.09 %	89.6 %	93.6 %	89.5 %	88.5 %
Taux de fausse détection	27.82 %	21.33 %	30.19 %	8.6 %	15.56 %	13.8 %	16.2 %	16.6 %	8.4 %	17.1 %	17.55 %

Tableau 3.1 : Performance de la détection de peau par seuillage.

	RGB	RGBn	HSV	YCbCr	YUV	Ydbdr	Ypbpr	HLab	Yiq	Yxy
Taux de détection correcte	96.03 %	94.83 %	91.9 %	95.15 %	94.22 %	92.7 %	93.1 %	94.1 %	93.02 %	91.11 %
Taux de fausse détection	32.22 %	23.97 %	17.4 %	13.48 %	14.23 %	15.98 %	15.33 %	9.37 %	15.35 %	19.12 %

Tableau 3.2 : Performance de détection de peau par modèle de peau.

Les méthodes paramétriques peuvent être vraiment lentes comme l'approche gaussiennes, sept fois plus lente que la méthode de seuillage, et leur performance dépend fortement de la forme de distribution de peau. En outre, la plupart des méthodes paramétriques de modèle de peau ignorent les statistiques de couleur non-peau, ce qui entraîne les fausses détections.

La meilleure performance a été réalisée par les espaces de couleur YCbCr et Hlab dans l'approche de modèle de peau avec un taux de bonne détection de 95.15 % et 94.1% respectivement et un taux de fausse détection de 13.48 % et 9.37% respectivement.

Plusieurs travaux sur la détection de peau ne fournissent pas la justification stricte de leur choix de l'espace de couleur, probablement en raison de la possibilité d'obtenir une détection acceptable de peau avec presque n'importe quel espace de couleur. Seulement quelques travaux ont été consacrés à l'étude comparative de différents espaces de couleur utilisés pour la détection de peau comme dans **[34]** et **[35]**. Dans l'approche paramétrique, plusieurs articles qui doutent sérieusement de n'importe quelle influence significative de choix de l'espace de couleur sur le résultat final de détection de peau comme dans **[35]**. Contrairement, la performance des approches paramétriques de détection de peau dépend fortement du choix de l'espace de couleur, ceci peut être observé par les résultats obtenus dans **[34]**.

Nous avons testé si une transformation de couleur améliore les performances de la détection ou pas. Où nous constatons que la transformation de l'espace de couleur aide d'une manière significative pour une meilleure segmentation des deux approches (par seuillage et par modèle), quelque soit la méthode de segmentation, le nombre de régions de peau finales dépend grandement de l'espace couleur utilisé.
Ainsi dans la méthode de seuillage et pour certains espaces de couleur, on a ignoré la composante illumination parce qu'elle n'ajoute pas beaucoup d'information à la séparation de la couleur de peau de non-peau.

Aux vues des résultats présentés sur les tableaux ci-dessus, nous pouvons conclure que l'utilisation de l'espace de couleur RGB et RGBn ne sont pas appréciables dans les deux méthodes à cause de leur taux de fausse détection très élevé, Dans la méthode de seuillage, et de même pour le HSV1, le HSV2 produit un taux de fausse détection inférieurs, mais son taux de détection correcte reste faible. Alors que pour la méthode de modèle de peau, le HSV donne un résultat raisonnable.
Le résultat de segmentation pour le Yxy dans les deux méthodes reste médiocre.
Les espaces couleurs Ypbpr, Ydbdr, YUV et Yiq, ont donné un résultat acceptable pour la méthode de modèle.
Une bonne segmentation a été obtenue par la méthode de modèle avec les espaces de couleurs Ycbcr et Hlab.

Pour la méthode de seuillage les espaces de couleur HLab et Ycbcr nous ont donné de bons résultats, mais l'inconvénient est qu'ils ont pas capable de segmenter certains couleurs de peaux qui diffèrent d'une personne à une autre (les gens de couleur noir par exemple), et ça se dit pour tous les espace couleurs utilisés dans la méthode de seuillage.
L'espace couleur HLab seul a réalisé la meilleure segmentation pour les deux approches avec un taux de fausse détection minimal.

Nous avons également constaté que l'approche de modèle de peau fait une différence sur la performance de détection, elle est significativement meilleure que l'approche de seuillage, mais il faut être attentif au choix de l'espace couleur.

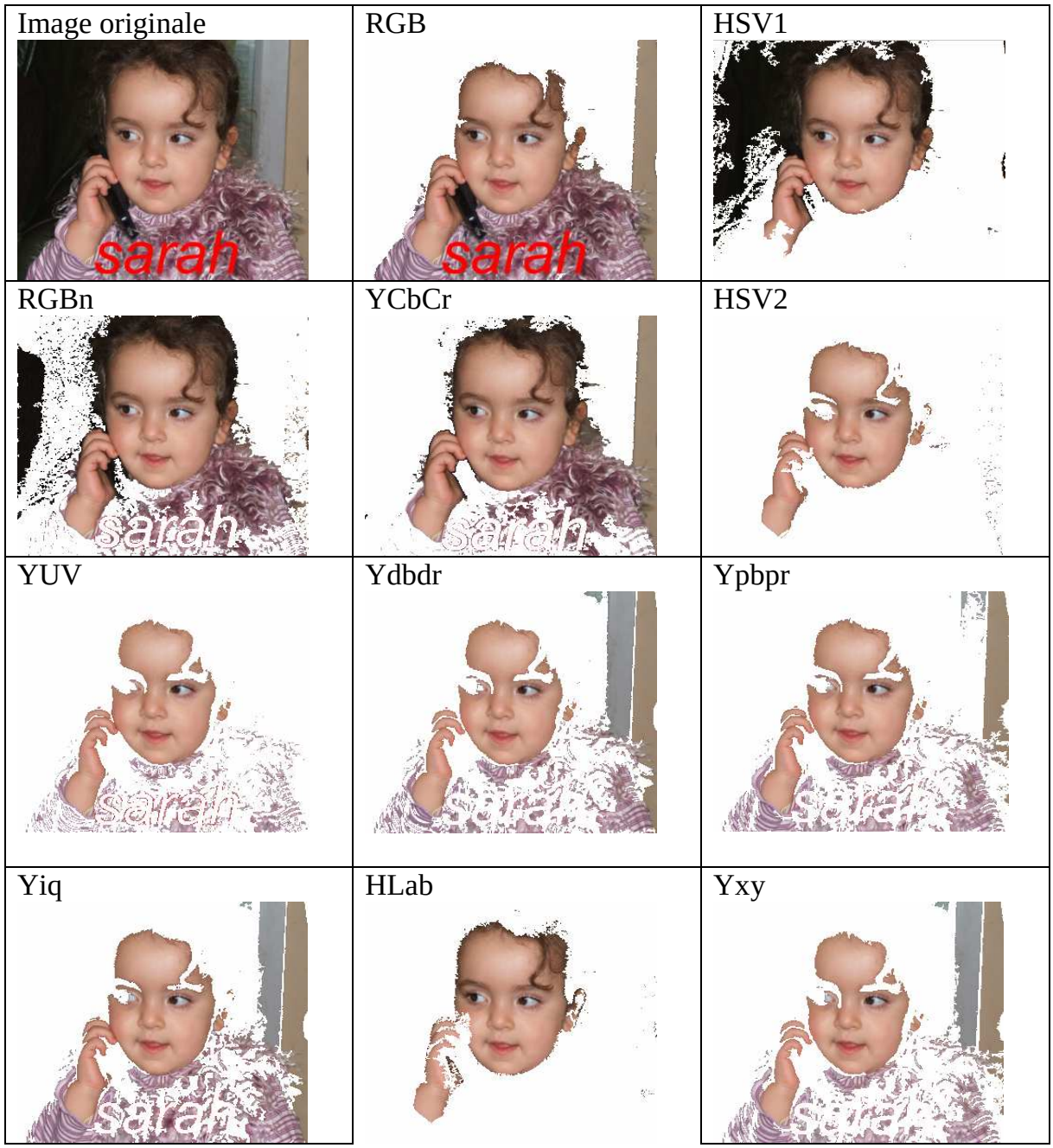


Figure 3.19 : segmentation de peau par seuillage pour l'image de test 1

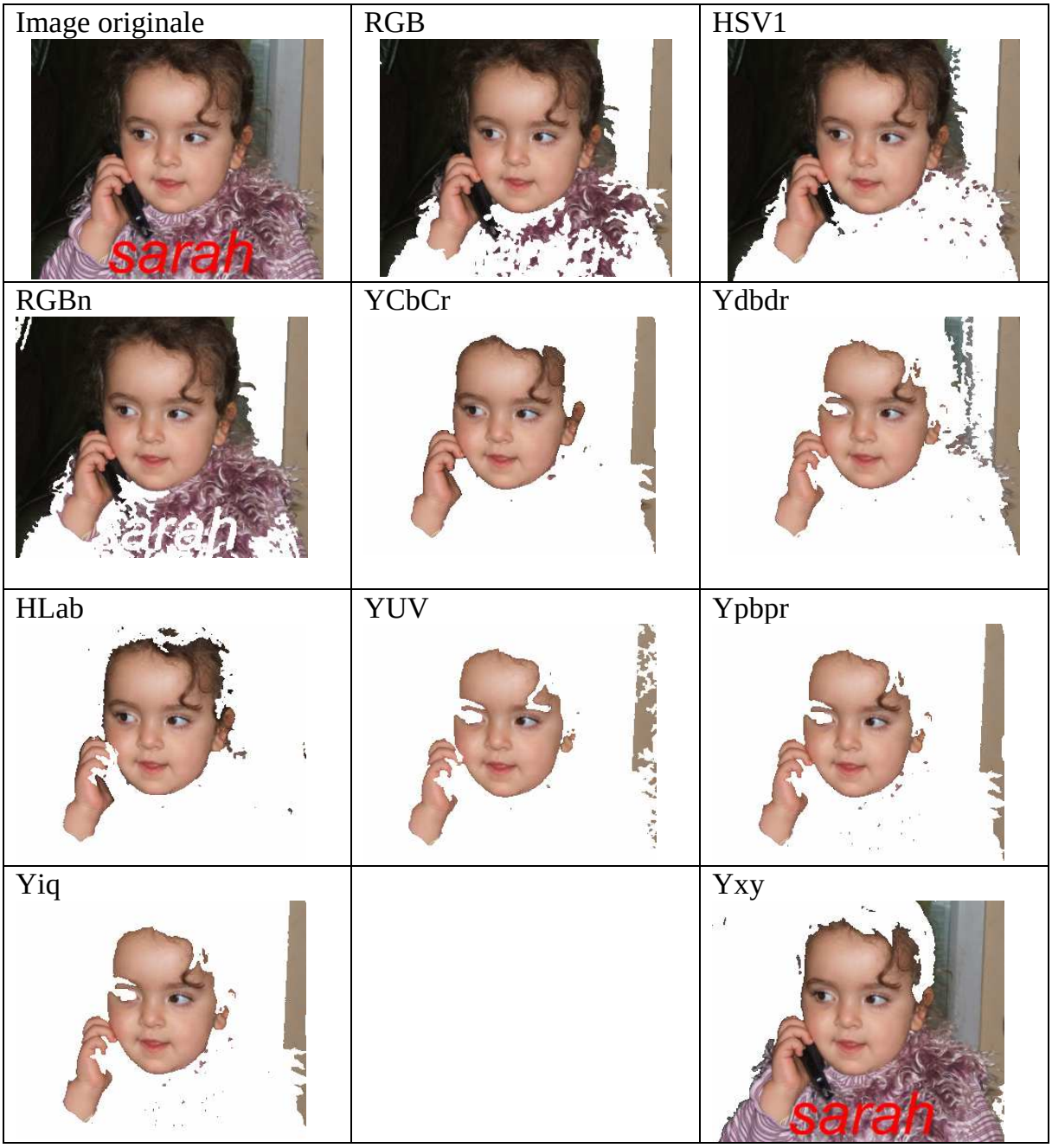


Figure 3.20: segmentation de peau par modèle pour l'image de test 1

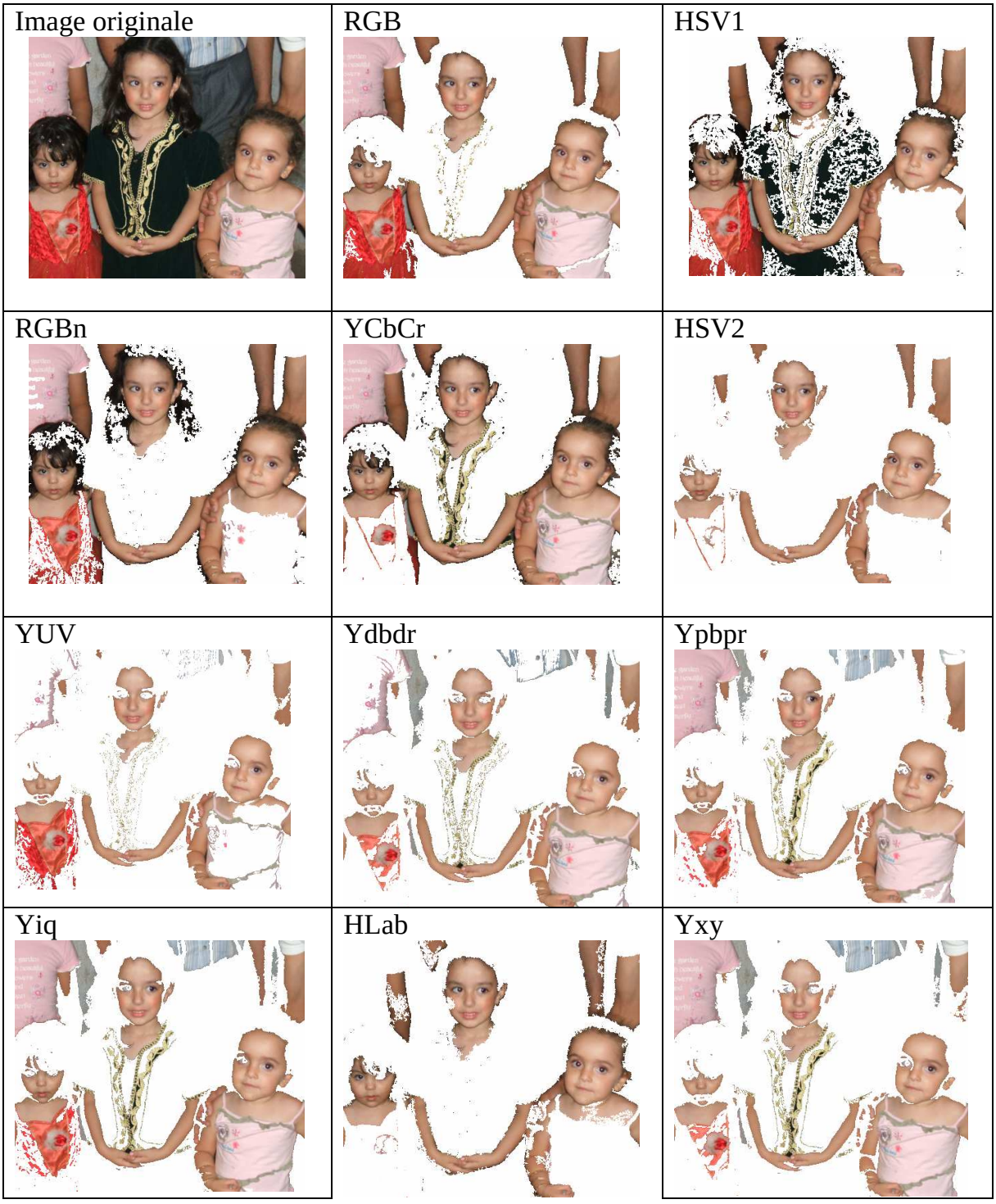


Figure 3.21 : segmentation de peau par seuillage pour l'image de test 2

Figure 3.22: segmentation de peau par modèle pour l'image de test 2

3.9. Conclusion :

La détection de peau est un processus préliminaire important dans la détection de visages humains. Ce travail fournit une méthodologie de test qui tient compte du choix de la meilleure approche de détection de peau. Cette analyse indique que la transformation de l’espace de couleur améliore la performance. La détection de la couleur de peau en utilisant l'approche de modèle est significativement meilleure que l'approche de seuillage, mais il faut être attentif au choix de l’espace de couleur.

Nous avons constaté que la meilleure performance de détection de peau a été obtenue en transformant la couleur de Pixel aux espaces de couleur YCbCr et Hunter lab, et en utilisant l'approche de modèle de peau. Les capacités de généralisation et d'interpolation de cette approche permettent de produire une performance acceptable.

Par cette étude comparative, nous avons fourni un outil pour choisir la meilleure approche ainsi que le meilleur espace couleur pour la détection de peau conçu pour notre détecteur de visage.

Chapitre IV

4.1. Introduction:

Les approches de reconnaissance de formes ont réalisé un succès mesurable dans le domaine de la détection visuelle. Chacune de ces approches utilise une machine d'apprentissage pour construire un détecteur d'un grand nombre d'exemples d'apprentissage.

Le système de détection de visage de viola, et Jones applique successivement des caractéristiques rectangulaire simples. Un visage "est détecté" si les combinaisons linéaires des réponses de ces caractéristiques excède un seuil prédéterminé. Chaque caractéristique rectangulaire est un filtre de somme de Pixel simple avec un poids et un seuil associés.

L'image de test est scannée à différentes échelles et positions en utilisant une fenêtre rectangulaire, et les régions qui passent par le classifieur sont déclarées comme des visages. Pour garantir une bonne performance, nous avons utilisé la segmentation de peau pour permettre au détecteur de scanner seulement les régions contenant la couleur de peau.

L'une des principales contributions de cette approche est l'extrêmement rapidité du calcul de ces caractéristiques rectangulaires en utilisant le concept de l'image intégral, ce qui permet de l'utiliser à la détection en temps réel.
Le Adaboost a trois buts principaux :

- Entraîné les caractéristiques efficaces d'un grand ensemble de caractéristiques.
- Construire les classifieurs faibles, dont chacun est basé sur un des caractéristiques choisies.
- Amplifier les classifieurs faibles pour construire un classifieur fort.

4.2. Caractéristiques de rectangle

La méthode de Viola et Jones est un système de détection basé sur des caractéristique, ces caractéristiques peuvent prendre plusieurs noms (Filtres de rectangle, caractéristique de rectangle, descripteurs de haar). Un ensemble de caractéristiques doit être créé et un système est utilisé pour trouver les bonnes caractéristiques. Ces bonnes caractéristiques, nous garantissent une meilleure discrimination entre les visages et les non visages. Différentes formes d'ensemble de caractéristiques peuvent être crées. Un souhaitable ensemble de caractéristique serait celui qui est exhaustive, qui a des formes caractéristiques capables de décrire l'objet et a des caractéristiques qui peuvent être appliquées à notre image et calculés de façon efficace.

Les caractéristiques de rectangle utilisées respectent toutes les conditions citées précédemment. Il y a six types de caractéristiques, des exemples de ces caractéristiques peuvent être vu dans la figure 4.2. La somme des intensités de pixel dans le cadre de la zone noir est déduite de la somme de celles de la zone blanche.

En appliquant à l'image un ensemble de filtres, à différentes échelles, on obtient la réponse de ces filtres sur l'image. Différents filtres permettent d'étudier les différences d'intensités horizontales, verticales et diagonales. Ainsi le procédé génère plusieurs ensembles de coefficients, un pour chaque orientation. Un ensemble des filtres est présenté sur la figure 4.1.

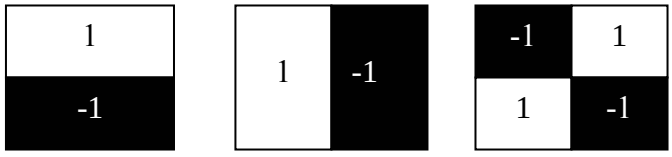


Figure 4.1 : Filtres de rectangle ou de Haar

La figure 4.3 montre le détecteur de base. La taille de détecteur de base est 24x24 pixels. Pour chaque type des caractéristiques, plusieurs caractéristiques sont crées à partir de ce type avec variation de taille et de position. La taille minimale de caractéristique rectangulaire est fixé à 8 pixels car une plus petite taille n'est pas

importante et aussi ils ont besoin de beaucoup de calcul. Ainsi, l'ensemble complet de caractéristiques rectangulaires est très élevé, 49554.

Type de caractéristique	(a)	(b)	(c)	(d)	(e)	(f)
Nombre de caractéristiques	12393	12393	7803	7803	6561	2601

Tableau 4.1 : Nombre de caractéristiques pour chaque type

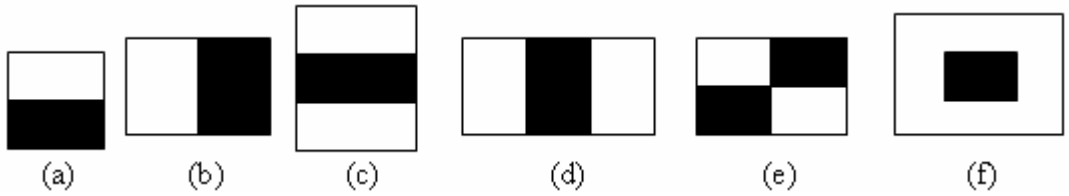


Figure. 4.2 : Six types de caractéristiques rectangulaires utilisées dans notre programme

caractéristique à *2-rec* : différence entre les régions (détection de contour).
caractéristique à *3-rec* : calcule la somme dans deux extrêmes rectangles déduite de la somme dans le rectangle de centre.
caractéristique à *4-rec* : différence entre les paires diagonales.

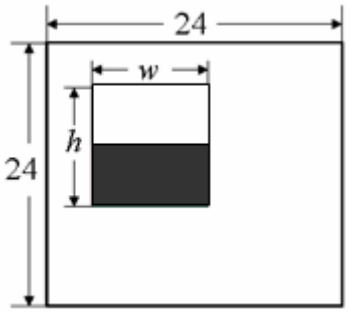


Figure. 4.3 : La taille du détecteur de base est 24x24 pixels. La taille de la caractéristique, w et h varie de 8 à 24.

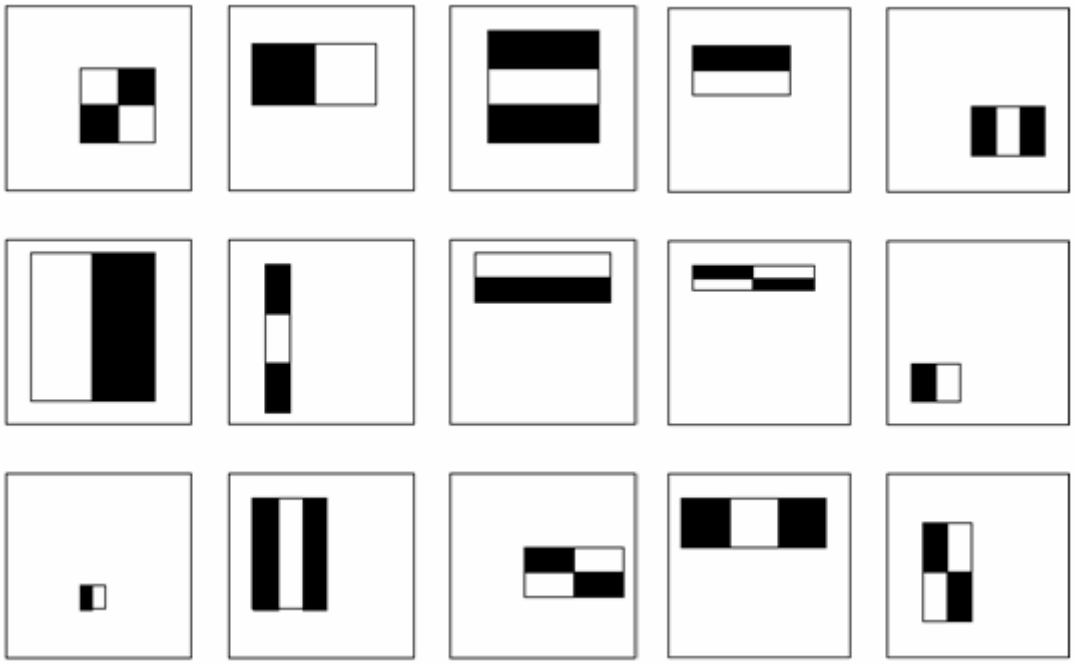


Figure 4.4 : Exemples des caractéristiques de rectangle dans une fenêtre de 24x24

Il y a aussi d'autres types de caractéristiques rectangulaires mais inclinés de 45 degrés comme présenté par Lienhart et Kuranov **[42]**, figure 4.5.

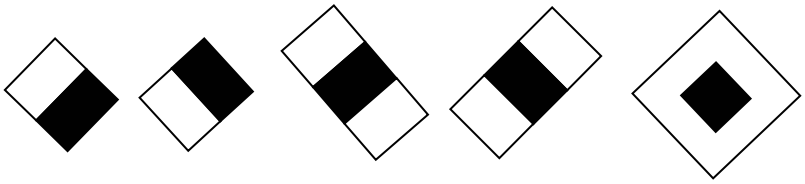


Figure 4.5 : caractéristiques rectangulaires orientées de 45°.

4.3. Images Intégrales

Les caractéristiques de rectangle peuvent être calculés de façon très efficace et rapide en utilisant une représentation intermédiaire de l'image appelée l'image intégrale. La valeur de l'image intégrale aux coordonnées (x, y) est la somme de tous les pixels au-dessus et à gauche de (x, y), y compris ce dernier point (Figure 4.6).

Soit ii l'image intégrale de l'image initiale i et $i(x, y)$ est la valeur de l'image intégrale au point (x, y):

On peut également définir l'image intégrale ii par:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \qquad (4.1)$$

Où $ii(x,y)$ est l'image intégrale et $i(x,y)$ est l'image originale.

Comme nous utilisons cette nouvelle représentation pour diminuer le temps de calcul, on peut expliquer ses avantages.

Tout d'abord, elle peut être calculée par un moyen efficace en utilisant la paire des récurrences suivante:

$$s(x,y) = s(x,y-1) + i(x,y) \qquad (4.2)$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \qquad (4.3)$$

D'où $s(x,y)$ est la somme cumulative, $\forall x, s(x,-1) = 0$, et $\forall y, ii(-1,y) = 0$. L'image intégrale peut être calculé en un seul passage sur l'image originale.

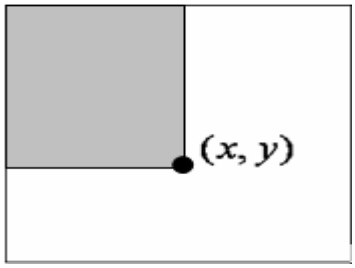


Figure 4.6 : La valeur de l'image intégrale au point (x, y)

En utilisant l'image intégrale, toute somme rectangulaire peut être calculée en quatre opérations (voir la figure 4.7). Clairement la différence entre les deux sommes rectangulaires peut être calculée en huit opérations, la caractéristique à deux rectangles définis ci-dessus, implique des sommes rectangulaires adjacentes qu'elles peuvent être calculés en six opérations, huit dans le cas d'une caractéristique à trois rectangles, et neuf pour la caractéristique à quatre rectangles.

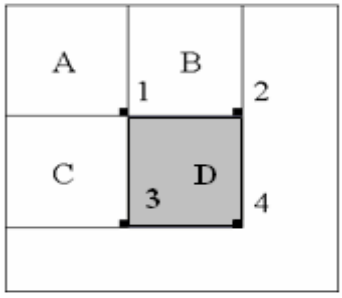


Figure 4.7: Calcul de la somme du rectangle D avec l'image intégrale

La somme des pixels dans le rectangle D peut être calculé avec quatre opérations. La valeur de l'image intégrale au point 1 est la somme des pixels dans le rectangle A. La valeur à la position 2 est A + B, à la position 3 est A + C, et à la position 4 est A + B + C + D. la somme dans D peut être calculée comme 4 +1-(2 +3).

Parce que la somme d'intensités dans une région rectangulaire ((x_1 ; y_1) ; (x_2 ; y_2)) est:

$$r = II(x_1, y_1) - II(x_1, y_2) - II(x_2, y_1) + II(x_2, y_2). \quad \ldots\ldots(4.4)$$

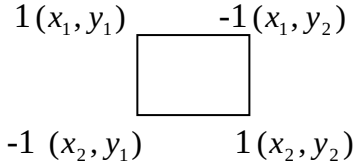
Où *II* est l'image intégrale.

Par conséquent, la réponse d'un rectangle est réduite de $M \times N$ opérations à juste quatre opérations, où M et N sont respectivement la largeur et la hauteur du rectangle.

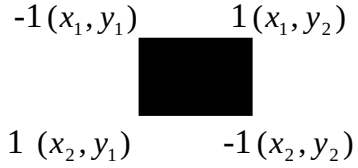
Pour cela une nouvelle représentation pour chaque filtre de rectangle est trouvée.

Une des conséquences de l'utilisation d'une telle représentation est la façon de scanner les images. Cette représentation utilise quelques nombres pour nous indiquer comment appliquer chaque caractéristique à une image d'apprentissage.

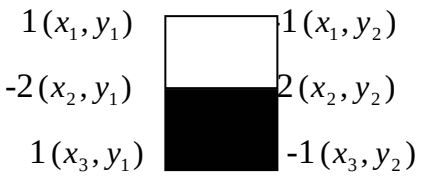
Par la dernière formule r on peut modeler chaque caractéristique comme le montre la présente figure :



Pour le rectangle blanc



Pour le rectangle noir



Pour l'ensemble de deux rectangles

Figure 4.8: Nouvelle représentation des filtres de rectangle

Chaque nombre nous indique que le Pixel de l'image à cet endroit devrait être ajouté ou être soustrait pour produire la réponse de la caractéristique.

(127,1)	1
(131,1)	-2
(135,1)	2
(139,1)	-1
(452,1)	-1
(456,1)	2
(460,1)	-2
(464,1)	1

Le 1er caractéristique choisi par AdaBoost « caractéristique à 3 rectangles »

(132,1)	1
(139,1)	-2
(146,1)	1
(332,1)	-1
(339,1)	2
(346,1)	-1

Le 10ème caractéristique choisi par AdaBoost « caractéristique à 2 rectangles »

(131,1)	1
(137,1)	-2
(143,1)	1
(331,1)	-2
(337,1)	4
(343,1)	-2
(531,1)	1
(537,1)	-2
(543,1)	1

Le 25ème caractéristique choisi par AdaBoost « caractéristique à 4 rectangles »

Figure 4.9 : Vecteurs creux de quelques caractéristiques avec l'endroit de chacun de leurs éléments

Chaque caractéristique est remodelée comme vecteur colonne clairsemé avec des éléments aux coins des rectangles noirs et blancs. L'endroit des coins (colonnes gauches) indique la position de la valeur de l'image intégrale d'une image qui doit être multipliée par le nombre de la caractéristique de rectangle à cet endroit (colonnes droites).

Dans ce cas, la sortie de chaque caractéristique appliquée à toute image est simplement le produit scalaire de vecteur caractéristique et le vecteur de l'image intégrale.

La représentation de l'image intégrale permet donc de calculer une seule caractéristique à chaque endroit et à chaque échelle en quelques opérations.

La puissance de toutes ces caractéristiques indépendantes est encore assez faible, Le défi est donc de trouver la manière dont les meilleures caractéristiques sont sélectionnées et comment nous pouvons les combiner pour produire un classifieur final.

4.4. Apprentissage par AdaBoost

Dans cette section, nous expliquons la méthode et les procédures utilisées et nous donnons une description précise de l'approche et la mise en œuvre des différents aspects de l'apprentissage de ce détecteur de visages.

Etant donné les exemples négatifs et positifs, on initialise les poids de chaque exemple et on applique un processus de boosting. T itérations sélectionnant T classifiers faibles $h_t(x)$ avec chacun un poids α_t pour obtenir finalement un classifier fort h_t. Pour chaque itération, un classifier est choisi en minimisant le taux d'erreur calculé avec les poids courrant des exemples. On augmente ensuite les poids des exemples mal classifiés 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.

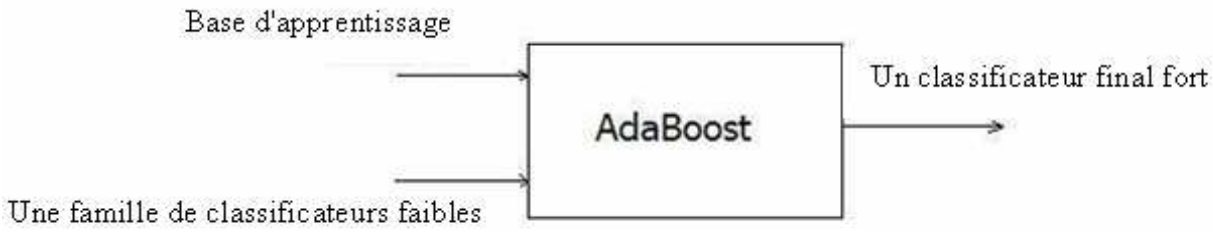


Figure 4.10 : Classifieur AdaBoost.

Soient les images d'exemple $(x_1, y_1),\ldots (x_n,y_n)$, ou $Y_i = 1;-1$ pour les exemples positifs et négatifs respectivement.

Initialiser les poids $W_{t,i} = \frac{1}{2l};\frac{1}{2m}$ pour $Y_i = 1;-1$ respectivement, ou l et m sont les nombres des exemples positifs et négatifs respectivement.

- pour t=1,…, T :

1. normaliser les poids.

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

2. pour chaque caractéristique j, former un classifieur h_j

L'erreur est évaluée à l'égard de w_t , $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$:

$\varepsilon_j = \sum_i w_i$, si et seulement si $y_i \neq h_j(x_i)$.

3. choisir le classifieur. h_t, avec la plus basse erreur ε_t .

4. faire la mise à jour des pondérations :

$w_{t+1,i} = w_{t,i} \exp(-1 \cdot (y_i \cdot \alpha_t \cdot h_i))$

Pour les visages $y_i = 1$, pour les non visages $y_i = -1$.

Si x_i est classé comme un visage $h_i = 1$ si non $h_i = -1$.

$w_{t+1,i} = w_{t,i} \exp(-\alpha_t)$: Pour les exemples bien classifiés ($y_i = h_i$).

$w_{t+1,i} = w_{t,i} \exp(\alpha_t)$: Pour les exemples mal classifiés ($y_i \neq h_i$).

Où $\alpha_t = \frac{1}{2} \log \frac{1}{\beta_t}$, et $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$

- le classifieur fort final est: $h(x) = \begin{cases} 1 & si \quad \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ -1 & si\ non \end{cases}$

Une procédure intuitive pour l'apprentissage de nos caractéristiques est de trouver le meilleur ensemble des caractéristiques consisterait à construire le classifieur faible pour chaque caractéristique et de classer les caractéristiques en vue de la plus petite erreur de classification. Malheureusement cette méthode simple, ne fonctionne pas parce que cela va produire la première meilleure caractéristique et aussi un ensemble de caractéristiques qui ont l'apparence très semblable à la première mais en légères échelles ou décalés. Ces caractéristiques sont tous fondamentalement les mêmes et échouent sur les mêmes images de test.

Le algorithme Adaboost corrige ce problème en modifiant les pondérations utilisées dans le calcul de l'erreur de classification pour chaque classifieur faible.

$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|.$

La fonction de pondération est de la forme suivante: $w_{t+1,i} = w_{t,i} \exp(-1 \cdot (y_i \cdot \alpha_t \cdot h_i))$

Avec : $\alpha_t = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$

	Classé visage ($h_i = 1$)	Classé non visage ($h_i = -1$)
Visage ($y_i = 1$)	VP (w_i)	FN (w_i)
Non visage ($y_i = -1$)	FP (w_i)	VN (w_i)

Tableaux 4.2 : Fonction de pondération

VP : vrai positif
FP : faux positif
VN : vrai négatif
FN : faux négatif

Les exemples mal classifiés sont désormais plus pondérées, alors qu'une petite erreur sera aussi plus pondérée et cela assure que notre première meilleure caractéristique et de toute autre caractéristique semblable à elle ne sera pas choisi comme notre deuxième meilleure caractéristique. Cela signifie que notre deuxième meilleure caractéristique n'est plus semblable à notre première meilleure caractéristique et toute différente caractéristique est sélectionnée comme notre prochaine meilleure caractéristique. Cette deuxième meilleure caractéristique est idéalement (parfaitement) complémentaire à notre première meilleure caractéristique dans le sens qu'elle classifié correctement les exemples dont la première s'échoue. Ce processus est répété pour trouver le plus grand nombre de meilleures caractéristiques qu'on désiré. Il s'agit généralement d'un très long processus (voir l'annexe C). Une fois un certain nombre de meilleures caractéristiques ont été découvertes, ils peuvent être utilisés pour détecter les visages dans une image de test. Chaque caractéristique vote sur ce qu'il croit qu'une image de test est un visage ou non. Chaque vote α_t des caractéristiques est pondéré proportionnellement en log-inverse avec l'erreur de ce caractéristique :

$$\alpha_t = \frac{1}{2}\log\frac{1}{\beta_t} \quad \text{avec : } \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}.$$

Donc, une caractéristique avec une plus petite erreur obtient le plus lourd vote pondéré.

4.4.1. Choix des caractéristiques:

Notons que l'ensemble exhaustif des caractéristiques de rectangle est complet. La pleine utilisation de cet ensemble c'est tout à fait un calcul inefficace et ne permet pas de détecter en temps réel. Donc, il faut réduire le nombre de caractéristiques tout en conservant le taux de détection.

Nous pouvons construire un détecteur de visage en appliquant tous les masques (filtres) à chaque endroit de l'image (pour Chaque décalage et chaque échelle). La prochaine étape dans la construction du détecteur de visage est donc d'utiliser une fonction d'apprentissage qui sélectionne un petit ensemble de ces caractéristiques. Celles qui sépare les meilleurs exemples positifs et négatifs. Le classifieur final résultant serait une simple combinaison linéaire de ces quelques caractéristiques rectangulaires.

Alors que les caractéristiques de rectangle initiales choisis par AdaBoost sont pertinentes et faciles à interpréter. La première caractéristique choisie se concentre sur la zone des yeux (Figure 4.11). La deuxième caractéristique sélectionnée caractérise la différence d'intensité entre la zone des yeux et la zone des pommettes. Ces caractéristiques sont relativement plus importantes, et devraient êtres un peu insensibles à la taille du visage.

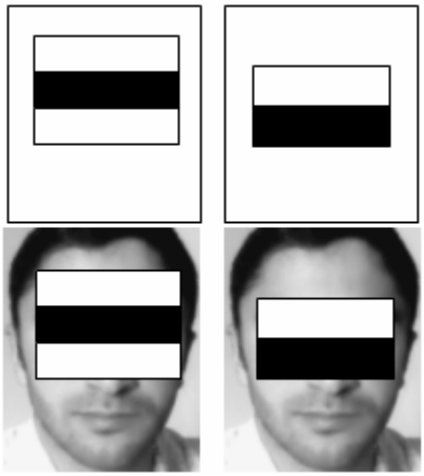


Figure. 4.11 : Première et deuxième meilleure caractéristique sélectionnée par AdaBoost

4.4.2. Classifieur faible :

La base d'apprentissage des visages et non visages normalisé ou est établie par une taille de 24x24 pixels pour chaque image. Pour chaque caractéristique un classifieur faible est défini par l'application de la caractéristique à la base d'apprentissage.

Le principe de boosting, est de combiner des classifieurs faibles qui sont appelés aussi faibles apprenants. Ces faibles apprenants dont AdaBoost a besoin, sont simplement établis à partir des filtres de rectangle.

Ces faibles apprenants sont appelés faibles, car nous ne prévoyons même pas la meilleure fonction de classification pour bien classifier les données. Un moyen facile de lier le faible apprenant et les caractéristiques de rectangle est d'assigner un faible apprenant à une caractéristique. Donc l'algorithme AdaBoost choisira à chaque itération la caractéristique qui offre la meilleure séparation entre les exemples positifs et négatifs, ainsi le meilleur seuil et meilleur parité associés à chaque caractéristique.

Un classifieur faible est donc l'évaluation d'une caractéristique suivie d'un seuillage optimal. Un seuil optimal qui sépare mieux les visages des non visages est donc trouvé, c'est-à-dire optimal dans le sens qu'il y a une meilleure discrimination entre les visages et les non-visages.

Ce seuil optimal est basé sur l'histogramme cumulatif (voir la figure 4.12).

Chaque caractéristique est calculée avec toutes les images d'apprentissage (visage et non-visage) à la fois, donc un ensemble des réponses de cette caractéristique avec tous les exemples positifs et négatifs est trouvé, on calcule ainsi l'histogramme cumulatif pour les réponses de la caractéristique avec les images positifs puis avec les images négatifs la meilleure réponse qui sépare mieux les visages des non-visages est donc choisi et on dénote par θ (seuil optimal).

Si la moyenne des réponses positives est inférieur ou égale celle des réponses négatives, alors la parité pour cette caractéristique est égale 1, sinon -1.

Alors chaque classifieur faible $h_j(x)$ a une caractéristique f_j, un seuil θ_j et une parité P_j qui lui sont assignées.

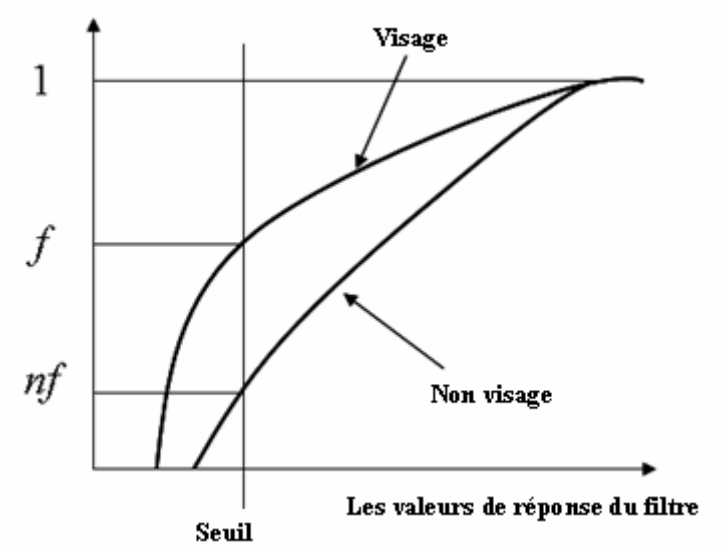


Figure. 4.12 : Seuil optimal

Pour notre implémentation la sortie de notre classifieur faible est 1 si l'image x est classée comme un visage et -1 si elle est classée dans la catégorie non visage. Mathématiquement, c'est l'équivalent de:

$$h_j(x) = \begin{cases} 1 & si \quad p_j f_j(x) < p_j \theta_j \\ -1 & si \quad non \end{cases}$$

Ou : $f_j = x \cdot f_i$, x est un exemple de la base d'apprentissage, et f_i est un filtre de rectangle.

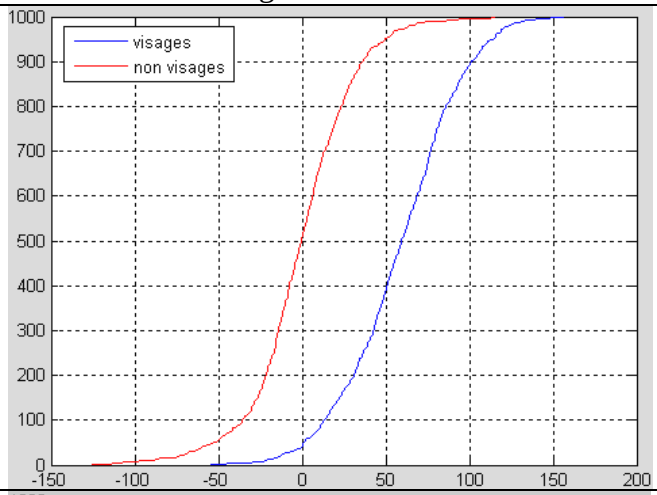
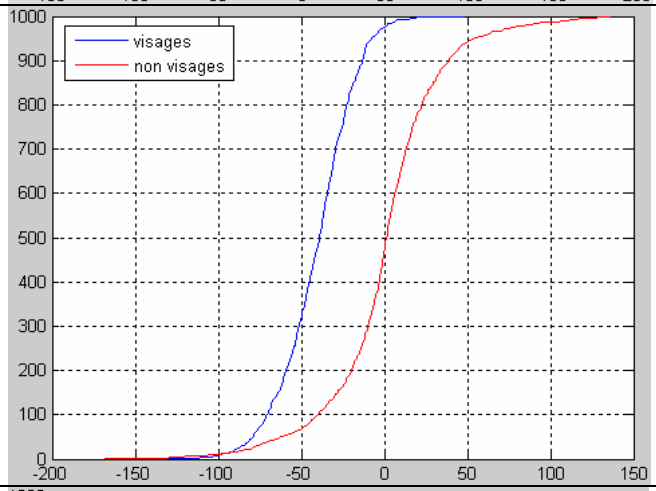
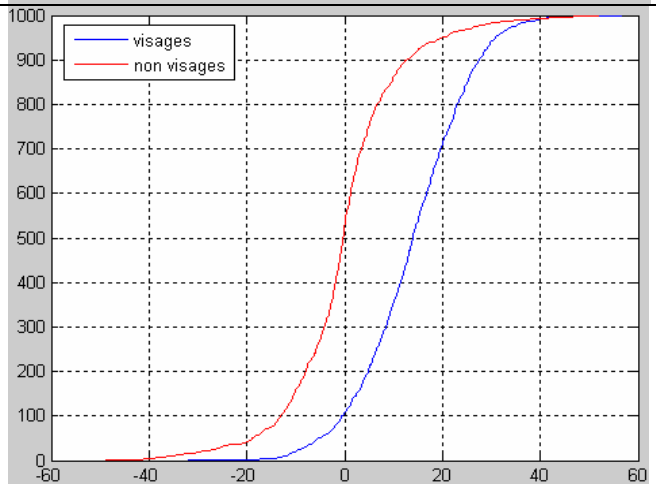
$h_j(x)$ est un classifieur faible qui se compose d'une caractéristique f_j, Un seuil θ_j et une parité p_j.

Ici x est une sous- fenêtre d'une image de 24x24 pixels.

Le classifieur fort (final) est la combinaison linéaire des classifieurs faibles sélectionnés

Par AdaBoost :

$$h(x) = \begin{cases} 1 & si \quad \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ -1 & si\, non \end{cases}$$

	L'histogramme cumulatif	Les valeurs associées
1er caract Choisie par AdaBoost		moy(visages)= 58.6462 moy(nonvisages)= -0.3635 Error=0,165 Theta= 30,458; P= -1; Alpha= 0,81074;
2ème caract Choisie par AdaBoost		moy(visages)= -40.8138 moy(nonvisages)= 0.7917 Error= 0,21024; Theta= -12,294; P= 1; Alpha= 0,66119;
3ème caract Choisie par AdaBoost		moy(visages)= 13.8300 moy(nonvisages)= -0.2118 Error= 0,22402; Theta= 5,434; P= -1 Alpha= 0,62119

Tableaux 4.3 : Trois premières caractéristiques choisies par AdaBoost et leurs valeurs associées

1er caract choisie par AdaBoost : N° de caractéristique : 26097 (caractéristique à 03 rectangles).
2ème caract choisie par AdaBoost : N° de caractéristique : 6264 (caractéristique à 02 rectangles).

3ème caract choisie par AdaBoost : N° de caractéristique : 43471 (caractéristique à 04 rectangles).

4.5. Implémentation de l'algorithme:

Nous avons utilisé une base de donnée des images de visages et de non visages téléchargée sur internet **[43]** (Nous avons utilisé 1000 images de visages et 1000 de non visages). Chaque image est de la taille de 24x24 pixels. Ainsi le détecteur ne doit pas être sensibles à leurs intensités, à cet effet les images ont été normalisées. Les figures (4.13 et 4.14) montre quelques exemples de la base d'apprentissage.

Figure 4.13: Un exemple des 1000 images de visage utilisé à l'apprentissage.

Figure 4.14 : Un exemple des 1000 images de non visage utilisé à l'apprentissage.

Une série complète de 49554 caractéristiques ont été produite en utilisant des caractéristiques de 24x24 pixels avec un minimum rectangle de la taille 8x8 pixels.

Le Stockage de 49554 caractéristiques de 24x24 pixels, prendrait une importante quantité de mémoire. Comme expliqué précédemment, le calcul serait également coûteux et inefficace. Pour faire face à ces problèmes, Adaboost choisi un certain nombre de ces caractéristiques. Afin de rendre plus efficace notre algorithme nous représentons chaque image intégrale comme un vecteur ligne, la sortie de chaque caractéristique appliquée à toute image est simplement le produit scalaire de vecteur caractéristique et le vecteur de l'image intégrale. Le seuil pour chaque classifieur faible a été trouvé en utilisant l'histogramme cumulatif.

Pour trouver chaque meilleure caractéristique on prend environ 3 à 4 heures de temps de calcul en utilisant un PC avec 2.6 GHz de vitesse de traitement et de 256 MB de mémoire. Cela était connu pour un inconvénient de la méthode de Viola et Jones à cause de la longue durée d'apprentissage.

4.6 Résultats expérimentaux :

Notre programme a été exécuté pendant environ une semaine pour trouver 30 meilleures caractéristiques. Viola et Jones utilisent 200 caractéristiques prenant ainsi plusieurs semaines, mais ces 30 caractéristiques nous ont donné de bons résultats. Figure 4.15, 4.16 et 4.17, indiquent les performances de notre détecteur. Nous pouvons vérifier que l'erreur diminue si nous utilisons plus de caractéristiques et notre détecteur qui utilise 30 caractéristiques donne un résultat raisonnable. Le détecteur de visage final balayer l'image sur les régions de peau en utilisant une fenêtre à différentes échelles et positions. Ce processus a un sens pour découvrir des visages à n'importe quelle échelle avec le même coût de temps. La première fenêtre est de taille 24*24, la suite devienne 30*30, alors que de bons résultats ont été obtenus en utilisant un facteur d'échelles de 1,25.

Le détecteur est également analysé dans tout endroit de peau. Les endroits ultérieurs sont obtenus en déplaçant la fenêtre à certains nombre de pixels Δ(Dans notre cas $\Delta = 10$) .

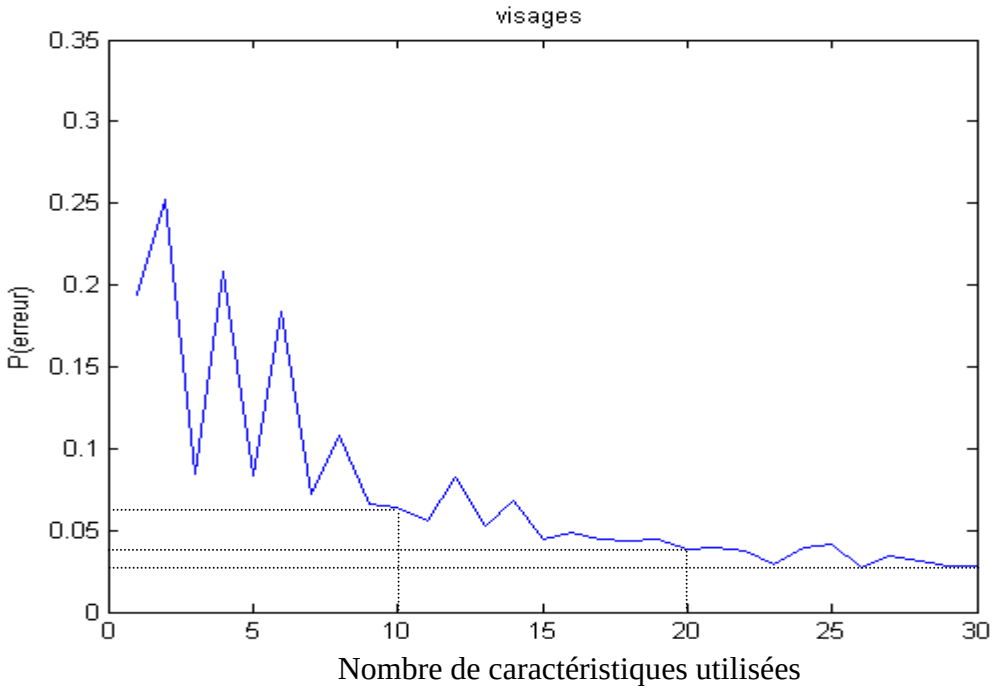


Figure 4.15: Taux d'erreur pour les visages par rapport au nombre croissant des caractéristiques utilisés

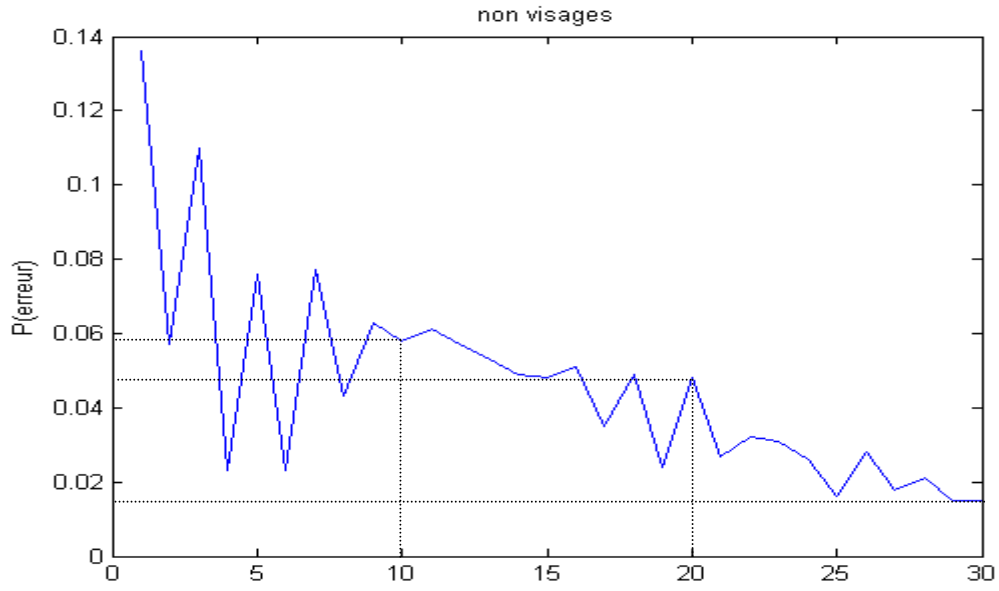


Figure 4.16: Taux d'erreur pour les non visages par rapport au nombre croissant des caractéristiques utilisés

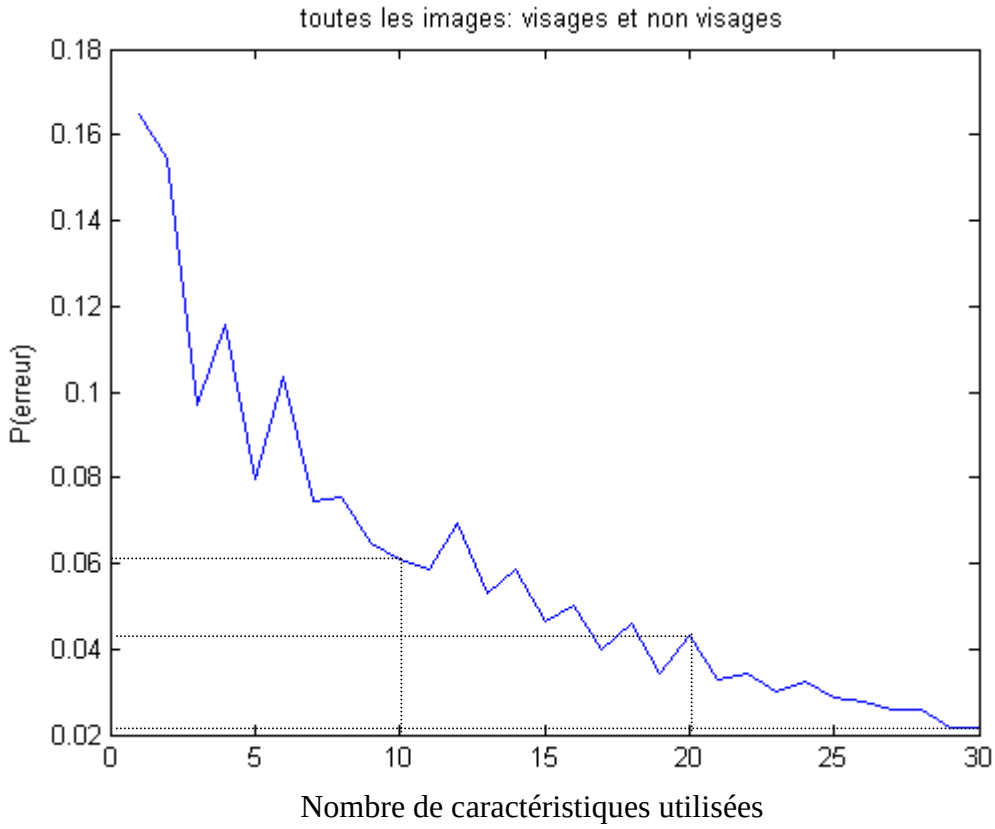


Figure. 4.17 : Le taux d'erreur global sur les images d'apprentissage positifs et négatifs

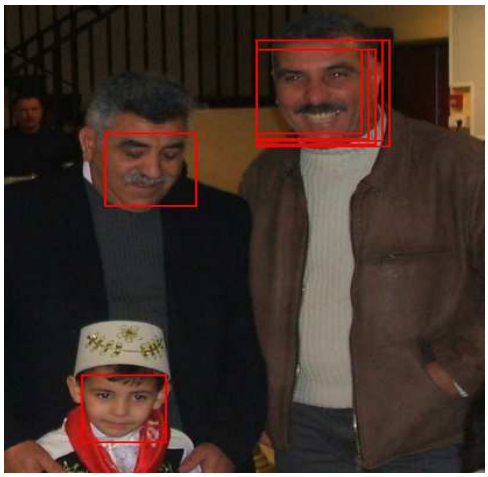
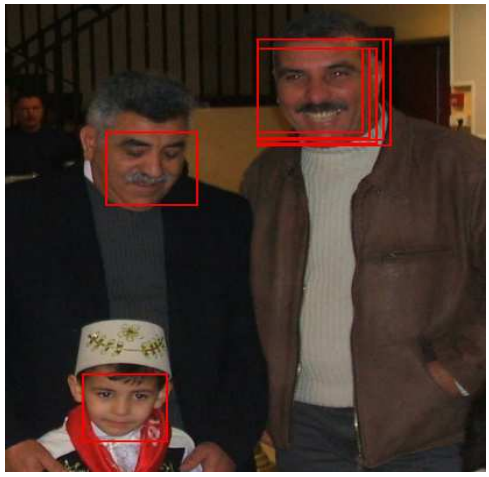
Le détecteur détecte les visages de face et de profil (jusqu'à 30°). L'information de couleur de peau permet de limiter l'espace où sont recherchés les visages ce qui présente l'avantage d'accélérer le processus de détection.

A partir de 200 caractéristiques de plusieurs types, le système de viola et jones a donné un taux de détection de 95% et un taux de fausse détection de 1 sur 14084 images de la base de test.

	T=10	T=20	T=30
classification correcte	88,06%	89,22%	91,25%
Fausse détection avec segmentation de peau	2,2%	3,1%	2,4%
Fausse détection sans segmentation de peau	8,7%	7,1%	8,5%

Tableau 4.4 : résultat de test sur 100 images en fonction de nombre d'itérations utilisées « T »

Evalué sur une base de 100 images fixes variées acquissent par nos soins et téléchargés sur internet, on a constaté qu'au fur et à mesure le nombre d'itérations augmente, le taux de classification correcte augmente aussi, figure 4.18.

	Sans segmentation de peau	Avec segmentation de peau
T=10	 101.156000 seconds	 15.265000 seconds
T=20	 122.610000 seconds	 16.984000 seconds
T=30	 129.344000 seconds	 17.109000 seconds

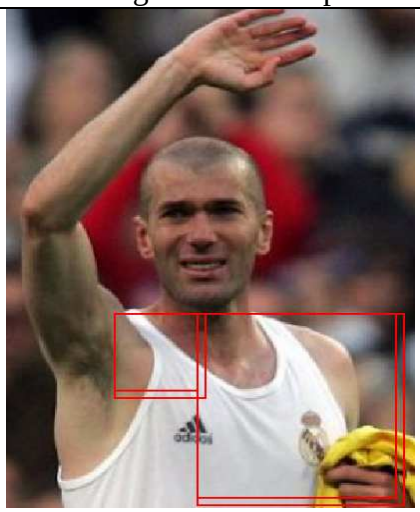
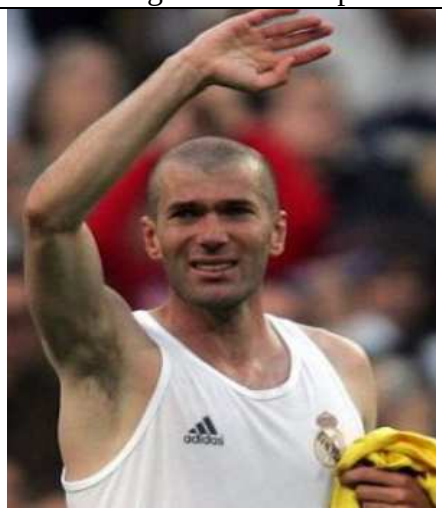
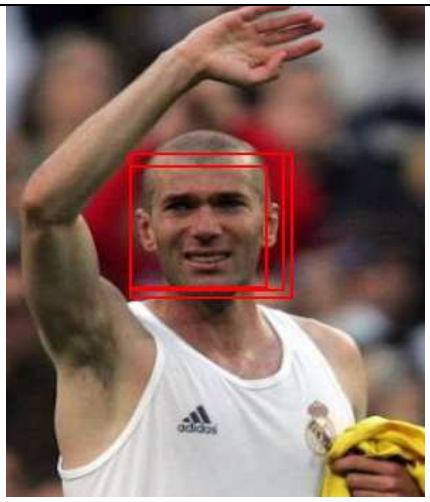
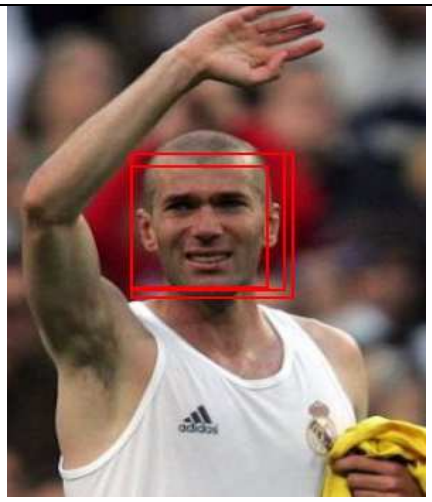
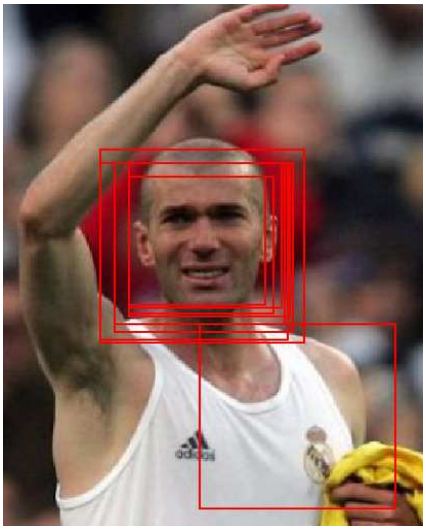
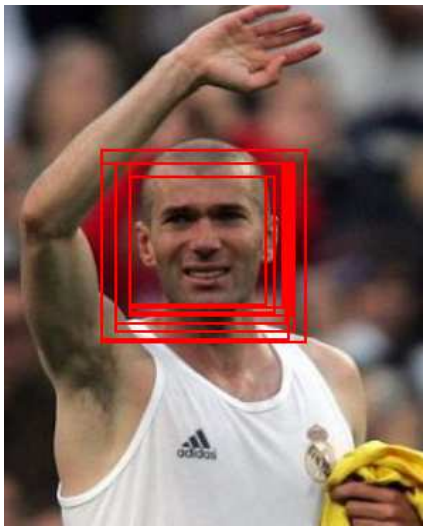
	Sans segmentation de peau	Avec segmentation de peau
T=10	 31.344000 seconds26.922000	 14.453000 seconds
T=20	 35.937000 seconds28.97800	 15.453000 seconds
T=30	 40.750000 seconds 31.54600	 18.375000 seconds

Figure 4.18 (b) : évaluation du détecteur en fonction de nombre d'itérations utilisées « T »

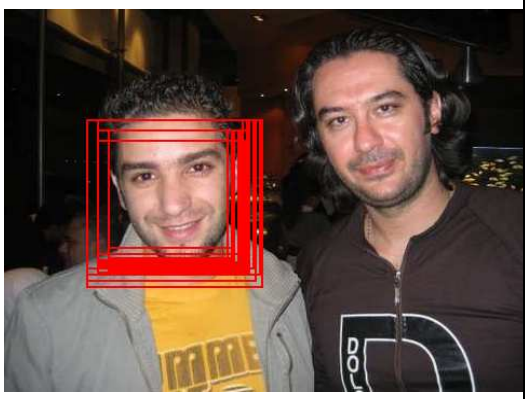
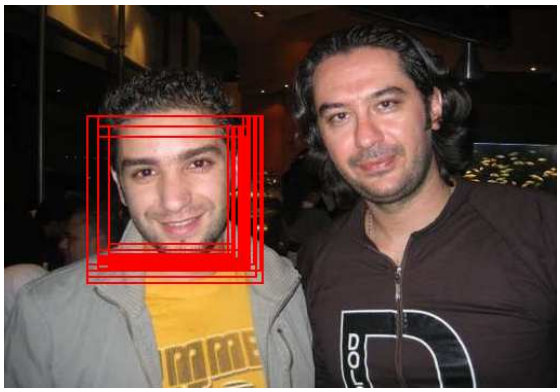
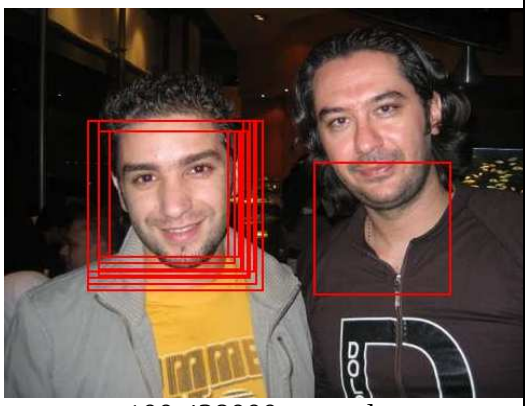
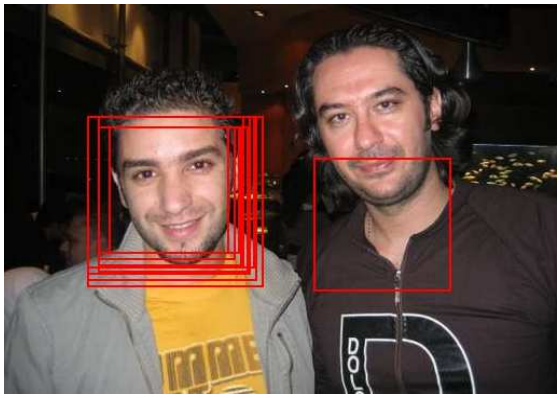
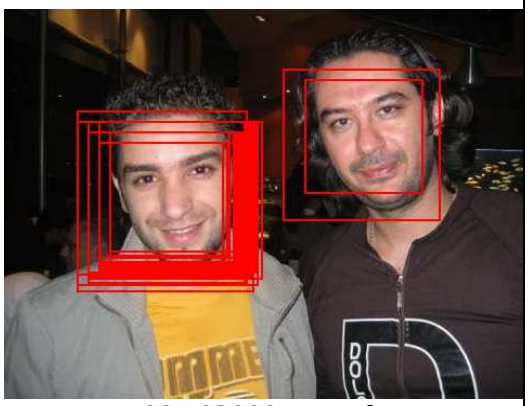
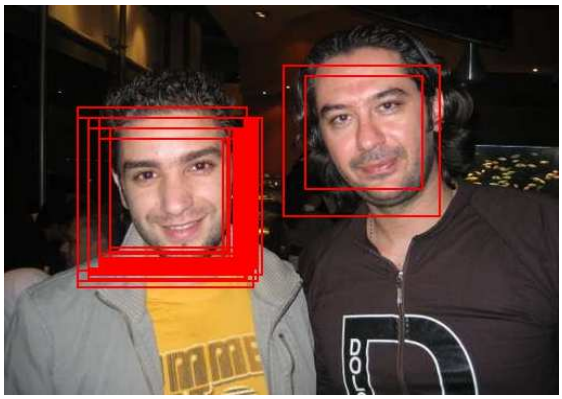
	Sans segmentation de peau	Avec segmentation de peau
T=10	 97.328000 seconds	 25.563000 seconds
T=20	 100.438000 seconds	 26.703000 seconds
T=30	 108.703000 seconds	 28.359000 seconds

Figure 4.18 (c) : évaluation du détecteur en fonction de nombre d'itérations utilisées « T »

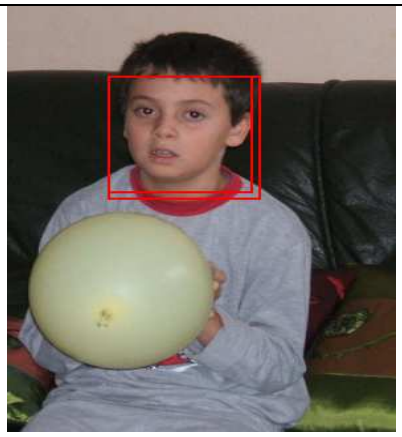
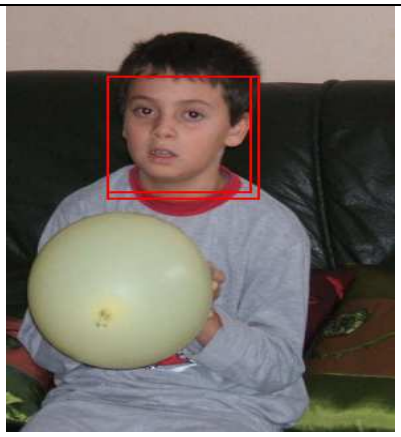
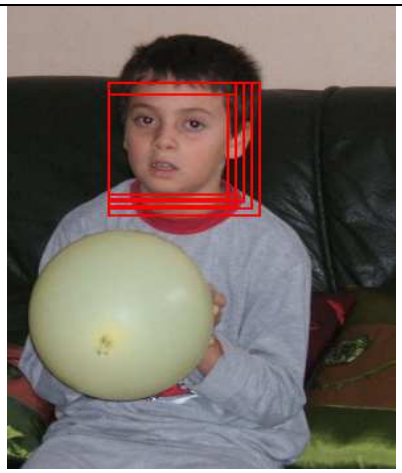
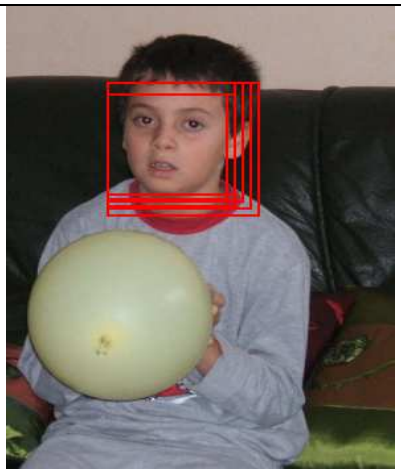
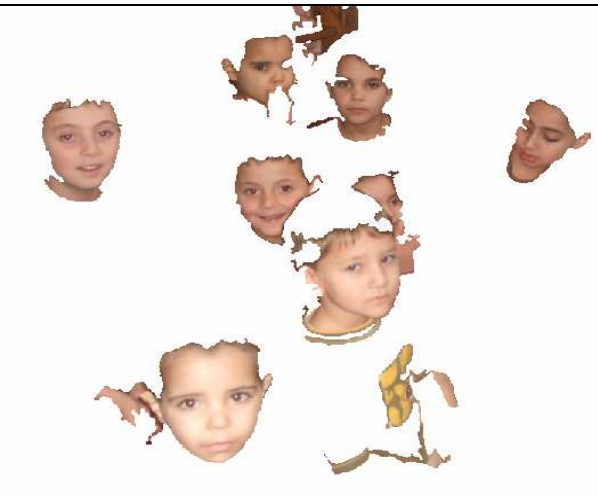
	Sans segmentation de peau	Avec segmentation de peau
T=10	 30.985000 seconds	 6.109000 seconds
T=20	 32.922000 seconds	 6.21000 seconds
T=30	 34.766000 seconds	 6.516000 seconds

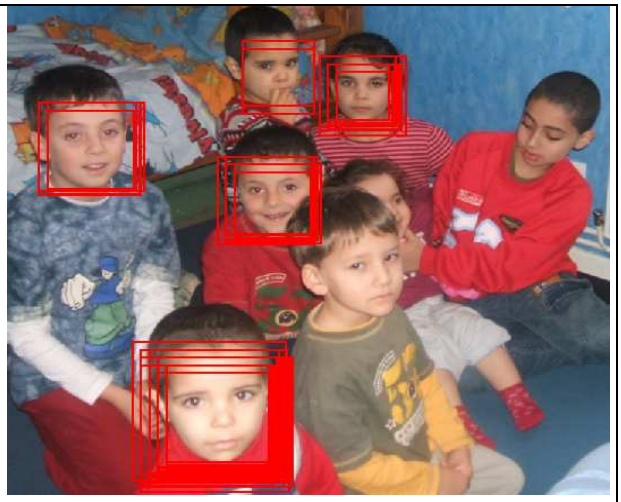
Figure 4.18 (d) : évaluation du détecteur en fonction de nombre d'itérations utilisées « T »

Notre détecteur réalise un taux de détection de l'ordre de 91,25%, le taux de fausses alarmes est de l'ordre de 2,4%, en utilisant seulement 30 caractéristiques de 6 types.
L'utilisation de l'information de couleur de peau aide à diminuer le taux de fausse détection, en segmentant l'image par approche modèle avec les espaces de couleur YCbCr et Hunter lab.
La vitesse de détection dépend de la taille de l'image testée, pour la dernière image de test de taille 301×404 pixels de la figure 4.18, la détection de visage prend environ 6.51 secondes en utilisant la couleur de peau, et environ 34,76 secondes en utilisant l'image entière, ce qui est plus long pour les deux par rapport à celle de Viola et Jones (0.7 secondes pour scanner une image de 384×288). Cela est dû principalement au logiciel de programmation (MATLAB.7) Si nous programmons avec d'autre langage comme C^{++}, et si nous utilisons un PC plus puissant, ce temps sera réduits, car la vitesse de traitement de l'ordinateur utilisé joue un rôle très important. Un autre moyen de le rendre plus rapide est de construire la structure en cascade. Cela est également proposé par Viola et Jones, mais nous laissons cette partie à un futur travail.
La figure 4.19, montre la détection de visage de quelques images de tests avec leur détection de peau.
L'image (1) montre les résultats de test de détection de visages en utilisant les deux espaces de couleur YCbCr et Hlab. Notre détecteur a réussi à détecter 4 visages parmi 6 six visages à différent échelle, les deux visages restantes ne sont pas détecté à cause de l'inclination et de l'orientation de ces deux visages, pour résoudre le problème il faut varier les images de la base d'apprentissage de sorte qu'elle puisse contenir plus des visages orientées et inclinées.
Dans les images de test (2), les visages sont soumissent sous une source lumineuse faible, notre détecteur a détecté tous les visages.
Dans l'image (3), un faux positif est supprimé en utilisant l'espace couleur Hlab.
L'image (4) contienne des visages à différentes échelles, un visage des trois n'est pas détecté, on peut résoudre ce problème en augmentant le nombre d'itérations dans la phase d'apprentissage. Dans l'image (5), un faux négatif dans l'image (5b) dû à cause de l'inclination de visage concerné. Pour l'image (5d), le Hlab a échoué

de détecter toutes les régions de peau ce qui a entraîné l'apparition d'un deuxième faux négatif.

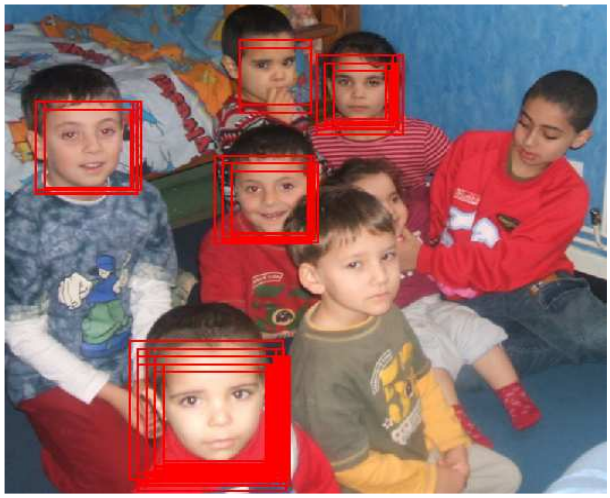


(1a) : YCbCr



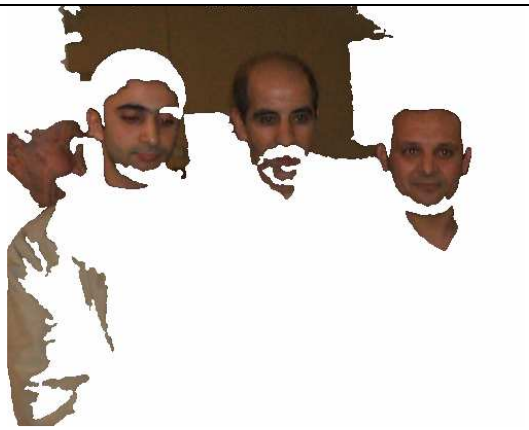
(1b)

(1c) : Hlab

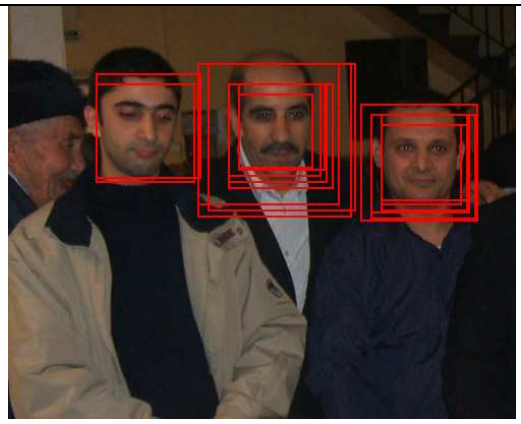


(1d)

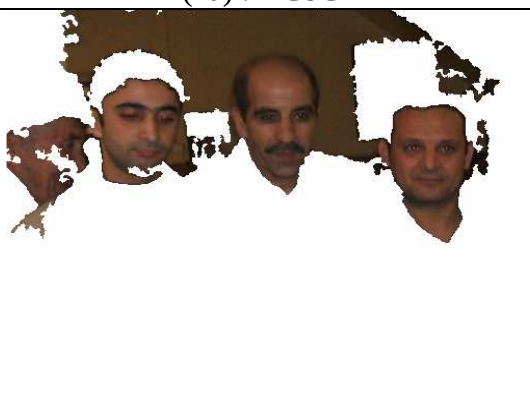
Figure 4.19 (a): Quelques tests de détection de peau et de visages



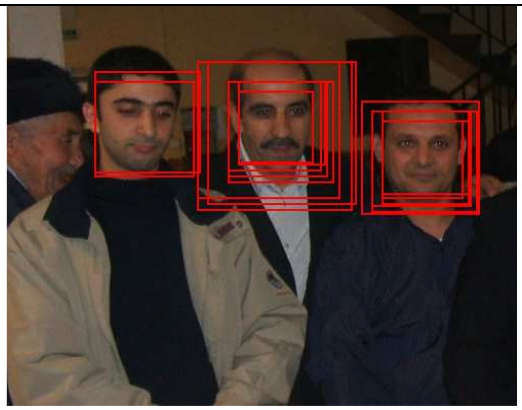
(2a) : YCbCr



(2b)

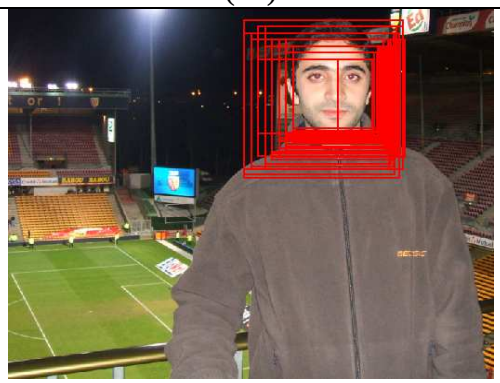


(2c) : Hlab

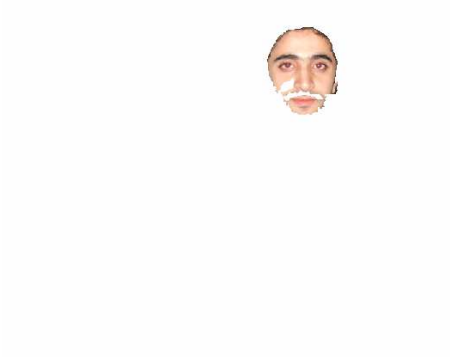


(2d)

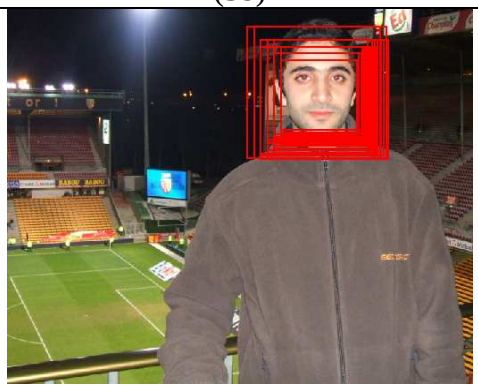
(3a) : YCbCr



(3b)



(3c) : Hlab



(3d)

Figure 4.19 (b): Quelques tests de détection de peau et de visages

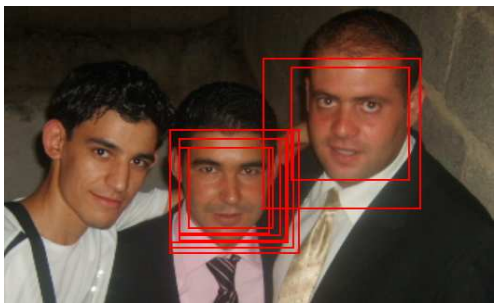
(4a) : YCbCr

(4b)

(4c) : Hlab

(4d)

(5a) : YCbCr



(5b)

(5c) : Hlab

(5d)

Figure 4.19 (c) : Quelques tests de détection de peau et de visages

4.7. Conclusion :

Nous avons essayé d’implémenter une nouvelle approche de boosting, en utilisant ainsi un procédé d'apprentissage pour extraire les caractéristiques qui représente mieux les visages. La principale contribution dans ce système de détection de visage, est l'utilisation de l’information de couleur à fin de réduire l’espace recherché pour trouver les visages, aussi utilisée pour améliorer la performance de notre détecteur surtout en terme de temps de calcul.

L'application de l'algorithme Adaboost modifié prévoit la réduction de l'ensemble exhaustif des caractéristiques rectangulaires. En l’absence de l’architecture de cascade et même si nous utilisons seulement 30 caractéristiques, ce détecteur donne des résultats très favorables.

Conclusion Générale & Perspective

Conclusion générale et perspective:

Beaucoup de méthodes peuvent être utilisées pour la tâche de détection de visage, un contexte précis pour chacune de ces méthodes est de détecter et localiser des visages dans une scène. Ces méthodes sont parfois combinées pour fournir des meilleurs résultats de détection.

Par le témoignage de plusieurs chercheurs travaillant sur l'approche de Viola et jones, cette méthode présente de très bons résultats et ouvre ainsi la possibilité à encore d'autres applications pratiques de l'analyse de visage. Ils ont pu réaliser un taux de détection correcte de 95% et un taux de fausse détection de 1 sur 14084 images de la base de test

Dans notre travail, nous avons choisi une méthode intermédiaire entre la méthode basée sur l'apprentissage et la méthode basée sur les caractéristiques invariantes. Cette méthode est basée sur la technique de détection de visage développée par Viola et jones en utilisant l'algorithme de AdaBoost et des filtres de rectangle. Nous avons vérifié que leur méthode a un taux de détection très élevé et capable de détecter en temps réel. Elle est également très souple dans le sens où elle peut être formé pour les différents niveaux de complexité, de vitesse et de taux de détection.

Notre contribution a permis la réduction de l'ensemble exhaustif des caractéristiques rectangulaires. Avec seulement 30 caractéristiques, nous avons réalisé un taux de détection de 91.25% et un taux de fausse détection de 2.4% sur 100 images de test acquissent par nos soins et d'autres téléchargés sur le web. Il est claire que les résultats de ce système de détection de visage ne sont pas vraiment comme celle obtenus par le travail original de Viola et jones, mais le système reste prometteur et beaucoup d'améliorations concernant la vitesse et les performances de détection sont possibles.

En l'absence de l'architecture de cascade, ce détecteur a montré un résultat raisonnable.

Les situations dans lesquelles, notre système a rencontré quelques difficultés de détections sont les suivantes :

- Il ne peut pas détecter des visages à basse résolution.
- Il n'est pas capable de détecter des visages qui sont tournés de plus de 30°.

- Lorsqu'un visage est illuminé d'un seul coté, dans ce cas les filtres de rectangle rendent la méthode moins robuste contre la variation de l'illumination.

Pour un futur travail nous proposons :

- D'intégrer l'architecture de cascade à l'apprentissage avec un grand nombre d'itérations.
- Une implémentation possible des autres types de caractéristiques décrits par viola et jones [21], et lienhart [23].
- Une amélioration de la vitesse du programme de système qui a été fait sous matlab et qui pourrait être réalisée avec un langage évolué tel que c^{++}.
- Nous avons essayé aussi d'utiliser les filtres de Gabor au lieu des caractéristiques rectangulaires, pour faire une comparaison entre les deux, mais on n'a pas obtenu de bons résultats, et nous laissons cette idée à un futur travail.

Bibliographie

Bibliographies:

[1] G.Yang et T.S.Huang, "**Human Face Detection In Complex Background**", Pattern Recognition, vol 27, no 1, pp 53-63, 1994.

[2] C.Kotropoulos et I. Pitas, "**Rule-based Face Detection in Frontal View**", Proc. Int'l Conf. Acoustics, Speech, and Signal Processing, vol. 4, pp. 2537-2540, 1997.

[3] H.P. Graf, T. Chen, E. Petajan, et E. Cosatto, "**Locating Faces and Facial Parts**," Proc. First Int'l Workshop Automatic Face and Gesture Recognition, pp. 41-46, 1995.

[4] T. K. Leung, M.C Burl, et P. Perona, "**Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching**", Proc. Fifth IEEE Int'l Conf. Computer Vision, pp 637-644, 1995.

[5] D.Chai et K. N. Ngan « **Face segmentation using skin-color map in videophone applications**» ;Circuits and Systems for Video Technology, IEEE Transactions on, Volume 9, Issue 4, Jun 1999 Page(s):551 – 564.

[6] S. McKenna, Y. Raja, et S. Gong, "**Tracking Colour Objects Using Adaptive Mixture Models**," Image and Vision Computing, vol. 17, nos. 3/4, pp. 223-229, 1998.

[7] Lin-Lin Huang Shimizu, et A. Kobatake, H."**Classification-based face detection using Gabor filter features**". Graduate Sch. of Bio-Applications & Syst. Eng., Tokyo Univ. of Agric. & Technol., Japan. This paper appears in: Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference.

[8] Jie Chen, et Shiguang Shan, “**Novel Face Detection Method Based on Gabor Features”,** SINOBIOMETRICS 2004, 90-99.

[9] Q. Chen, H. Wu, et M. Yachida, “**Face Detection by Fuzzy Matching**,” Proc. Fifth IEEE Int’l Conf. Computer Vision, pp. 591-596,1995.

[10] K. Sobottka, et I. Pitas, **"Face Localization and Feature Extraction Based on Shape and Color Information"**, Proc. IEEE Int'l Conf. Image Processing, pp 483-486, 1996.

[11] Liang Luhong, Ai Haizhou et Guangyou Xu “**Face. Detection Based on Template Matching and Neural Network Verification**” In Proceedings, Edited by: Y.J.Zhang ICIG’2000, First International Conference on Image and Graphics, Journal of Image and Graphics (JIG), Vol.5, Supp. pp.553-556 (2000).

[12] Joong-In Shin Hyun-Sool Kim, Woo-Seok Kang et Sang-Hui Park.
"**Face detection using template matching and ellipse fitting**". In IEICE TRANS.INF.SYST, volume E83-D,2000.

[13] T. Kohonen, "**Self-Organization and Associative Memory**". Springer 1989.

[14] T. Agui, Y. Kokubo, H. Nagashashi, et T. Nagao, “**Extraction of FaceRecognition from Monochromatic Photographs Using Neural Networks**,” Proc. Second Int’l Conf. Automation, Robotics, and Computer Vision, vol. 1, pp. 18.8.1-18.8.5, 1992.

[15] H. Rowley, S. Baluja, et T. Kanade, “**Neural Network-Based Face Detection,**” Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 203-208, 1996.

[16] K.-K. Sung et T. Poggio, “**Example-Based Learning for View-Based Human Face Detection**,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 39-51, Jan. 1998.

[17] Haizhou Ai, Lihang Ying et Guangyou Xu "**A Subspace Approche to Face Detection with Support Vector Machines**", p 45-48, 16 th International conference on Pattern (ICPR 2002), 11-15 august 2002, Guebec, Canada, IEEE Computer Society, ISBN 0-7695-1695-X**.**

[18] Michael Jones et Paul Viola. **"Fast Multi-view Face Detection",** the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 18, 2003.

[19] Yoav Freund et Robert Shapire. "**A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**". Journal of Computer and System Sciences, 55(1):119{139, aôut 1997.

[20] Paul Viola et Michael Jones. "**Rapid Object Detection using a Boosted Cascade of Simple Features**". Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 511{518, Kauai, Haway, Etats-Unis, décembre 2001.

[21] Paul Viola, Michael Jones et Daniel Snow."**Detecting Pedestrians Using Patterns of Motion and Appearance**". Dans Proceedings of the 9th IEEE International Conference on Computer Vision, volume 2, pages 734{741, Nice, France, octobre 2003.

[22] Paul Viola et Michael J. Jones. **"Robust Real-Time Face Detection".** International Journal of Computer Vision 57(2), 137–154, 2004. Kluwer Academic Publishers. Manufactured in The Netherlands.

[23] Rainer Lienhart et Jochen Maydt. "**An Extended Set of Haar-like Features for Rapid Object Detection**". Dans Proceedings of the IEEE International Conference on Image Processing, volume 1, pages 900{903, Rochester, New York, Etats-Unis, septembre 2002.

[24] Shachar Kaufman et Idan Ram "**Face Detection & Recognition**", project finished at VISL « Vision and Image Sciences Laboratory "Israël"», 2005.

[25] Anders Jörgensen, "**AdaBoost and Histograms for Fast Face Detection",** Royal Institute of Technology School of Computer Science and Communication, Master of Science Thesis Stockholm, Sweden 2006.

[26] Peng Yang, Shiguang Shan, **"Face Recognition Using Ada-Boosted Gabor Features",** The 6th International Conference on Automatic Face and Gesture Recognition. Seoul, Korea, May, 2004. Institute of Computing Technology of Chinese Academy Science.

[27] Leo Breiman, "**Bagging Presictors**", Machine learning, pages 123-140,1996.

[28] Yoav Freund Robert E. Schapire "**A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**", Proceedings of the Second European Conference on Computational Learning Theory, Barcelona, March, 1995."

[29] Leo Breiman. "**Arcing *Classifiers***". Annals of Statistics, 26(3):801-849, 1998.

[30] Yoav Freund et Robert E. Schapire "**Experiments with a New Boosting Algorithm**" in Proceedings of the Thirteenth International Conference on Machine Learning, 1996, pp. 148-156.

[31] R. Meir and G. Ratsch. "**An Introduction To Boosting and Leveraging"**.

Department of Electrical Engineering, Technion, Haifa 32000, Israel, 2002.
[32] V. I. Pavlovic and A. Garg. “**Boosted Detection of Objects and Attributes”**. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[33] Boris Ruf , “**Face Recognition using Boosting”,** EPFL / Computer Science Department**.** June 29, 2007.

[34] Jean-Christophe Terrillon et Shigeru Akamatsu, “**Comparative Performance of Different Chrominance Spaces for Color Segmentation and Detection of Human Faces in Complex Scene Images**,” International Conf on Face and Gesture Recognition, pp. 54-61, 2000.

[35] Alberto Albiol, Luis Toress, Edward Delp, “**Optimal color spaces for skin detection,**” *ICIP*, 2001.

[36] Poynton, C. A. 1995. “**Frequently asked questions about colour”**. COPYRGT. Jan. 4, 1998.

[37] Fleck, M., Forsyth, D. A., et Bregler, C. “**Finding naked people”**. In *Proc. of the ECCV*, vol. 2, 592–602. 1996.

[38] D.Chai et K. N. Ngan, “**Face segmentation using skin color map in videophone applications**”, IEEE Transactions on Circuits and Systems for video Technology 9 (4) (1999) 551-564.

[39] J. Kovac, P. Peer et F. Solina, "**2D versus 3D color space face detection**" 4th EURASIP Conference on Video/Image Processing and Multimedia Communication, Croatia, 2003, pp.449-454.

[40] S. Tsekeridou et I. Pitas, " **Facial feature extraction in frontal views using biometric analogies**", Proc. of the IX European Signal Processing Conference, vol.I,315-318 (1998).

[41] G. Gomez et E. F. Morales, "**Automatic feature construction and a simple rule induction algorithm for skin detection**", Proc. of the ICML workshop on Machine Learninng in Computer Vision, A. Sowmya, T. Zrimec (eds), 31-38 (2002).

[42] Rainer Lienhart, Alexander Kuranov, et Vadim Pisarevsky "**Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection"** Microprocessor Research Lab, Intel Labs Intel Corporation, Santa Clara, CA 95052, USA, 2003.

[43] Peter Carbonetto's homepage (**http://www.cs.ubc.ca/~pcarbo/**).

Annexes

Annexe (A)

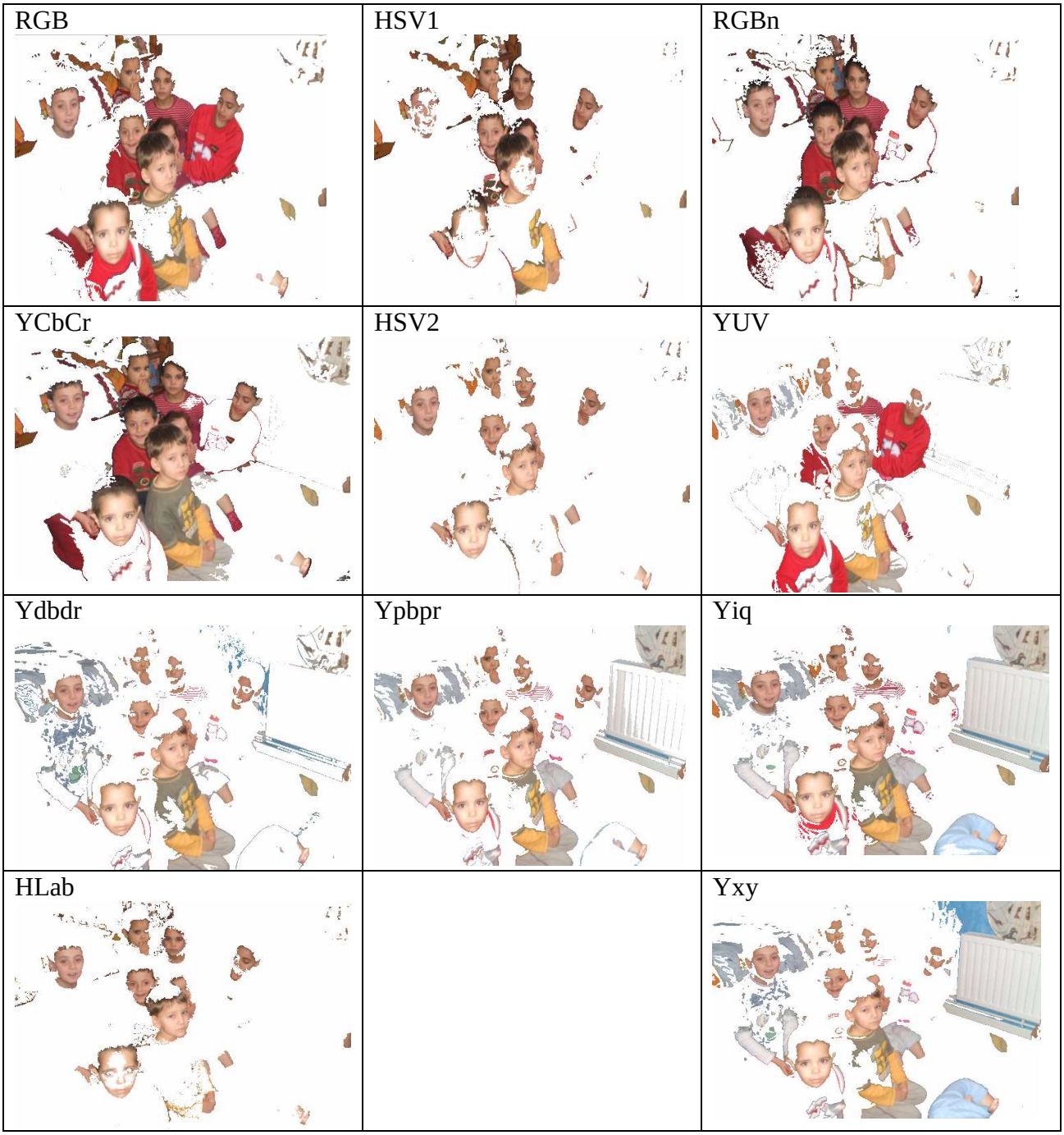


Figure 1: Segmentation de peau par seuillage

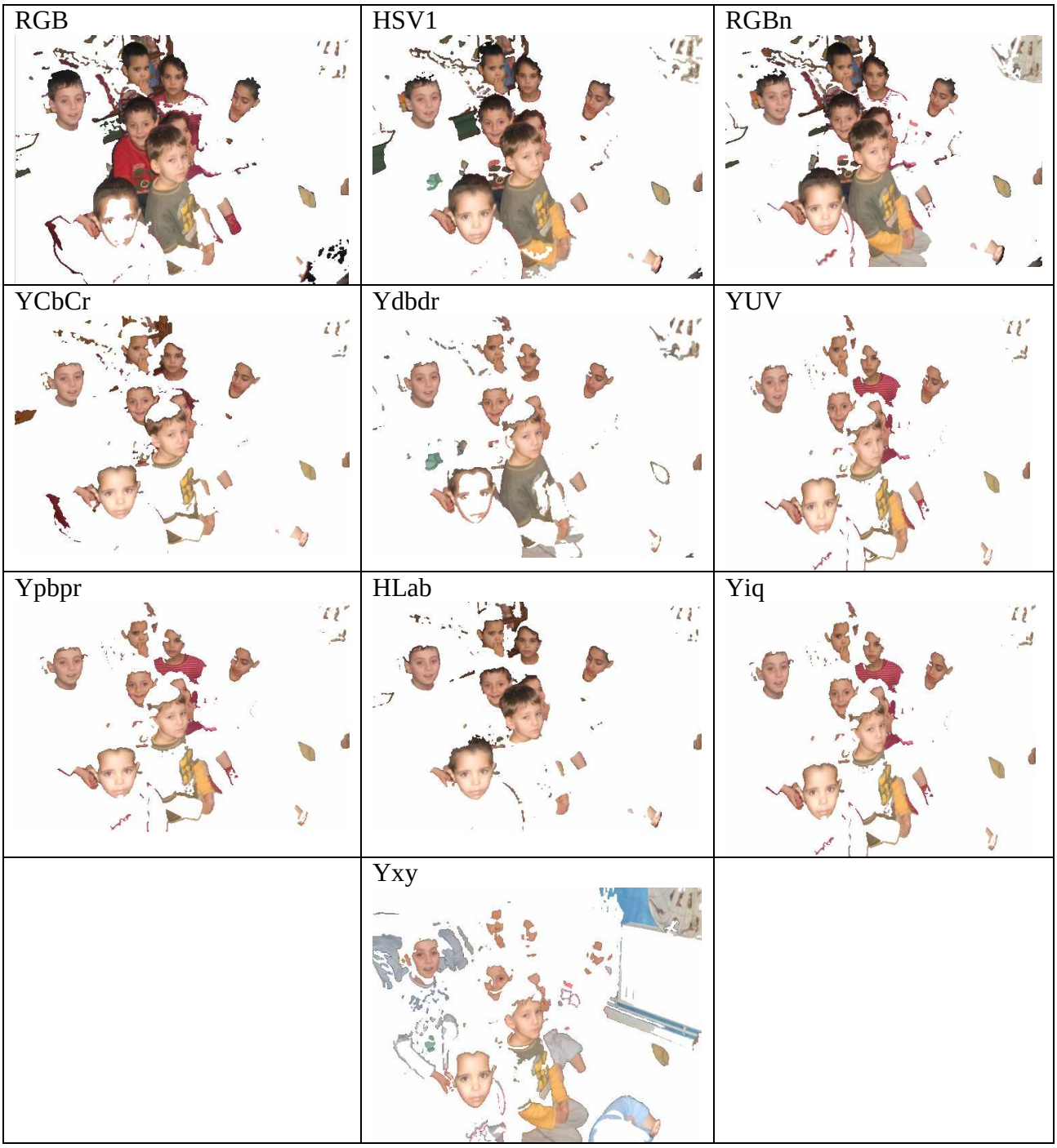


Figure 2: Segmentation de peau par modèle

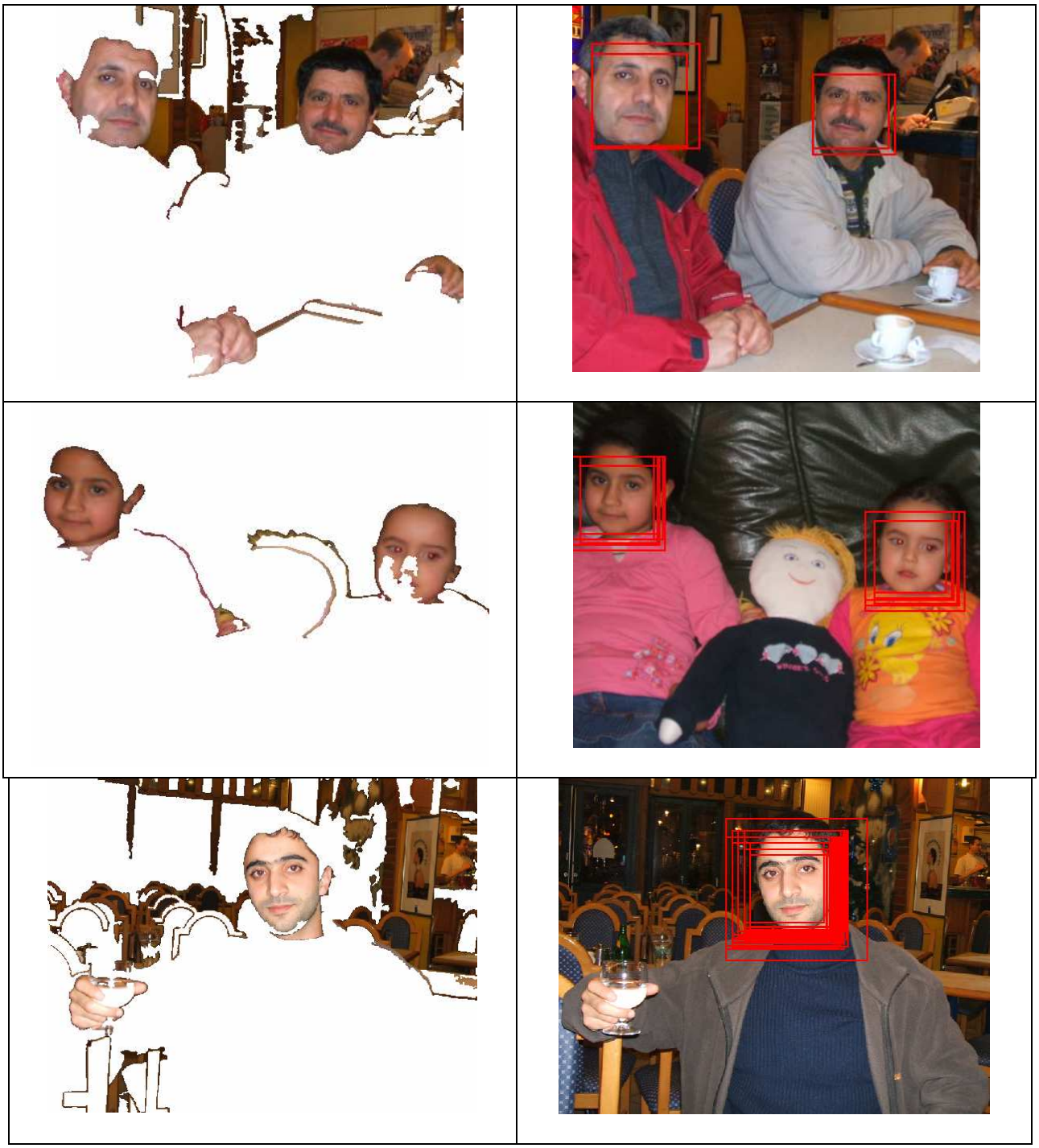


Figure 3(a) : Quelques Résultats de test, à gauche la détection de peau avec l'espace de couleur YCbCr, à droite la détection de visages

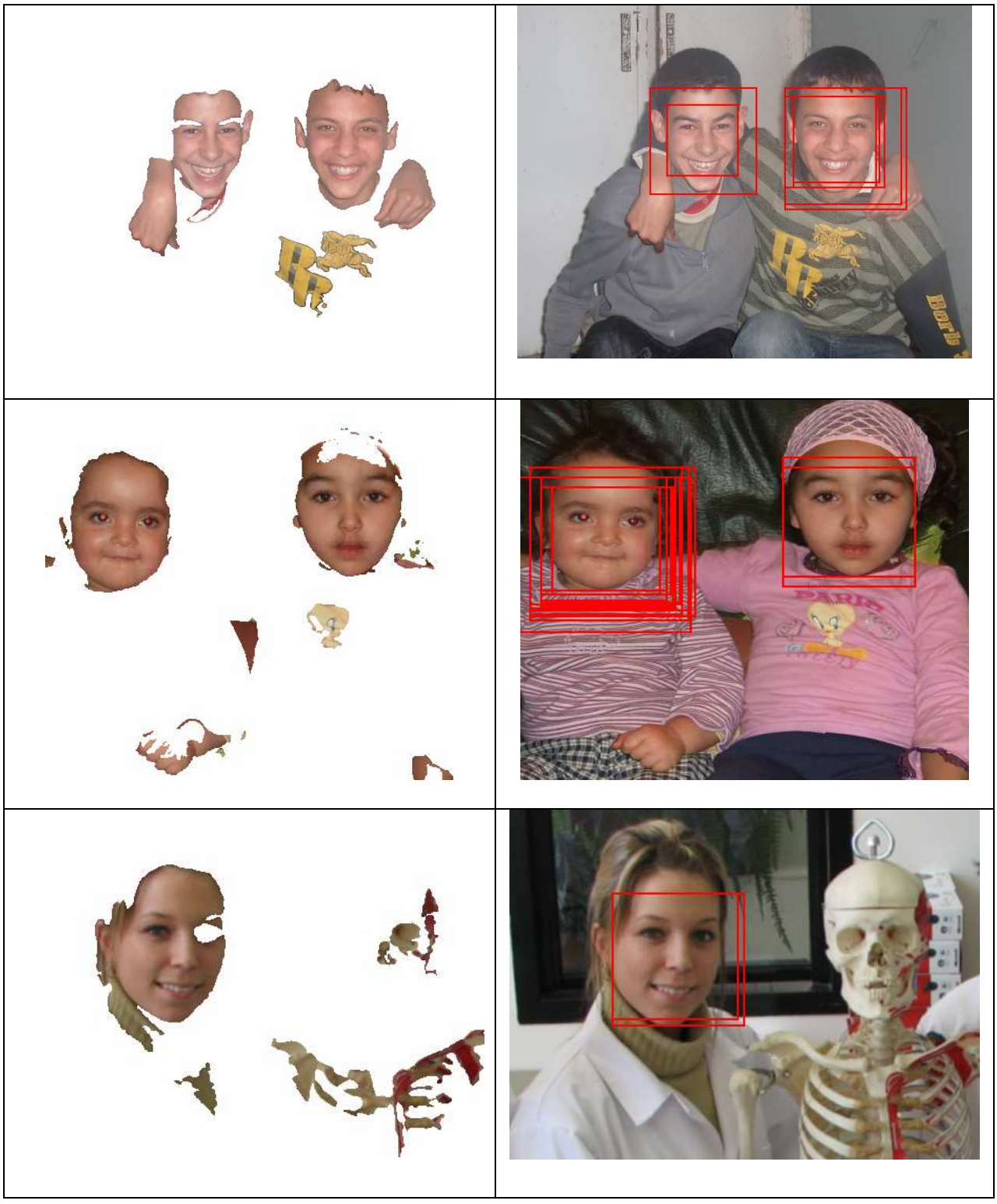


Figure 3(b) : Quelques Résultats de test, à gauche la détection de peau avec l'espace de couleur YCbCr, à droite la détection de visages

Annexe (B)

Construction de caractéristiques rectangulaires :

On a utilisé six caractéristiques

nI = 24 ; nJ = 24. c = 0, ou c est le compteur de nombre de caractéristiques crées.

Pour t = 1 :6 ; faire

Calcule des plus petites tailles de caractéristique en i et j

$S_{i0} = S_i * \text{int}(nMin / Si)$

$S_{j0} = S_j * \text{int}(nMin / S_j)$

Ou nMin=8 ; int est un opérateur pour mettre les valeurs réels en entiers.

On met nI = nI+1.et nJ = nJ+1.

Pour i=2 :nI ; pour j=2 :nJ faire :

$nI\_fin = \min(i + nMax, nI)$

$nJ\_fin = \min(j + nMax, nJ)$

Ou nMax =24.

Pour $i2 = i + S_{i0} - 1 : S_i : nI\_fin$

Pour $j2 = j + S_{j0} - 1 : Sj : nJ\_fin$

$iw = i2 - i + 1$
$jw = j2 - j + 1$

Compteur de nombre de caractéristiques : c = c + 1.

Calcule de la matrice creuse ‘F’ :

F = sparse([],[],[],625, 1,625);
$a = i2 + 25(j2 - 1)$
$b = (i - 1) + 25(j - 2)$
$c = (i - 1) + 25(j2 - 1)$
$d = i2 + 25(j - 2)$

F(a)=1 ;

F(b)=1 ;

F(c)=-1 ;

F(d)=-1 ;

Pour le 1e caractéristique  :

$S_i = 2; S_j = 1; iw = iw/2$

On calcule b1 et b2 :

b1 = F, avec : $i2 = iw - 1$

b2 = F, avec : $i = i + iw$

le vecteur caractéristique v est calculé comme suit :

$v = b1 - b2.$

Pour le 2e caractéristique  :

$S_i = 1; S_j = 2; jw = jw/2$

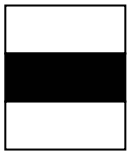
On calcule b1 et b2 :

b1 = F, avec : $j2 = j + jw - 1$

b2 = F, avec : $j = j + jw$

le vecteur caractéristique v est calculé comme suit :

$v = b1 - b2.$

Pour le 3e caractéristique  :

$S_i = 3; S_j = 1; iw = iw/3$

On calcule b1,b2 et b3 :

b1 = F, avec : $i2 = i + iw - 1$

b2 = F, avec : $i = i + iw$

$$i2 = i + 2(iw) - 1$$

b3 = F, avec : $i = i + 2 \cdot iw$

Le vecteur caractéristique v est calculé comme suit :

$v = b1 - b2 + b3.$

Pour le 4e caractéristique  :

$S_i = 1; S_j = 3; jw = jw/3$

On calcule b1,b2 et b3 :

b1 = F, avec : $j2 = j + jw - 1$

b2 = F, avec : $j = j + jw$

$$j2 = j + 2(jw) - 1$$

b3 = F, avec : $j = j + 2 \cdot jw$

Le vecteur caractéristique v est calculé comme suit :

$v = b1 - b2 + b3.$

Pour le 5e caractéristique  :

$S_i = 2; S_j = 2; iw = iw/2; jw = jw/2$

On calcule b1,b2,b3 et b4:

b1 = F, avec : $i2 = i + iw - 1$

$$j2 = j + jw - 1$$

b2 = F, avec : $i = i + iw$

$$j2 = j + jw - 1$$

b3 = F, avec : $j = j + jw$

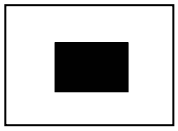
$$i2 = i + iw - 1$$

b4 = F, avec : $i = i + iw$

$$j = j + jw$$

Le vecteur caractéristique v est calculé comme suit :

$v = b1 - b2 - b3 + b4.$

Pour le 6e caractéristique  :

$S_i = 3; S_j = 3; iw = iw/2;\ jw = jw/2$

On calcule b1 et b2 :

b1 = F,

b2 = F, avec : $i = i + iw$

$$j = j + jw$$

$$i2 = i + 2 \cdot iw - 1$$

$$j2 = j + 2 \cdot jw - 1$$

le vecteur caractéristique v est calculé comme suit :

$v = b1 - 2 \cdot b2.$

Annexe (C)

Déroulement du programme pour calculer l'erreur et la mise à jour des pondérations pour chaque itération :

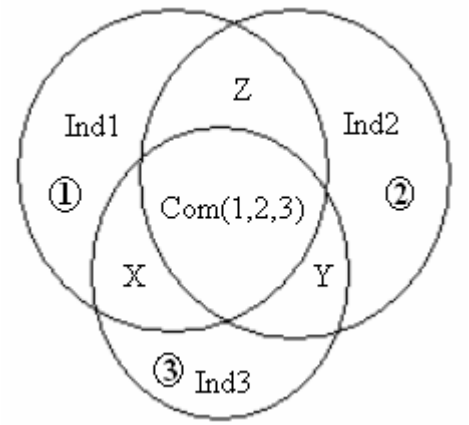


Figure 4 : Appartenance des exemples mal classifiés pour les trois premiers classifieurs

Ind : signifié le nb des exemples mal classifiés indépendant pour chaque classifieur.
Com(1,2,3) : signifié le nb des exemples mal classifiés commun pour les trois classifieurs.

Ind1=185.
Ind2=218.
Ind3=319.
Com(1,2,3)=20 .
X=59.
Y=49.
Z=66.
MC(1) = Ind1 + Com(1,2,3) + X + Z = 185 + 20 + 59 + 66 = 330 .
MC(2) = Ind2 + Com(1,2,3) + Y + Z = 218 + 20 + 49 + 66 = 353.
MC(3) = Ind3 + Com(1,2,3) + X + Y = 319 + 20 + 59 + 49 = 447.
CC1=1670.
CC2=1647.
CC3=1553.
Calcule de l'erreur de chaque classifieur :

A la 1er itération :

$W_{bien}=W_{mal}=0.0005$.
MC1=330.
CC1=1670.

$Erreur_{1\_1}= MC1\times W_{mal}$.
$Erreur_{1\_1}=330\times 0.0005=0.1650$.

$Erreur_{1\_2}= MC2\times W_{mal}$.
$Erreur_{1\_2}=353\times 0.0005=0.1765$.

$Erreur_{1\_3}= MC3\times W_{mal}$.

$Erreur_{1\_3}$= 447×0.0005=0.2235.

α_1 = (0.5)*log((1-minErreur)/ minErreur).
α_1 = (0.5)*log((1-$Erreur_{1\_1}$)/ $Erreur_{1\_1}$).
α_1 = 0.81074.

Mise à jour des pondérations :

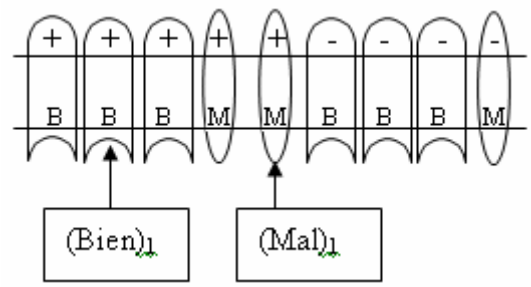


Figure 5: Cas possibles de classification incorrecte à la 1er itération

λ_{bien} = exp(-α_1) = 0.44452.
λ_{mal} = exp(+α_1) = 2.24957
$W(Bien)_1 = W_{bien} . \lambda_{bien}$ =0.00022226.
$W(Mal)_1 = W_{mal} . \lambda_{mal}$ =0.001124785.

Normalisation des poids :

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

$\sum w = (W(Bien)_1 \times nb(\text{bien classifiés})) + (W(Mal)_1 \times nb(\text{mal classifiés}))$.
$\sum w = (CC1 \times W(Bien)_1) + (MC1 \times W(Mal)_1)$.
$= (1670 \times 0.00022226) + (330 \times 0.001124785)$.
$= 0.3711742 + 0.37117905$.
$= 0.74235325$.

$$W(Bien)_1 \leftarrow \frac{W(Bien)_1}{\sum w}$$

$$W(Bien)_1 \leftarrow \frac{0.00022226}{0.74235325}$$

$$W(Bien)_1 = 0.000299399$$

$$W(Mal)_1 \leftarrow \frac{W(Mal)_1}{\sum w}$$

$$W(Mal)_1 \leftarrow \frac{0.001124785}{0.74235325}$$

$W(Mal)_1 = 0.001515161$

A la 2ème itération :

Les valeurs de $W(Bien)_1$ et $W(Mal)_1$ calculés par Matlab :
$W(Bien)_1$= 0.0002994.
$W(Mal)_1$= 0.001515147.
MC2=353.
CC2=1647.

$Erreur_{2\_1}$= MC1× $W(Mal)_1$.
$Erreur_{2\_1}$= 330×0.001515147 =0.49999851.

$Erreur_{2\_2}$= (Z+Com(1,2,3)) × $W(Mal)_1$+(Ind2+Y) × $W(Bien)_1$.
$Erreur_{2\_2}$= (66+20) × 0.001515147 +(218+49) × 0.0002994.
$Erreur_{2\_2}$= 86 × 0.001515147 + 267 × 0.0002994.
$Erreur_{2\_2}$= 0.130302642 + 0.0799398.
$Erreur_{2\_2}$=0.210242442.

$Erreur_{2\_3}$= ((X+Com(1,2,3))× $W(Mal)_1$)+((Y+Ind3) × $W(Bien)_1$).
$Erreur_{2\_3}$= ((59+20)× 0.001515147)+((49+319) × 0.0002994).
$Erreur_{2\_3}$= (79× 0.001515147)+(368× 0.0002994).
$Erreur_{2\_3}$= 0.119696613+0.1090752.
$Erreur_{2\_3}$= 0.228771813.

α_2 = (0.5)*log((1-minErreur)/ minErreur).
α_2 = (0.5)*log((1- $Erreur_{2\_2}$)/ $Erreur_{2\_2}$).
α_2 = 0,66173.

Mise à jour des pondérations :

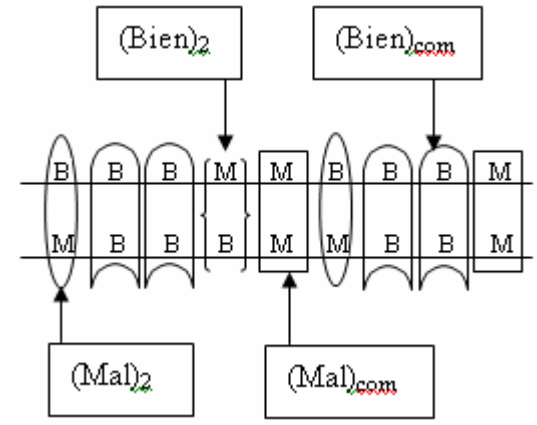


Figure 6 : Cas possibles de classification incorrecte à la 2ème itération

$\lambda_{bien} = \exp(-\alpha_1)$ =0.515957954.

$\lambda_{mal} = \exp(+\alpha_1)$ = 1.938142422.

$W(Bien)_2 = W(Mal)_1 \times \lambda_{bien}$.

= 0.001515147×0.515905329.

= 0.000781752.

$W(Bien)_{com} = W(Bien)_1 \times \lambda_{bien}$.

=0.0002994×0.515905329.

=0.000154477.

$W(Mal)_2 = W(Bien)_1 \times \lambda_{mal}$.

= 0.0002994×1.938340123.

= 0.000580279.

$W(Mal)_{com} = W(Mal)_1 \times \lambda_{mal}$.

=0.001515147×1.938340123.

=0.00293657.

Normalisation des poids :

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

$\sum w = (W(Bien)_2 \times (Ind1+X)) + (W(Bien)_{com} \times (CC2 - (Ind1+X)) + (W(Mal)_2 \times (Ind2+Y)) + (W(Mal)_{com} \times (Z+Com(1,2,3))$.

$\sum w = (0.000781752 \times (185+59)) + (0.000154477 \times (1647-(185+59))) + (0.000580279 \times (218+49)) + (0.00293657 \times (66+20))$.

$\sum w = (0.000781752 \times 244) + (0.000154477 \times 1403) + (0.000580279 \times 267) + (0.00293657 \times 86)$.

$\sum w = 0.190747488 + 0.216731231 + 0.154934493 + 0.25254502$.

$\sum w = 0.814958232$.

$$W(Bien)_2 \leftarrow \frac{W(Bien)_2}{\sum w}$$

$$W(Bien)_2 \leftarrow \frac{0.000781752}{0.814958232}$$

$W(Bien)_2 = 0.000959254$.

$$W(Bien)_{com} \leftarrow \frac{W(Bien)_{com}}{\sum w}$$

$$W(Bien)_{com} \leftarrow \frac{0.000154477}{0.814958232}$$

$W(Bien)_{com} = 0.000189552$.

$$W(Mal)_2 \leftarrow \frac{W(Mal)_2}{\sum w}$$
$$W(Mal)_2 \leftarrow \frac{0.000580279}{0.814958232}$$
$$W(Mal)_2 = 0.000712035.$$

$$W(Mal)_{com} \leftarrow \frac{W(Mal)_{com}}{\sum w}$$
$$W(Mal)_{com} \leftarrow \frac{0.00293657}{0.814958232}$$
$$W(Mal)_{com} = 0.003603338.$$

A la 3ème itération :

Les valeurs des pondérations des exemples d'apprentissage calculés par Matlab :
$W(Bien)_2$= 0.00095925.
$W(Bien)_{com}$=0.00018955.
$W(Mal)_2$= 0.00071204.
$W(Mal)_{com}$=0.0036.
MC3=447.
CC3=1553.

$Erreur_{3\_1}$= ((MC1-(Z+Com(1,2,3)))×$W(Bien)_2$)+((Z+Com(1,2,3))×$W(Mal)_{com}$).
$Erreur_{3\_1}$= ((330-(66+20))× 0.00095925)+((66+20)× 0.0036).
$Erreur_{3\_1}$= (244× 0.00095925)+(86× 0.0036).
$Erreur_{3\_1}$= 0.234057+0.3096.
$Erreur_{3\_1}$=0.543657.

$Erreur_{3\_2}$= ((MC2-(Z+Com(1,2,3)))×$W(Mal)_2$)+((Z+Com(1,2,3))×$W(Mal)_{com}$).
$Erreur_{3\_2}$= ((353-(66+20))× 0.00071204)+((66+20)× 0.0036).
$Erreur_{3\_2}$= (267× 0.00071204)+(86× 0.0036).
$Erreur_{3\_2}$= 0.19011468+0.3096.
$Erreur_{3\_2}$=0.49971468.

$Erreur_{3\_3}$= Ind3 × $W(Bien)_{com}$ +Com(1,2,3)× $W(Mal)_{com}$ + X × $W(Bien)_2$ + Y× $W(Mal)_2$.
$Erreur_{3\_3}$= 319×0.00018955 + 20×0.0036 + 59×0.00095925 + 49×0.00071204.
$Erreur_{3\_3}$= 0.06046645 + 0.072 + 0.05659575 + 0.03488996.
$Erreur_{3\_3}$ = 0.22395216.

α_3 = (0.5)*log((1-minErreur)/ minErreur).
α_3 = (0.5)*log((1-$Erreur_{3\_3}$)/ $Erreur_{3\_3}$);
α_3 =0,621390855.

	1er itération	2ème itération	3ème itération
1er classifieur	0.1650	0.49999851	0.543657
2ème classifieur	0.1765	0.210242442	0.49971468
3ème classifieur	0.2235	0.228771813	0.22395216

Tableau 1: Evolution de l'erreur de chaque classifieur pour les trois premières itérations