LOCALISATION DE VISAGES

Projet de Diplôme

Fabrice Vermont

Assistant: Julien Meynet
Professeur: Prof. Jean-Philippe Thiran
Lausanne, Février 2005

Résumé

Dans ce rapport, une méthode pour la localisation de visages est présentée. La localisation de visages est une des composantes importantes dans l'analyse et la compréhension de visages. Les méthodes existantes pour localiser des visages peuvent être subdivisées en méthodes image-based et méthodes feature-based. Le programme implémenté utilise une méthode image-based. Pour entraîner les classificateurs, nous avons utilisé un algorithme de boosting (AdaBoost) qui permet d'avoir une bonne performance de détection. AdaBoost est un algorithme d'apprentissage agressif qui construit un classificateur fort à partir d'un ensemble de classificateurs faibles. Comme classificateurs faibles, des classificateurs linéaires ont été utilisés. Chaque classificateur linéaire utilise un descripteur, basé sur des masques visuels qui sont une variante d'ondelette de Haar modulé par une Gaussienne. Les classificateurs ont été entraînés et testés sur la base de données classique BANCA.
Pour la détection des objets caractéristiques, une technique de fenêtre glissante à taille variable a été employée. Pour trouver à partir de toutes les détections retournées par les classificateurs les positions les plus probables des objets caractéristiques, trois méthodes d'arbitrage ont été implémentées et testées sur la base de données classique BANCA.

Table des matières

Chapitre 1 4
Introduction 4
1.1 Contexte 4
1.2 Histoire 5
1.3 Difficultés de la localisation de visages par une machine 6
1.4 Les deux approches principales 8
1.5 Critères utilisés pour mesurer la performance dans la localisation de visages 10
1.6 Approche 14
Chapitre 2 15
État de l'art 15
2.1 Introduction 15
2.2 Méthodes Image-Based 15
2.2.1 Introduction 15
2.2.2 Méthodes statistiques réduisant l'espace des descripteurs 17
2.2.2.1 Introduction 17
2.2.2.2 Principal Component Analysis (PCA) 18
2.2.2.3 Linear Discriminant Analysis (LDA) – Fisher's Linear Discriminant (FLD) 19
2.2.3 Méthodes « entraînant des classificateurs » statistiques 20
2.2.3.1 Machines à Vecteurs de Support (SVM) 20
2.2.3.2 Réseaux de Neurones 21
2.2.3.3 Hidden Markov Models (HMM) 21
2.3 Méthodes Feature-Based 21
2.3.1 Introduction 21
2.3.2 Analyse bas niveau 22
2.3.3 Analyse des objets caractéristiques 22
2.3.4 Active Shape Models 23
Chapitre 3 24
Méthode pour la localisation de visages 24
3.1 Introduction 24
3.1.1 Objectif 24
3.1.2 Choix de la méthode 25
3.2 Les descripteurs 25
3.2.1 Introduction 25
3.2.2 Choix des descripteurs 26
3.2.3 Les masques visuels 27
3.3 Les classificateurs faibles 30
3.3.1 Introduction 30
3.3.2 Algorithme d'entraînement 31
3.3.3 Définition d'un classificateur faible 33
3.3.4 Pourquoi des classificateurs linéaires 34
3.4 AdaBoost 35
3.4.1 Introduction 35
3.4.2 Idées de base du Boosting 35
3.4.2 Algorithme 38
3.4.4 La convergence de l'erreur d'entraînement vers zéro 40
3.5 Arbitrage 43
3.5.1 Introduction 43
3.5.2 Première méthode pour arbitrer 43
3.5.3 Deuxième méthode pour arbitrer 43

3.5.4 Troisième méthode pour arbitrer 43
4 Résultats et Expérimentations 46
4.1 Introduction 46
4.2 Base de données utilisée 47
4.3 Taille idéale des images 50
4.4 Résultats de la performance des classificateurs....52
4.5 Résultats de localisation....54
4.5.1 Localisation avec la première méthode pour arbitrer....55
4.5.2 Localisation avec la deuxième méthode pour arbitrer 57
4.5.3 Localisation avec la troisième méthode pour arbitrer 60
4.5.4 Temps d'exécution du programme 63
5 Conclusion et travaux futurs 64
5.1 Conclusion....64
5.2 Travaux futurs 65
Bibliographie....66

Chapitre 1

Introduction

1.1 Contexte

Dans ce rapport, une méthode pour localiser automatiquement des visages par machine est présentée. La localisation de visages est, avec la détection de visages, la reconnaissance de visages, la reconnaissance d'expressions faciales et la poursuite de visages, une des composantes importantes de l'analyse et la compréhension de visages. Des applications se trouvent dans le domaine du divertissement, des cartes intelligentes, de la sécurité des informations, des droits d'accès ou encore de la surveillance (TAB. 1.1).

Domaines	Applications spécifiques
Divertissement	jeux vidéo, réalité virtuelle,
	Interactions humain-robot, interactions humain-ordinateur
Cartes intelligentes	permis de conduire, programmes d'autorisation,
	Immigration, ID national, passeports, registration des électeurs
Sécurité d'informations	s'enregistrer sur une installation personnelle,
	Applications de sécurité, sécurité de bases de données,
	cryptage de fichiers, sécurité de l'Intranet,
	accès à l'Internet, notes médicales
Droit d'accès et surveillance	vidéo surveillance avancée, CCTV control
	contrôle d'accès, analyse d'évènements,
	vole, poursuite de suspects et investigation

TAB. 1.1 - Applications pour lesquelles, enter autre, la localisation de visages par machine est utilisée.

Le but de la localisation de visages, qui est souvent précédée par la détection de visages, est de localiser précisément la position du visage dans une image ou une séquence de vidéo d'une scène. Actuellement, il existe déjà un grand nombre de méthodes qui permettent la localisation de visages. Les méthodes les plus importantes seront présentées plus tard dans ce rapport.
La figure suivante (FIG. 1.1) donne une idée de comment la localisation de visages est liée avec les autres étapes importantes dans le processus de l'analyse et la compréhension de visages par machine. Elle est suivie par un commentaire bref sur les autres étapes importantes.

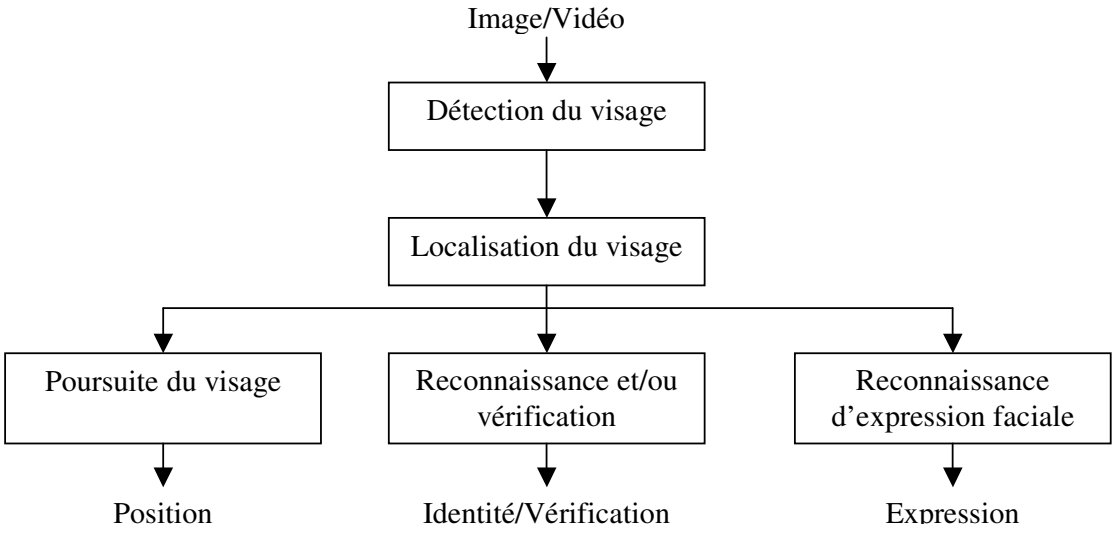

FIG. 1.1 – Liaison des différentes étapes importantes dans le processus de l'analyse de visages par machine.

- La détection de visages [2] vérifie la présence d'un ou plusieurs visages dans une image. Pour chaque visage détecté, la position approximative est retournée.

- La reconnaissance et/ou vérification de visages [1] vise à identifier la personne dans une image. Elle retourne l'identité, dans le cas de la reconnaissance, ou les droits d'accès, dans le cas de la vérification, de la personne.

- La poursuite de visages essaie de suivre un visage dans une séquence d'images. Elle retourne la position du visage dans l'image.

- La reconnaissance d'expressions faciales cherche à reconnaître l'état émotionnel de la personne (triste, heureux, de mauvaise humeur, etc.)

1.2 Histoire

La recherche dans le domaine de l'analyse de visages par une machine commence dans les années 70. Elle attire des chercheurs de plusieurs disciplines comme :

- le traitement des signaux,
- le réseau de neurones,
- la reconnaissance des formes,
- et la psychologie.

Des recherches extensives sont faites sur différents aspects de l'analyse de visages par un humain et une machine. Les chercheurs traitent habituellement l'analyse de visages comme un problème d'analyse d'un modèle 2D. Typiquement, ils appliquent des algorithmes de classification sur des descripteurs qui utilisent des attributs mesurés dans le visage de face ou de profil, comme par exemple la distance entre des objets caractéristiques.

Dans les années 80, la communauté scientifique perd l'intérêt dans l'analyse de visages par une machine. Elle la retrouve début des années 90 dû

- aux opportunités commerciales qui augmentent,
- au real-time hardware qui devient disponible,
- et à l'accroissement de l'importance des applications reliées à la surveillance.

Depuis, la recherche dans le domaine de l'analyse des visages par une machine s'est concentrée à automatiser le processus de la reconnaissance de visages, en se concentrant sur des problèmes comme la localisation de visages et l'extraction d'objets caractéristiques dans une image ou une séquence de vidéo.
Actuellement, à part quelques exceptions, la reconnaissance de visages par une machine est traitée comme un problème dans lequel il s'agit de trouver des objets 3D dans des images 2D.

1.3 Difficultés de la localisation de visages par une machine

Lors de la localisation de visages par une machine, on veut localiser précisément la position d'un visage dans une image ou une séquence de vidéo. Pour cela, on peut chercher à localiser des objets caractéristiques d'un visage qui permettent, selon leur positionnement, de tirer des conclusions sur le positionnement du visage. Par exemple :

- si l'œil droit du visage se trouve plus bas que l'œil gauche, alors la tête de la personne est inclinée sur le côté droit,
- si les deux yeux du visage sont trop près l'un de l'autre, alors le visage n'est pas vu de face et donc la tête de la personne est tournée d'un certain nombre de degrés sur le côté. Une bonne façon pour trouver le côté sur lequel la tête est tournée est de faire des tests sur l'apparence du nez, comme celui-ci est un objet qui sort significativement du plan du visage.

Les objets caractéristiques d'un visage sont, par exemple, l'œil droit, l'œil gauche, le nez, la bouche, etc.
La difficulté est que les objets caractéristiques ne sont pas rigides. Ils peuvent être vus comme des objets facilement déformables ou comme des objets avec une grande variance intra-classe. L'apparence peut varier d'une personne à l'autre ou simplement entre deux images prises de la même personne. Les causes pour la grande variance intra-classe des objets caractéristiques sont :

- Le changement de l'apparence physique entre les personnes : les yeux peuvent êtres plutôt ronds ou minces, le nez peut être grand ou petit, pointu ou émoussé, la bouche peut être grande ou petite, la couleur de la peau peut être foncé ou claire, etc.
- Le changement de la disposition d'esprit : lorsqu'une personne est étonnée, elle peut avoir la bouche ouverte et faire de grands yeux. Si par contre elle est fâché, elle va avoir un regard sérieux et la bouche fermée.
- Le positionnement du visage : le visage avec ses objets caractéristiques peut être vu de face, de profil ou dans une position intermédiaire.
- Les objets qui modifient ou masquent les objets caractéristiques : une barbe ou une moustache modifie l'apparence de la région autour de la bouche, des cheveux ou une paire de lunettes peuvent cacher des parties des yeux, etc.
- La qualité de l'image : les objets caractéristiques changent leur apparence en fonction du type et de l'intensité d'éclairage, de la qualité du système d'acquisition et d'autre bruit.

Les images suivantes illustrent des causes pour la grande variabilité intra-classe des objets caractéristiques.

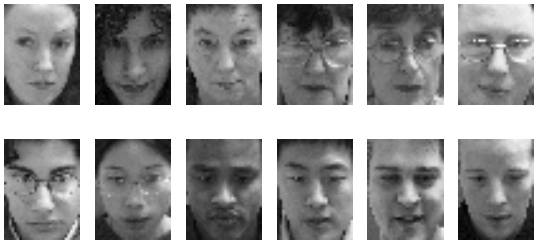

FIG. 1.2 – Images illustrant des causes pour la grande variabilité intra-classe des objets caractéristiques. Les images sont tirées de la base de données BANCA.

1.4 Les deux approches principales

Il y a différentes méthodes pour localiser des objets caractéristiques du visage qui essaient de manipuler les inconvénients décrits dans le chapitre 1.3. Les plus importantes d'entre elles sont décrites dans le chapitre 2. Au fond, les différentes méthodes utilisent toutes une des deux approches principales ou une combinaison de celles-ci. Les deux approches principales sont :

1. L'approche basée sur des images exemples, aussi appelée *« image-based approach »*. Cette approche apprend à reconnaître un objet caractéristique sur la base d'images représentant l'objet caractéristique (images positives) et d'images ne représentant pas l'objet caractéristique (images négatives).

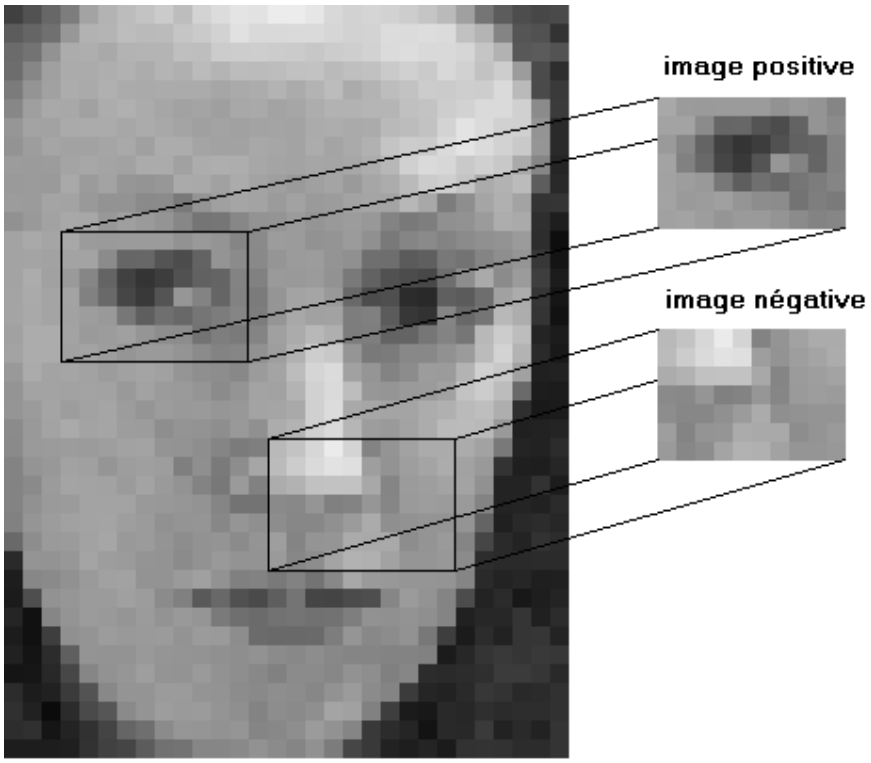

FIG. 1.3 – Image illustrant une image positive et négative d'un objet caractéristique qui est dans ce cas l'œil droit du visage.

2. L'approche utilisant des connaissances sur l'objet caractéristique à localiser, aussi appelée *« feature-based approach »* ou *« geometrical-based approach »*. Cette approche se sert par exemple des connaissances à priori sur la géométrie de l'objet caractéristique ou le positionnement de l'objet caractéristique par rapport à d'autres objets caractéristiques.

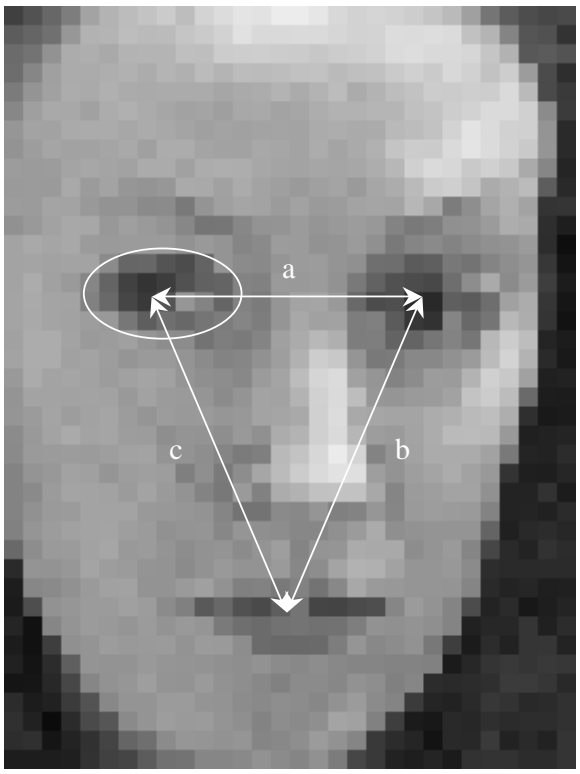

FIG. 1.4 – Image illustrant la configuration géométrique d'un ensemble d'objets caractéristiques qui sont dans ce cas l'œil droit, l'œil gauche et la bouche du visage. Elle montre aussi l'œil droit qui est modélisé par une ellipse.

Les premières approches traitaient des visages vus de face et se concentraient sur chaque objet caractéristique individuellement (œil droit, nez, bouche, etc.). Les méthodes utilisaient l'approche feature-based et modélisaient seulement la géométrie de l'objet caractéristique à extraire. Ils avaient des difficultés lorsque l'apparence de l'objet caractéristique changeait significativement, par exemple œil fermé, œil masqué par lunette, bouche ouverte, bruit (par exemple éclairage) etc. Pour augmenter la fiabilité des détections, les approches feature-based plus récentes utilisent normalement, en plus de l'information sur la géométrie des objets caractéristiques du visage, aussi de l'information sur la configuration géométrique des objets caractéristiques. Les méthodes utilisant de l'information sur la configuration géométrique sont plus robustes lors de variations dans l'intensité de l'image et de la forme des objets caractéristiques.
Aujourd'hui, la méthode la plus utilisée est la méthode image-based. La méthode image-based est plus robuste aux changements d'apparence des objets caractéristiques et les influences des conditions environnementales.
Le tableau suivant (TAB. 1.2) présente une sous-classification possible des deux approches principales et des méthodes qui correspondent à ces sous-classes.

Classification Principale	Sous-Classification	Méthodes
Image-Based	Méthodes statistiques réduisant l'espace des descripteurs	PCA [5], Fisher's Linear Discriminant [5], FA [5]
	Méthodes entraînant des classificateurs statistiques	SVM [5], Réseaux de Neurones [8], AdaBoost [4], Classificateurs Bayésiens [12], Perceptron [12], Clustering [12]
Feature-Based	Analyse de bas niveau	Seuillage [12], Filtrage [12], Ouverture [12], Fermeture [12], (Segmentation Classique)
	Analyse des objets caractéristiques	Analyse de la constellation [2], chercher après des objets caractéristiques [2]
	Active Shape Models	Contour Actif [2], Masques Déformables [2], PDM's [2]

TAB. 1.2 – Sous-classification possible des deux approches principales.

1.5 Critères utilisés pour mesurer la performance dans la localisation de visages

Dans l'analyse et la compréhension d'images par machines, il est usuel d'utiliser les critères comme le *taux des détections* d_p (detection rate), le *taux des mauvaises détections négatives* f_n (false negative rate) et le *taux des mauvaises détections positives* f_p (false positive rate) pour mesurer les performances d'un algorithme. La localisation de visages, étant un sous-domaine de l'analyse et la compréhension d'images, utilise aussi ces critères. La suite donne une définition brève de ces critères dans le cas de la localisation de visages.

Détection positive
On appelle « une détection positive » une fenêtre dans l'image qui, selon le détecteur, contient un objet caractéristique.

Détection négative
Réciproquement, on appelle « une détection négative » une fenêtre dans l'image qui, selon le détecteur, ne contient pas d'objet caractéristique.

Le taux des (bonnes) détections (positives)
Le *taux des détections* d_p est le pourcentage des objets caractéristiques pour lesquels on a une détection positive dans une série d'images.

Le taux des mauvaises détections négatives

Le *taux des mauvaises détections négatives* f_n est le pourcentage des objets caractéristiques pour lesquels on a une détection négative, dans une série d'images. La relation entre le *taux des détections* d et le *taux des mauvaises détections négatives* f_n est :

$$f_n = 1 - d_p \tag{1.1}$$

Le taux des bonnes détections négatives

Le *taux des bonnes détections négatives* d_n est le pourcentage des régions qui ne contiennent pas d'objet caractéristique pour lesquelles on a une détection négative dans une série d'images.

Le taux des mauvaises détections positives

Le *taux des mauvaises détections positives* f_p est le pourcentage des régions qui ne contiennent pas d'objet caractéristique pour lesquelles on a une détection positive dans une série d'images. La relation entre le *taux des bonnes détections négatives* et le *taux des mauvaises détections positives* est :

$$f_p = 1 - d_n \tag{1.2}$$

Dans les deux figures suivantes (FIG. 1.5 et 1.6), les différents taux sont illustrés. Les deux figures montrent deux distributions (histogrammes) possible d'un descripteur A_1. Les deux distributions sont calculées à partir d'une base de données annotée qui contient des échantillons positifs (régions d'images contenant un objet caractéristique et labellisées avec 1) et négatives (régions d'images ne contenant pas d'objet caractéristique et labellisées avec -1). Sur les figures se trouvent aussi le seuil de décision Θ qui est utilisé par le classificateur linéaire $h(A_1)$ pour réaliser la classification.

$$h(A_1) = \begin{cases} 1 \text{ si } A_1 > \Theta \\ -1 \text{ si } A_1 < \Theta \end{cases} \tag{1.3}$$

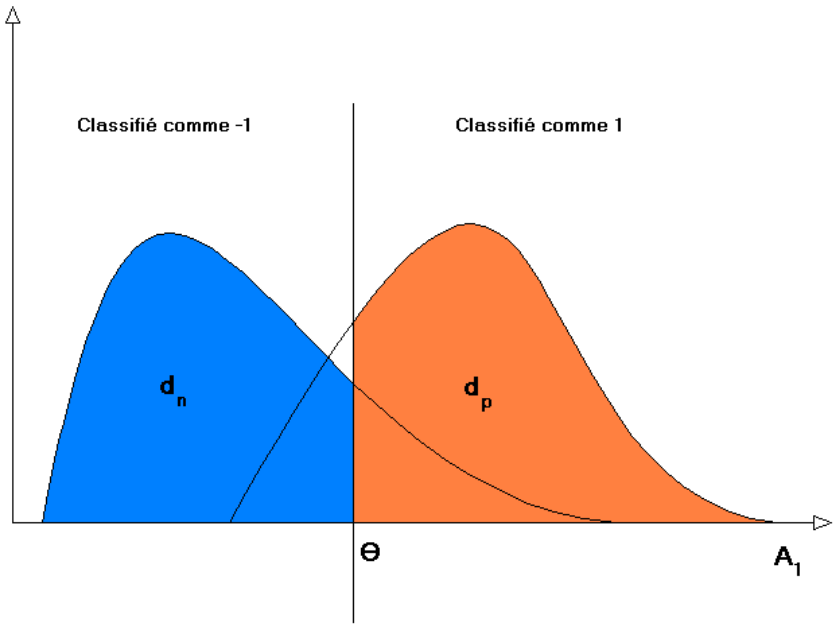

FIG. 1.5 – Image illustrant le taux des bonnes détections positives d_p et le taux des bonnes détections négatives d_n en fonction du seuil de classification Θ.

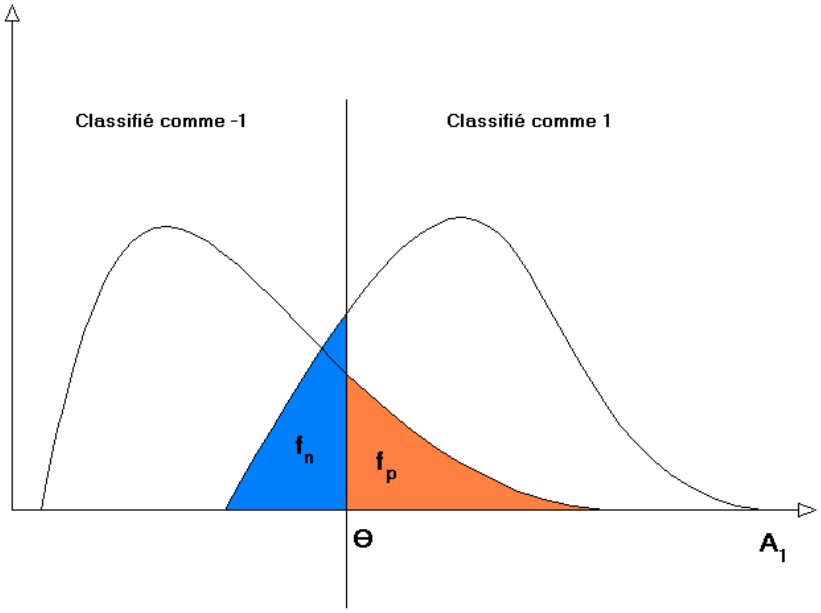

FIG. 1.6 – Image illustrant le taux des mauvaises détections positives f_p et le taux des mauvaises détections négatives f_n en fonction du seuil de classification Θ.

L'objectif de la localisation de visages est de maximiser le taux des bonnes détections positives d_p et de minimiser le taux des mauvaises détections positives f_p. Au cas des deux figures précédentes, on voit bien qu'il est difficile de satisfaire aux deux contraintes en même temps, car souvent les distributions des deux classes sont superposées partiellement. Il s'agit de trouver un compromis entre le taux des bonnes détections positives d_p et le taux des mauvaises détections positives f_p.

Une courbe souvent utilisée et illustrant très bien la difficulté de satisfaire les deux contraintes mentionnées précédemment s'appelle ROC (Receiver Operating Characteristics). Cette courbe décrit le taux des mauvaises détections négatives f_n en fonction du taux des mauvaises détections positives f_p lorsqu'on fait varier le seuil Θ. La figure suivante (FIG. 1.7) illustre une courbe ROC typique.

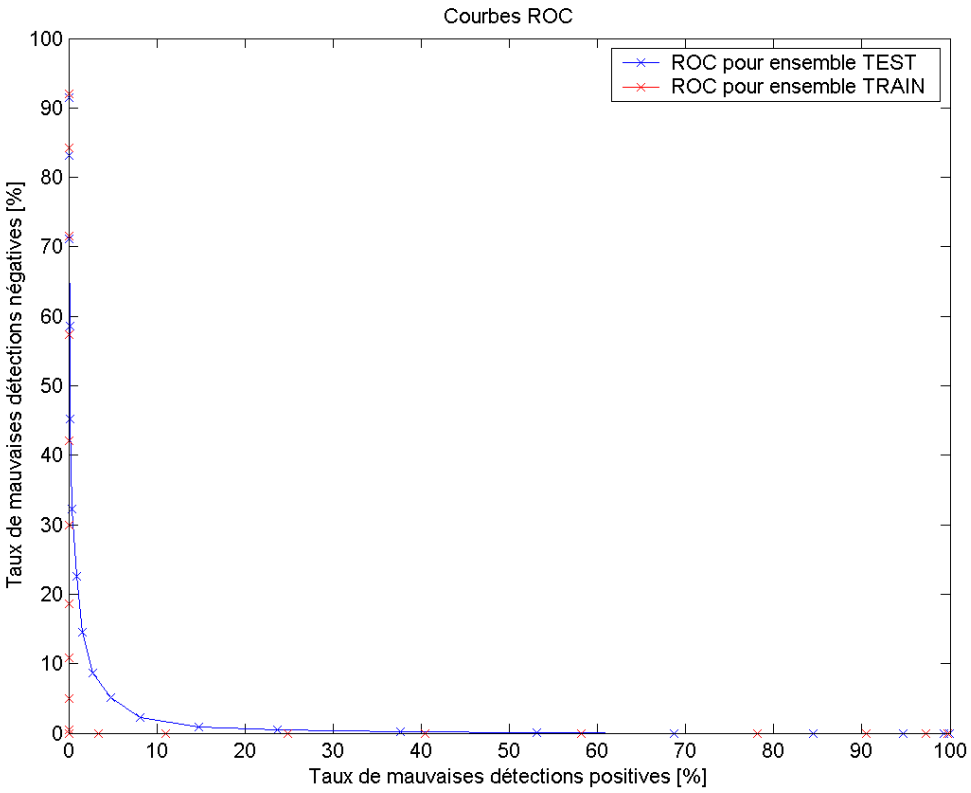

FIG. 1.7 – Courbes ROC. Elle illustre le taux des mauvaises détections négatives f_n en fonction du taux des mauvaises détections positives f_p lorsqu'on fait varier le seuil Θ.

On voit bien que si on choisit d'avoir un taux des mauvaises détections négatives f_n bas, alors le taux des mauvaises détections positives sera grand et réciproquement.

1.6 Approche

La méthode présentée dans ce rapport est basée sur un algorithme qui permet de booster la performance de classification d'un simple classificateur. L'algorithme s'appelle AdaBoost et a été présenté la première fois en 1996 par Freund & Schapire. Comme classificateurs simples, un ensemble de classificateurs linéaires est utilisé. Chaque classificateur simple utilise un descripteur construit à partir d'un masque visuel. Les masques visuels sont une variante d'ondelette de Haar modulée par une Gaussienne. Pour trouver la position la plus probable des objets caractéristiques à partir de toutes les détections retournées par les classificateurs, trois méthodes pour faire un arbitrage sont présentées.

Chapitre 2

État de l'art

2.1 Introduction

Dans ce chapitre, différentes méthodes pour extraire des objets caractéristiques sont présentées. Elles sont divisées en méthodes *image-based* et méthodes *feature-based* (TAB. 1.2). Lorsque le visage est vu de face, les méthodes image-based obtiennent normalement, avec leur approche statistique, des meilleurs résultats de détection que les méthodes feature-based. Par contre, si le visage est vu de profil ou légèrement tourné de côté, alors des méthodes feature-based atteignent des meilleurs résultats de détection. Ceci est dû à la nature géométrique des méthodes featrure-based qui les rend plus robuste par rapport à d'éventuelles rotations.

2.2 Méthodes Image-Based

2.2.1 Introduction

Les méthodes Image-Based sont construites à partir d'un ensemble de M images d'apprentissage. Contrairement aux méthodes Feature-Based, elles ne nécessitent pas des connaissances préalables sur l'objet caractéristique à localiser dans le visage. Les images exemples forment une base de données déjà annotée. Dans le cas de la localisation du visage, la base de données consiste souvent d'*images positives* et d'*images négatives* pour chaque objet caractéristique à localiser. Une image positive est une image qui représente l'objet caractéristique à localiser et une image négative est une image qui ne représente pas l'objet caractéristique à localiser.
Dans l'approche image-based, il y a deux catégories de méthodes :

- Les méthodes réduisant l'espace des descripteurs
- Les méthodes entraînant des classificateurs statistiques

Un algorithme de localisation de visages basé sur l'approche image-based peut être construit en combinaison des méthodes des deux catégories ou en utilisant seulement une méthode de la deuxième catégorie.
Lors de la localisation, la majorité des approches image-based utilisent une fenêtre glissante qui parcourt toute l'image du visage (FIG. 2.1). À chaque position, la partie de l'image qui est enfermée par la fenêtre est extraite et classifiée. De telle façon une recherche après la position possible de l'objet caractéristique est faite.

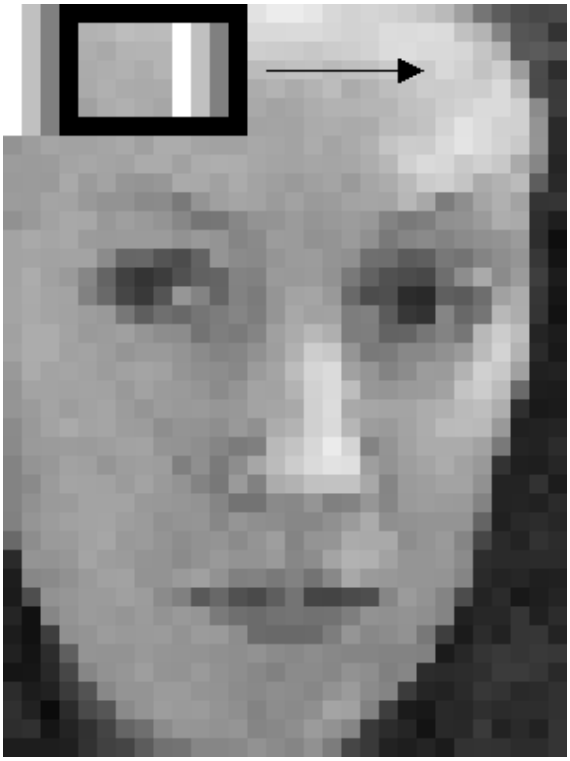

FIG. 2.1 – Image illustrant la fenêtre qu'on fait clisser à travers toute l'image. Elle à commencé son parcours en haut à gauche et continue à glisser sur le côté droit.

Les méthodes image-based considèrent l'image comme une *variable aléatoire discrète multidimensionnelle*

$$\mathbf{X}_i = \begin{pmatrix} X_{i,1} \\ X_{i,2} \\ . \\ . \\ X_{i,N} \end{pmatrix} \tag{2.1}$$

avec N le nombre de pixels dans l'image (l'image est mise sous forme vectorielle). N est aussi appelé la dimension de l'espace des images. La variable aléatoire $\mathbf{X}_i$ est multidimensionnelle par-ce que chaque pixel est représenté par une dimension et elle est discrète par-ce que un pixel permet seulement un nombre fini de valeurs (256 valeurs de luminance/pixel pour une image à 8 bits/pixel).
Les M images exemples sont notées

$$\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, ..., \mathbf{X}_M \tag{2.2}$$

et forment un échantillon de la population des images positives, négatives ou positives et négatives, selon ce qu'on aura besoin.

2.2.2 Méthodes statistiques réduisant l'espace des descripteurs

2.2.2.1 Introduction

Souvent, la dimension de l'espace des images est très élevée et l'espace des images positives et négatives occupe seulement un sous-espace de l'espace des images (FIG. 2.2).

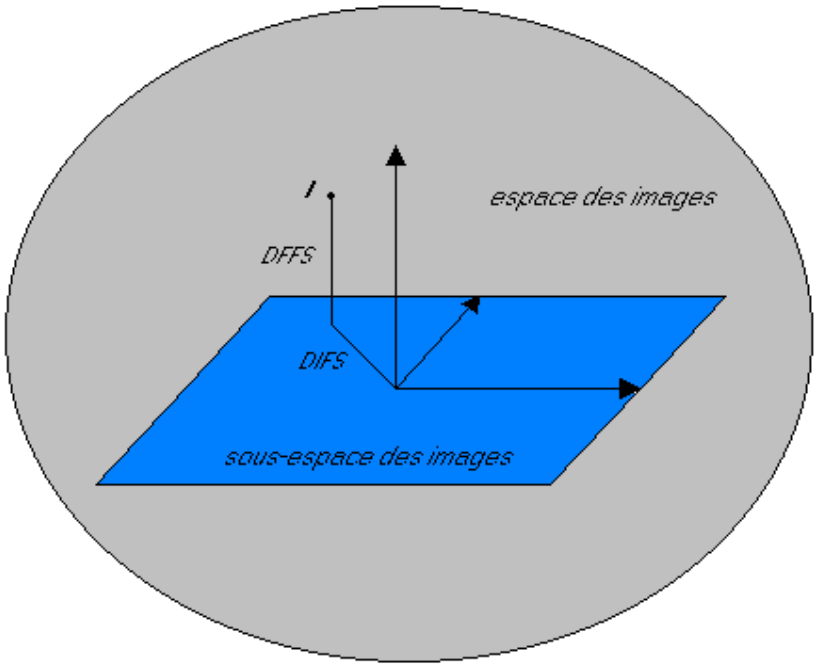

FIG. 2.2 – Image illustrant l'espace des images et le sous-espace des images contenant les images positives et négatives. DFFS signifie « Distance From Feature Space » et représente la distance entre l'image I et le sous-espace des images. DIFS signifie « Distance In Feature Space » et représente la distance entre l'origine du sous-espace et la projection de l'image I dans le sous-espace.

Si on faisait donc l'hypothèse que chaque dimension de l'espace des images représente un descripteur, alors on aurait beaucoup de descripteurs inutiles et les descripteurs utiles ne seraient pas forcément les meilleurs pour séparer les deux classes d'images positives et négatives par un classificateur statistique. C'est pour cela que différentes méthodes ont été mises au point pour construire des sous-espaces plus appropriés comme :

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA) ou Fisher's Linear Discriminant (FLD)
- Factor Analysis (FA)

Les sous-espaces plus appropriés sont aussi utilisés pour faciliter les calculs et filtrer le bruit.

2.2.2.2 Principal Component Analysis (PCA)

Comme mentionné précédemment, PCA [5,6] sert à construire un sous-espace plus approprié pour représenter les images exemples. La méthode cherche les axes orthogonaux pour lesquels la variance des images exemples projetées est maximale. Ces axes conviennent particulièrement bien pour représenter les images exemples, mais ils ne permettent par contre pas forcément de bien séparer les images exemples si celles-ci appartiennent à plusieurs classes. L'axe possédant la variance maximale est appelée *premier axe principal*. Dans tout l'espace des images, il n'existe aucun vecteur de projection avec une variance plus grande que celle du premier axe principal. Le *deuxième axe principal* est l'axe qui est perpendiculaire au première axe principal et qui pointe dans la direction pour laquelle la projection des images exemples possède de nouveau la variance maximale. Les axes principaux suivants possèdent les mêmes propriétés par rapport aux axes principaux précédents que le deuxième axe principal par rapport au premier. Comme les premiers axes principaux possèdent les plus grandes variances, ce sont eux qui permettent de représenter le mieux les images exemples. Le nombre d'axes principaux à utiliser comme descripteurs est choisi de façon à garder 85-90% de la variance totale.

La procédure pour calculer les axes principaux normalisés $\mathbf{u}_i$ est la suivante. D'abord, il faut calculer l'image moyenne des images exemples :

$$\bar{\mathbf{X}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{X}_i \tag{2.3}$$

Ensuite, il faut soustraire à chaque image exemple l'image moyenne :

$$\hat{\mathbf{X}}_i = \mathbf{X}_i - \bar{\mathbf{X}}, \textit{ pour } i = 1,...,M \tag{2.4}$$

Les vecteurs $\hat{\mathbf{X}}_i$ sont rassemblés dans la matrice

$$\mathbf{D} = \left[\hat{\mathbf{X}}_1\hat{\mathbf{X}}_2...\hat{\mathbf{X}}_M\right] \tag{2.5}$$

et la matrice de covariance

$$\mathbf{C} = \frac{1}{M-1}\mathbf{D}\mathbf{D}^T \tag{2.6}$$

est calculée. Maintenant, les vecteurs propres $\mathbf{v}_i$ avec leur valeur propre λ_i sont calculés et mis dans l'ensemble

$$\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M\} \tag{2.7}$$

de façon à ce que

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_M \tag{2.8}$$

Le vecteur propre $\mathbf{v}_i$ de la matrice de covariance $\mathbf{C}$ pointent dans la même direction que la *i*-ème axe principal normalisé $\mathbf{u}_i$ de $\mathbf{D}$. On peut alors calculer les axes principaux $\mathbf{u}_i$:

$$\mathbf{u}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} \quad \text{pour } i = 1,2,...,M \tag{2.9}$$

Une image $\mathbf{X}$ peut être projetée dans le sous-espace en effectuant le calcul

$$\boldsymbol{\omega} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ . \\ \mathbf{u}_Q^T \end{bmatrix} \cdot \mathbf{X} \tag{2.7}$$

avec $\boldsymbol{\omega}$ le vecteur des poids et Q nombre d'axes principaux choisis. L'image $\mathbf{X}$ peut être reconstruite approximativement par

$$\mathbf{X}_r \approx \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & . & \mathbf{u}_Q \end{bmatrix} \cdot \boldsymbol{\omega} \tag{2.8}$$

avec une erreur de reconstruction

$$\varepsilon = \|\mathbf{X} - \mathbf{X}_r\|^2 \tag{2.9}$$

L'erreur de reconstruction ε donne une indication de la distance entre l'image $\mathbf{X}$ et le sous-espace.
Si les images exemples comprennent seulement des images positives, alors l'erreur de reconstruction ε peut être directement utilisée comme classificateur. On supposera que l'objet caractéristique à localiser se trouve dans la région avec l'erreur de reconstruction ε minimale dans l'image scannée. Mais si les images exemples consistent d'images positives et négatives, alors un classificateur statistique sera nécessaire.

2.2.2.3 Linear Discriminant Analysis (LDA) – Fisher's Linear Discriminant (FLD)

Similaire au PCA, LDA [5] cherche aussi à construire un sous-espace plus approprié pour représenter les images exemples. Elle est applicable lorsque les images exemples appartiennent à deux classes ou plus. La méthode cherche les axes orthogonaux normalisés $\mathbf{u}_i$ pour lesquels le quotient

$$J(\mathbf{u}_i) = \frac{\mathbf{u}_i^T \mathbf{S}_B \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{S}_W \mathbf{u}_i} \tag{2.10}$$

est maximale avec $\mathbf{S}_B$ la matrice de variance entre les classes et $\mathbf{S}_W$ la matrice de variance à l'intérieur des classes (la somme des matrices de covariance des différentes classes). Les deux matrices sont données par :

$$\mathbf{S}_B = \sum_{i=1}^{C} \frac{n_i}{n} (\mu_i - \mu)(\mu_i - \mu)^T \tag{2.11}$$

et

$$\mathbf{S}_W = \sum_{i=1}^{C} \frac{n_i}{n} \sum_{x_k \in \omega_i} (x_k - \mu_i)(x_k - \mu_i)^T \tag{2.12}$$

avec C le nombre de classes, n_i le nombre d'échantillons (d'images exemples) appartenant à la classe i et n le nombre d'échantillons (d'images exemples) total.
Le sous-espace, qui peut être créé avec cette méthode, est aux maximum de dimension $C-1$.
Pour classifier les C classes, un classificateur statistique est appliqué sur le sous-espace.

2.2.3 Méthodes « entraînant des classificateurs » statistiques

2.2.3.1 Machines à Vecteurs de Support (SVM)

SVM [5] est une technique pour entraîner un classificateur complexe sur la base d'images exemples. SVM cherche l'hyperplan qui permet de séparer le « mieux » les données d'entraînement. L'hyperplan est appelé le « meilleur » parce qu'il maximise la marge entre lui et les deux classes. Cela veut dire qu'il maximise la distance entre lui et les points les plus proches des deux classes.
Pour construire l'hyperplan, l'algorithme cherche à minimiser la fonction coût

$$F(\mathbf{w}) = \mathbf{w}^T \mathbf{w} \tag{2.13}$$

en respectant la contrainte

$$C_1 : \; y_i(\mathbf{w}^T \cdot \mathbf{x}_i + \omega_0) \geq 1 \quad \text{pour} \quad i = 1,...,M \tag{2.14}$$

avec $y_i = 1$ si $\mathbf{x}_i \in \text{classe}\,1$, $y_i = -1$ si $\mathbf{x}_i \in \text{classe}\,2$, $\mathbf{w}$ vecteur des poids qui est perpendiculaire à l'hyperplan, ω_0 le seuil et M le nombre d'images exemples.
La méthode standard pour résoudre ce problème est d'utiliser la méthode de Lagrange. Si les images exemples ne sont pas linéairement séparables, on peut relaxer la contrainte et introduire un terme de coût supplémentaire dans la fonction coût. La fonction coût devient

$$F(\mathbf{w}) = \mathbf{w}^T \mathbf{w} / 2 + C \sum_{i=1}^{M} \xi_i \tag{2.15}$$

et les nouvelles contraintes

$$\begin{aligned} &C_1 : \; \mathbf{w}^T \cdot \mathbf{x}_i + \omega_0 \geq 1 - \xi_i \quad \text{pour} \quad y_i = 1 \\ &C_2 : \; \mathbf{w}^T \cdot \mathbf{x}_i + \omega_0 \leq -1 + \xi_i \quad \text{pour} \quad y_i = -1 \\ &C_3 : \; \xi_i \geq 0 \end{aligned} \tag{2.16}$$

avec C un paramètre de régulation et ξ_i une variable d'erreur.
Si les images exemples ne sont pas linéairement séparables, une autre méthode consiste à projeter les images exemples par des fonctions non-linéaires $\mathbf{\Phi}(\mathbf{x})$ dans un espace dans lequel ils sont linéairement séparables et à appliquer l'algorithme SVM dans cet espace. Il est conseillé d'utiliser des fonctions non-linéaires $\mathbf{\Phi}(\mathbf{x})$ simples, car sinon il est probable de créer un « overfitting » des données.

2.2.3.2 Réseaux de Neurones

Un réseau de neurones [7] peut être un perceptron multicouche. L'unité de base du réseau de neurones est le perceptron. Chaque perceptron effectue un travail relativement simple : il reçoit des données pondérées des voisins ou des sources externes et calcule sur cette base un signal de sortie qui est propagé à d'autres unités. On distingue entre unité d'entrée, de sortie et cachée.
Un réseau de neurones doit être configuré pour que l'application d'un ensemble de données d'entrées produisent le résultat désiré à la sortie. Par configurer, on entend choisir les poids à attribuer aux connexions. Il y a plusieurs façons de configurer un réseau de neurones :

- Une est de choisir les poids explicites en utilisant du savoir à priori.
- Une autre est d'entraîner le réseau de neurones en lui donnant des images exemples et en le laissant changer la valeur de ses poids selon une règle d'apprentissage.

Pour la deuxième façon de configuration, il existe deux situations d'apprentissage distinctes :

- L'apprentissage supervisé. Dans ce cas, les images exemples doivent être classifiées.
- L'apprentissage non-supervisé. Dans ce cas, les images exemples ne doivent pas être classifiées. Le système est supposé découvrir les images qui se ressemblent statistiquement et de les attribuer à une classe.

2.2.3.3 Hidden Markov Models (HMM)

Hidden Markov Models [10,11] sont un ensemble de modèles statistiques utilisés pour caractériser les propriétés statistiques d'une image. L'image est divisée en N régions significatives qui sont, par exemple pour le visage, les cheveux, le front, les yeux, le nez et la bouche. Chacune de ces régions est ensuite assignée à un état S_i dans un HMM 1D :

$$\begin{aligned} S_1 &\rightarrow \text{cheveux} \\ S_2 &\rightarrow \text{front} \\ S_3 &\rightarrow \text{yeux} \\ S_4 &\rightarrow \text{nez} \\ S_5 &\rightarrow \text{bouche} \end{aligned} \qquad (2.17)$$

Chaque état est caractérisé par une fonction de probabilité, estimée sur la base des images exemples.
Le principe de HMM, lors de la localisation du visage, est de toujours extraire les mêmes régions de l'image d'entrée et de vérifier si les objets caractéristiques apparaissent dans le même ordre que défini dans le modèle HMM.

2.3 Méthodes Feature-Based

2.3.1 Introduction

Les méthodes feature-based [2] (ou geometrical-based) nécessitent des connaissances préalables sur les objets caractéristiques pour pouvoir les localiser. D'où leur nom : méthodes utilisant de l'information à priori sur le feature à localiser (objet caractéristique) ou méthodes

utilisant de l'information à priori sur les propriétés géométriques de l'objet caractéristique à localiser.
Les méthodes feature-based peuvent être divisées en trois catégories d'approches :

- L'*analyse bas niveau* : Elle essaie de segmenter des objets caractéristiques en utilisant des propriétés des pixels comme par exemple la couleur. Les objets caractéristiques localisés à l'aide de cette méthode sont ambigus.
- L'*analyse des objets caractéristiques* : Elle essaie de traiter les objets caractéristiques à localiser dans une vue plus globale. Pour cela elle utilise par exemple de l'information sur la géométrie du visage. À l'aide de l'analyse des objets caractéristiques l'ambiguïté de ceux-ci peut être réduite.
- L'utilisation de *formes actives* (*actif shape models*) : Ces méthodes ont été développées pour extraire des objets caractéristiques complexes et non-rigides comme la pupille de l'œil ou les lèvres.

Un algorithme de localisation feature-based utilise normalement une combinaison de ces trois catégories d'approches.

2.3.2 Analyse bas niveau

L'analyse de bas niveau [2] effectue des analyses sur les propriétés des bords ou de la couleur des objets caractéristiques.
Lors de l'*analyse des bords*, on applique d'abord un filtre pour trouver les bords dans l'image. Des filtres utilisés sont :

- l'opérateur de Sobel
- l'opérateur de bord de Marr-Hildreth
- une variété de dérivé première ou deuxième de gaussiennes

Ensuite les bords sont labellisés et la forme analysée (comparée à un modèle) pour trouver les objets caractéristiques.
Lors de l'*analyse de la couleur*, on utilise la propriété que les cils, la pupille, les lèvres, etc. apparaissent normalement plus sombre que les régions les entourant. Cette propriété est utilisée pour différencier différentes parties du visage.

2.3.3 Analyse des objets caractéristiques

Les objets caractéristiques localisés lors de l'analyse bas niveau [2] sont souvent ambigus. Par exemple, lorsqu'on localise des objets caractéristiques à l'aide de la couleur, des autres objets de couleur similaire peuvent aussi être détectés. Donc, on a plusieurs candidats pour un objet caractéristiques. L'analyse des objets caractéristique peut aider à résoudre ce problème. En utilisant une connaissance à priori sur la géométrie du visage, elle permet de vérifier si un objet détecté est l'objet caractéristique cherché ou pas. Il y a deux approches possibles dans l'application du savoir sur la géométrie du visage.
L'*approche cherchant les objets caractéristiques* cherche d'abord des objets simples à détecter. La détection des objets caractéristiques simples permet de faire des hypothèses sur la position des objets caractéristiques plus durs à trouver, à l'aide de connaissances géométriques du visage. Souvent se sont les yeux qui servent comme référence de départ.
L'*approche vérifiant la constellation des objets caractéristiques* utilise un modèle de la constellation de plusieurs objets caractéristiques et vérifie quel ensemble d'objets détectés

correspond le mieux à ce modèle. Le modèle peut être un modèle probabiliste de l'arrangement spatial des objets caractéristique.

2.3.4 Active Shape Models

Comme mentionné dans la section 2.3.1, ces méthodes ont été développées pour extraire des objets caractéristiques complexes et non-rigides, comme l'œil ou les lèvres. Au départ, un active shape model [2] doit être placé proche de l'objet caractéristique à localiser dans l'image. L'active shape model va par la suite interagir avec les propriétés de l'image locale et se déformer lentement pour prendre la forme de l'objet caractéristique à extraire. Il y a trois types d'active shape models.

Le *contour actif*, aussi appelé snake, est utilisé pour trouver le contour d'un objet caractéristique. Pour faire cela, on doit initialiser le contour actif autour de l'objet caractéristique à localiser. Le contour actif se concentre en suite sur les bords proches et prend pas à pas la forme de l'objet caractéristique. Le but du contour actif est de minimiser une fonction énergie

$$E_{snake} = E_{\text{int}} + E_{ext} \tag{2.18}$$

avec E_{int} l'énergie interne et E_{ext} l'énergie externe. L'énergie interne dépend des propriétés intrinsèques du contour actif et définit l'évolution naturelle de celui-ci. L'évolution naturelle d'un contour actif est typiquement rétrécir et dilater. L'énergie externe agit contre l'énergie interne et permet au contour actif de se comporter contre l'évolution naturelle. C'est ce qui permet au contour actif d'entourer le bord de l'objet caractéristique.

Les *masques déformables* (deformable template) sont une évolution des contours actifs. Ils incorporent de l'information globale de l'objet caractéristique à localiser. Un masque déformable consiste en un contour qui est défini par n paramètres. Ce contour permet de modéliser des formes typiques que l'objet caractéristique peut prendre, pour l'œil par exemple différentes ellipses. Si on pose le masque déformable près de l'objet caractéristique, alors le masque (contour paramétré) va changer les valeurs de ses paramètres (va se déformer) pour converger vers le contour optimal de l'objet caractéristique. De nouveau, le but est de minimiser une fonction énergie.

Le *PDM* (Point distributed models) est une description paramétrique compacte de la forme de l'objet caractéristique à localiser sur la base de statistiques. Lors du PDM, le contour est discrétisé en un nombre de points labellisés. La variation de ces points est calculée à l'aide d'images exemples sur l'objet caractéristique à localiser en utilisant PCA. De telle façon, on aboutit à un modèle linéaire flexible. Le modèle consiste en la valeur moyenne de tous les points et des composantes principales pour chaque point.

Pour localiser l'objet caractéristique, le « mean shape model », étant constitué des valeurs moyennes des points, est placé près de l'objet caractéristique. Ensuite, une stratégie, utilisant l'intensité locale de l'image, est utilisée pour déplacer chaque point du modèle vers le point sur le contour de l'objet caractéristique. Pendant la déformation, la forme peut seulement changer son apparence d'une façon qui coïncide avec l'information modélisée.

Chapitre 3

Méthode pour la localisation de visages

3.1 Introduction

3.1.1 Objectif

Dans ce chapitre, la méthode utilisée pour effectuer la localisation de visages est présentée. Rappelons pour commencer l'objectif à atteindre. On possède une image contenant un visage dont on veut détecter trois objets caractéristiques : l'œil droit, l'œil gauche et la bouche. Le but est de connaître en fin de compte la position exacte du visage dans l'image à l'aide de ces objets caractéristiques. Comme montré dans la figure 1.1, la localisation est seulement une étape intermédiaire qui peut être suivie par la reconnaissance d'expressions faciales ou la reconnaissance et/ou vérification d'identité de personnages. Les algorithmes de la reconnaissance et/ou vérification de personnages et de la reconnaissance d'expressions faciales ont besoin d'images de visages, vus plus ou moins de face. Pour cette raison, nous supposons que le visage à localiser dans l'image est vu plus ou moins de face. On suppose aussi que la personne n'a pas la tête trop penchée sur un des côtés et que la taille du visage est normalisée. Si ce n'est pas le cas, on pourra toujours, en implantant une fonction qui effectue un changement de taille par interpolation adaptée de l'image, détecter des visages non-normalisés. Les paramètres critiques sont donc surtout la variance de l'illumination du visage, la variation du visage lui-même et le fond non connu entourant la tête. Dans ce contexte, le but est d'implémenter un algorithme de localisation qui est plus ou moins insensible aux paramètres critiques précédemment mentionnés.
Comme décrit dans 1.5, il est difficile dans le monde réel de concevoir un classificateur qui minimise le taux des mauvaises détections négatives f_n et le taux des mauvaises détections positives f_p en même temps. On est donc obligé de faire un compromis. Il existe trois compromis principaux qui peuvent être faits :

- Le premier est de minimiser le taux de mauvaises détections négatives f_n en augmentant en même-temps le taux de mauvaises détections positives f_p. Ce compromis permet d'augmenter la probabilité de vraiment détecter tous les objets dans une image en augmentant évidemment aussi les fausses détections. Par un traitement a posteriori, les mauvaises détections peuvent être éliminées. Un traitement a posteriori est de chercher des régions dans l'image dans lesquelles il y a des fenêtres de détection superposées. Ces régions ont une grande probabilité de contenir un objet caractéristique. Ce compromis est souvent fait lorsqu'on veut détecter tous les visages dans une image.
- Le deuxième compromis qui peut être fait est de minimiser le taux de mauvaises détections positives f_p en augmentant en même-temps le taux des mauvaises détections négatives f_n. Ce compromis permet d'augmenter la probabilité d'avoir une bonne détection ou de diminuer la probabilité d'avoir une détection pour une région qui ne contient pas d'objet. L'inconvénient est que des régions contenant un objet peuvent être ignorées. Ce compromis est fait dans les cas où on veut être sûr de seulement détecter des régions contenant des objets.
- Le dernier compromis qui peut être fait est de minimiser la somme du taux de mauvaises détections positives f_p et du taux de mauvaises détections négatives f_n. Ce compromis permet de minimiser la probabilité d'avoir une mauvaise détection qu'elle soit positive ou négative.

Dans le cas de la localisation du visage, on souhaite au moins détecter les 2 yeux et la bouche. On est prête à avoir plus de détections négatives pour un objet caractéristique car on peut ensuite garder la fenêtre ayant la plus grande probabilité de contenir l'objet caractéristique recherché.

3.1.2 Choix de la méthode

Récapitulons les points essentiels qui définissent un bon algorithme de localisation. Il doit être robuste par rapport :

- à la variation de l'illumination du visage ;
- à la variation du visage lui-même ou la déformation ;
- à la taille du visage ;
- et au fond entourant le visage non connu.

En plus, on aimerait qu'il possède un taux de mauvaises détections négatives f_n minimal.

Comme décrit dans le chapitre 2, on a à disposition deux approches différentes : image-based et feature-based. L'approche image-based, grâce à sa nature statistique, est plus robuste par rapport à la variation de l'illumination du visage, à la variation du visage lui-même et au fond entourant le visage non connu que l'approche feature-based. La caractéristique qui marque l'approche feature-based est sa robustesse par rapport à des rotations de la tête dans l'espace. Comme le visage est seulement vu de face ou légèrement tourné de côté, il se dégage que l'approche image-based est plus adaptée pour l'algorithme de localisation.
L'algorithme de localisation implémenté se base sur un algorithme appelé AdaBoost qui permet de combiner plusieurs hypothèses simples pour créer une autre hypothèse plus performante. Comme il peut être montré que l'erreur de généralisation pour AdaBoost peut être limitée, on a une certaine garantie d'avoir des bons résultats lors de la détection. Les hypothèses simples sont des classificateurs statistiques linéaires qui sont crées à partir d'un algorithme d'entraînement basé sur des images positives et négatives (approche image-based). Chaque hypothèse utilise un descripteur basé sur un masque visuel. Comme les classificateurs créés par AdaBoost classifieront probablement plusieurs endroits dans l'image comme contenant un objet caractéristique trois variantes différentes pour faire un arbitrage sont implémentées dans l'algorithme de localisation.
Les descripteurs sont présentés plus en détail dans la section suivante (chapitre 3.2). Le chapitre 3.3 présente les classificateurs faibles et le chapitre 3.4 AdaBoost. Pour finir, les trois méthodes d'arbitrage sont présentées dans 3.5

3.2 Les descripteurs

3.2.1 Introduction

Le but est de trouver une bonne méthode permettant la description des trois objets caractéristiques à extraire au moyen d'un ensemble de paramètres adéquats. Les éléments de la représentation sont appelés les attributs ou les descripteurs [12]. Normalement, les attributs à extraire doivent idéalement répondre aux exigences suivantes :

- Faible variance inter-classe ;
- Grande variance intra-classe ;
- Faible nombre d'attributs ;

- Indépendance en translation, rotation et facteur d'échelle.

Comme mentionné dans 2.2.1, les méthodes image-based font glisser une fenêtre dans toute l'image du visage et classifient avec cette méthode toutes les régions pour trouver la position des objets caractéristiques. Ceci donne une garantie que les attributs à extraire sont indépendants en translation. Avec les hypothèses faites dans 3.1.1, on peut aussi assumer que les attributs sont indépendants en rotation et en facteur d'échelle. Le nombre d'attributs ne pose pas vraiment de problème, puisqu'AdaBoost les sélectionne séquentiellement.
Il reste donc à chercher après une méthode de description qui a des attributs avec une faible variance intra-classe et une grande variance inter-classe.

3.2.2 Choix des descripteurs

Pour commencer, définissons le vecteur $\mathbf{Y}$ comme représentation de la fenêtre que l'on fait glisser dans toute l'image pour détecter les objets caractéristiques :

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ . \\ Y_S \end{pmatrix} \tag{3.1}$$

avec S le nombre de pixels dans la fenêtre.
Principalement, il existe deux méthodes pour créer des descripteurs D_i pour des images non binaires [13]:

- La première est d'attribuer un descripteur D_i à chaque pixel de la fenêtre (à chaque variable Y_i). Le descripteur D_i prend alors la valeur d'intensité du pixel auquel il est attribué.

$$D_i = Y_i \quad \text{avec} \quad i = 1,...,S \tag{3.2}$$

- La deuxième est d'attribuer un descripteur D_i à une transformation (linéaire ou non-linéaire) des variables originales $\mathbf{Y}$.

$$D_i = f_i(\mathbf{Y}) \tag{3.3}$$

avec f_i la fonction effectuant la transformation.

Dans notre cas, où on attribue un descripteur à une hypothèse simple, la première approche n'est pas idéale. La raison est qu'elle n'est pas robuste par rapport à la variation de l'illumination de l'image. Imaginons, pour illustrer cela, qu'on ait entraîné un classificateur linéaire ou bayésien par pixel sur la base d'images positives et négatives avec une faible illumination. Intuitivement, les seuils de décision des différents classificateurs seront bas. Si maintenant, lors de la localisation, on doit classifier des images avec une forte illumination, il est fort probable que l'on a des détections pour presque toutes les régions. Donc, on aurait beaucoup de fausses détections. Le cas contraire peut aussi être imaginé. Dans ce cas là, on n'aura presque pas de détections, voir même aucune.

La solution est d'appliquer une transformation f_i à la variable. Des méthodes connues qui effectuent des transformations linéaires sont par exemple PCA, LDA et Factor Analysis (voir 2.2.2). Ces méthodes sont des techniques pour trouver un espace de représentation adapté selon l'application. En attribuant à chaque nouvelle variable un descripteur, on pourra construire des hypothèses assez bonnes. On pourrait appliquer AdaBoost à ces hypothèses, mais l'hypothèse résultante ne sera probablement pas beaucoup meilleure.
Ce qui est intéressant, et pourquoi AdaBoost a été conçu, est d'appliquer AdaBoost à des hypothèses simples h_i, c'est à dire à des hypothèses classifiant juste plus que 50% des données correctement, c'est à dire juste meilleures que la sélection aléatoire.
Une autre méthode pour créer des descripteurs D_i, qui peut aussi être classifiée de transformation linéaire, est de construire des masques visuels $\mathbf{M}_i$.

$$\mathbf{M}_i = \begin{pmatrix} M_1 \\ M_2 \\ . \\ M_S \end{pmatrix} \tag{3.4}$$

avec S le nombre de pixels dans la fenêtre $\mathbf{Y}$.
Les masques visuels $\mathbf{M}_i$ ont la même taille que la fenêtre $\mathbf{Y}$ que l'on fait glisser dans l'image. Le descripteur D_i attribué au masque $\mathbf{M}_i$ prend la valeur de la corrélation de la région de l'image se trouvant dans la fenêtre $\mathbf{Y}$ avec le masque $\mathbf{M}_i$, ce qui peut aussi être écrit comme

$$D_i = \mathbf{M}_i^T \mathbf{Y} \tag{3.5}$$

Les masques visuels ont l'avantage de pouvoir encoder du savoir à priori. Si on prend certaines précautions, on peut même créer des descripteurs D_i qui sont invariants par rapport à la variation de l'illumination. Ces masques visuels $\mathbf{M}_i$ sont intéressants, car ils permettent de créer des hypothèses h_i simples et robustes par rapport à la variation de l'illumination. C'est la raison pour laquelle des masques visuels $\mathbf{M}_i$ ont été choisis pour construire des descripteurs D_i. Le chapitre suivant montre la façon dont sont construit les masques visuels $\mathbf{M}_i$.

3.2.3 Les masques visuels

Comme expliqué dans le chapitre précédent, des masques visuels $\mathbf{M}_i$ sont utilisés pour construire des descripteurs D_i. Les masques visuels $\mathbf{M}_i$ utilisés sont une variante de l'ondelette de Haar. L'ondelette de Haar est utilisée dans l'analyse de signaux 1-D pour calculer la différence d'intensité dans une région. Rappelons la fonction de l'ondelette de Haar :

$$\Psi(X)=\begin{cases}1 \text{ si } 0\le x\le 0.5\\ -1 \text{ si } 0.5< x\le 1\\ 0 \text{ sinon}\end{cases} \tag{3.6}$$

Comme on peut voir facilement, la moyenne de l'ondelette de Haar est 0. Cette propriété est intéressante pour la raison suivante : L'ondelette de Haar permet de filtrer des composantes des basses fréquences du signal 1-D. L'illumination du visage est dans tout le visage environ constante, indépendamment de l'intensité de l'illumination. On peut dire que c'est un effet se manifestant dans les fréquences basses. Une variante d'ondelette de Haar permet donc, grâce au fait que la moyenne est 0, de filtrer l'illumination lorsqu'on l'applique sur des petites régions du visage. L'usage d'ondelettes de Haar permet donc de rendre l'algorithme de localisation plus robuste en vers la variation de l'illumination, ce qui était un critère mentionné dans 3.1.2.

Dans 3.2.1, il a été dit que les descripteurs D_i devraient avoir une faible variance inter-classe et une grande variance intra-classe. Comme les masques visuels $\mathbf{M}_i$ permettent d'encoder du savoir à priori, il faut maintenant trouver un modèle des objets caractéristiques à localiser qui utilise l'ondelette de Haar, augmente la variance intra-classe et diminue la variance inter-classe. Le modèle utilisé est une combinaison de la fonction de Haar et une Gaussienne. On va d'abord présenter ce modèle et ensuite justifier pourquoi avoir choisit ce modèle.

Supposons que le rectangle gris suivant représente la fenêtre $\mathbf{Y}$ qu'on fait glisser dans toute l'image.

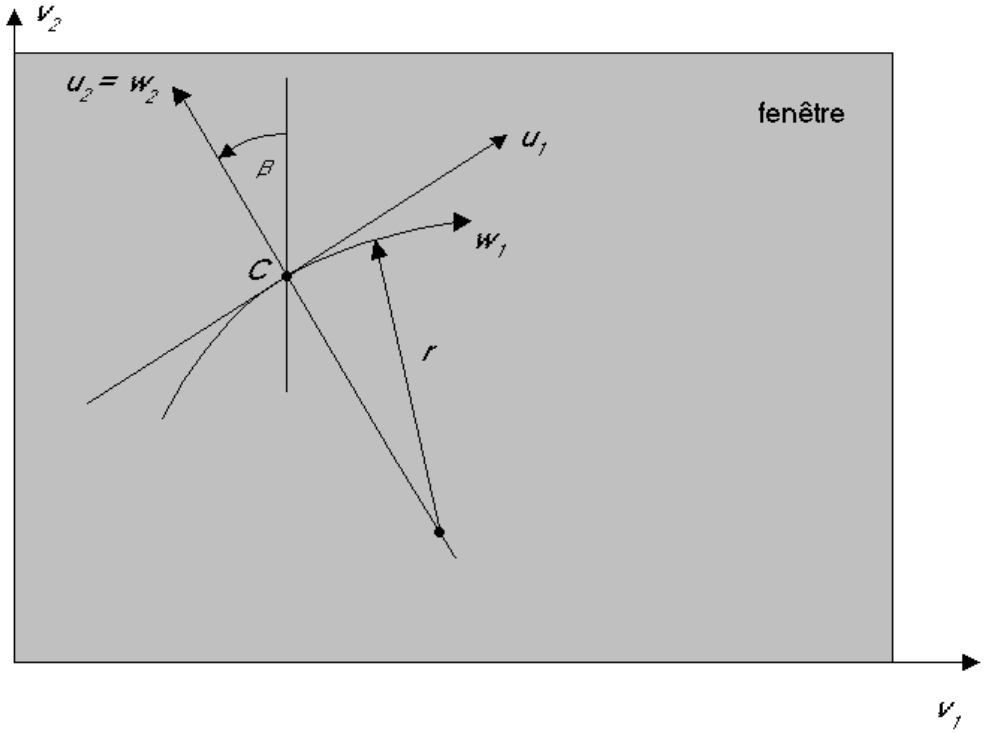

FIG. 3.1 – Image illustrant la fenêtre qu'on fait glisser dans l'image avec le modèle du masque visuel.

Alors v_1 et v_2 représentent le référentiel relatif de la fenêtre $\mathbf{Y}$ que l'on fait glisser dans l'image. Le modèle, représentant les objets caractéristiques, est un modèle 2-D qui en chaque point de la fenêtre prend une valeur calculée à partir du produit de la fonction de Haar et une Gaussienne. La fonction de Haar varie selon l'axe w_2 et la Gaussienne selon l'axe w_1. Le modèle est défini par 6 paramètres :

- Les deux coordonnées du centre $\mathbf{C}$.
- Le facteur S_{w1} qui permet de changer l'échelle de l'axe w_1.
- Le facteur S_{w2} qui permet de changer l'échelle de l'axe w_2.
- L'angle β qui permet de définir de combien de degrés le référentiel $\mathbf{w}$ est tourné par rapport au référentiel $\mathbf{v}$.
- Le rayon r qui permet de définir le cercle selon lequel l'axe w_1 est tordu.

L'équation du modèle dans le référentiel $\mathbf{w}$ est donnée par :

$$f(w_1, w_2) = e^{-(s_{w1} \cdot w_1)^2} \cdot \begin{cases} 1 \text{ si } 0 <= s_{w2} \cdot w_2 < 1 \\ -1 \text{ si } -1 < s_{w1} \cdot w_2 < 0 \\ 0 \text{ sinon} \end{cases} \tag{3.7}$$

Le modèle est assez flexible ce qui est illustrer avec l'image suivante.

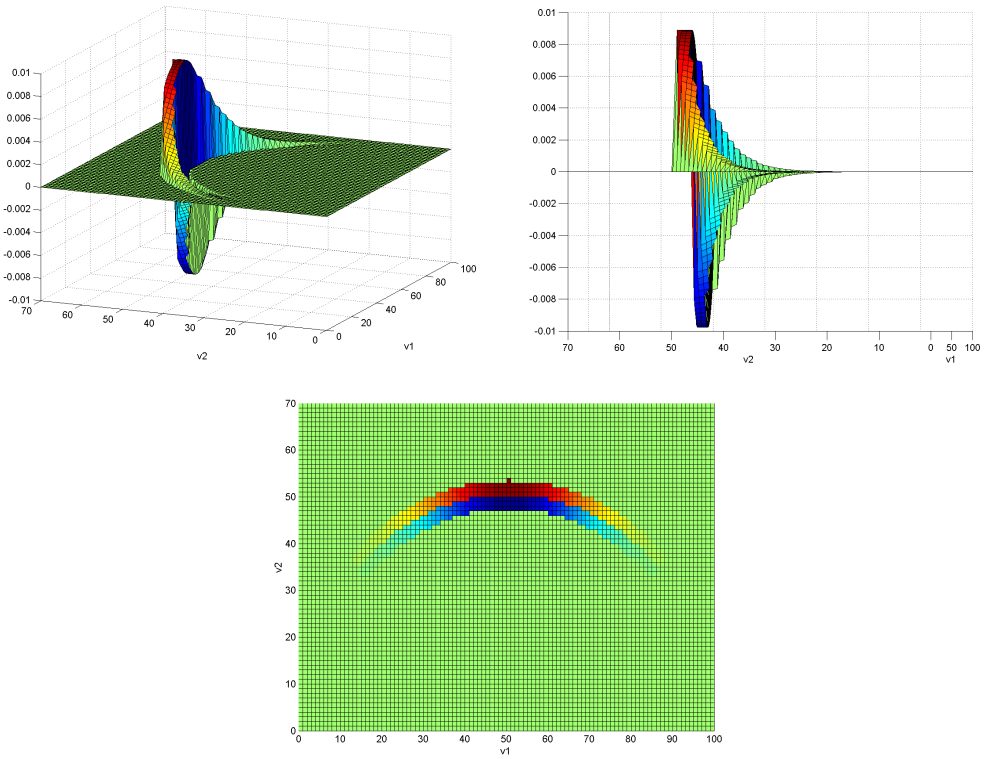

FIG. 3.2 – Image illustrant le modèle du masque visuel. Les paramètres pour ce modèle sont :
$C_1 = 50, C_2 = 50, S_{w1} = 0.05, S_{w2} = 1, \beta = 0, r = 50$

En voyant cette image, on peut comprendre pourquoi ce modèle est un bon candidat pour modéliser les objets caractéristiques qui sont l'œil gauche, l'œil droit et la bouche. Ce modèle est capable de faire des approximations assez bonnes des lèvres ou du contour d'un œil, etc. Comme on utilise des images de faible résolution (fenêtre 7x10), on simplifie les masques en quantifiant sur 3 niveaux {-1,0,1} en garantissant une moyenne nulle. La figure suivante montre quelques masques visuels $\mathbf{M}_i$.

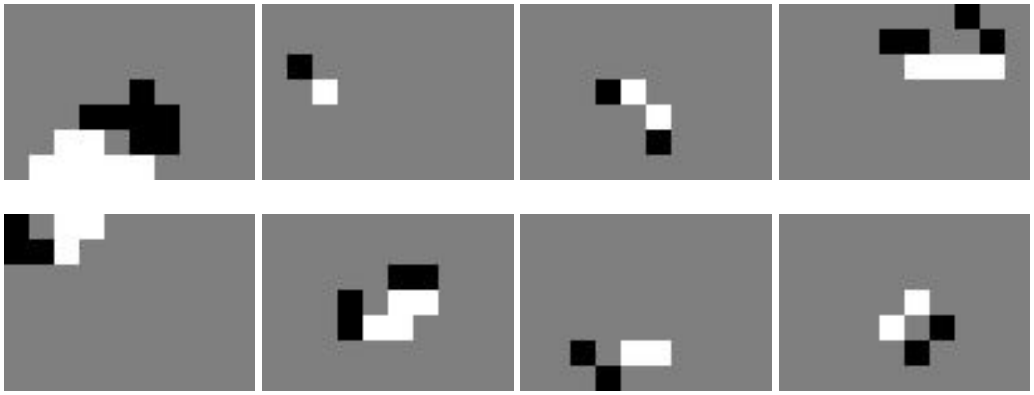

FIG. 3.3 – Image illustrant des masques visuels $\mathbf{M}_i$. Les pixels en noir correspondent à –1, ceux en blanc à +1 et les grises à 0.

3.3 Les classificateurs faibles

3.3.1 Introduction

Dans ce chapitre, les classificateurs faibles sont présentés. Un classificateur est dit faible dans le langage du Boosting lorsqu'il ne possède pas des bonnes performances de classifications. Il doit seulement pouvoir classifier plus que 50% des données correctement. Comme classificateur faible, des classificateurs linéaires sont utilisés. Les classificateurs linéaires utilisés h_j se composent d'un descripteur D_j, d'un seuil Θ_j et d'une variable de parité p_j qui indique la direction du signe d'inégalité.

$$h_j(\mathbf{X}) = \begin{cases} 1 & \text{si } p_j \cdot D_j(\mathbf{X}) < p_j \cdot \Theta_j \\ -1 & \text{sinon} \end{cases} \tag{3.8}$$

Le descripteur D_j est construit à partir d'un masque visuel $\mathbf{M}_j$ et est calculé selon l'équation 3.5. Les classificateurs faibles h_j sont entraînés par un algorithme d'entraînement sur la base d'images positives et négatives. L'algorithme d'entraînement est détaillé dans le chapitre suivant. Le chapitre 3.3.3 donne une définition de classificateurs faibles. Il est suivi par le chapitre 3.3.4 qui explique pourquoi des classificateurs linéaires ont été choisis comme classificateurs faibles.

3.3.2 Algorithme d'entraînement

Le but de l'algorithme d'entraînement est de chercher le seuil optimal Θ_j pour un classificateur faible h_j. Le seuil est optimal s'il minimise le nombre d'images exemples mal classifiées.

Entrée : $S = \{(\mathbf{x}_1, y_1, d_1^t), ..., (\mathbf{x}_N, y_N, d_N^t)\}$ *avec* N *le nombre d'images d'entraînement*

Faire :

1) *Construire l'histogramme cumulatif des images positives* $f_{p,j}$ *et négatives* $f_{n,j}$.
2) *Cherche les deux seuils* Θ_j^1 *et* Θ_j^2 *qui minimisent les fonctions* c_j^1 *respectivement* c_j^1.
3) *Calculer les erreurs d'entraînement* ε_j^1 et ε_j^2.
4) *Attribuer aux paramètres de sortie* p_j et Θ_j *leur valeur :*

$$\Theta_j = \begin{cases} \Theta_j^1 & \text{si} \quad \varepsilon_j^1 < \varepsilon_j^2 \\ \Theta_j^2 & \text{sinon} \end{cases}$$

$$p_j = \begin{cases} p_j^1 & \text{si} \quad \varepsilon_j^1 < \varepsilon_j^2 \\ p_j^2 & \text{sinon} \end{cases}$$

Sortie : (Θ_j, p_j)

Algorithme 3.1 : Algorithme d'entraînement d'un classificateur faible h_j.

L'algorithme d'entraînement reçoit comme entrée l'ensemble des images positives, négatives et leurs pondérations correspondantes $S = \{(\mathbf{x}_1, y_1, d_1^t), ..., (\mathbf{x}_N, y_N, d_N^t)\}$. N est le nombre total d'images et y_i indique s'il s'agit d'une image positive ($y_i = 1$) ou négative ($y_i = -1$). Les pondérations d_i^t sont calculées par AdaBoost et seront introduites plus en détail dans le chapitre 3.4.
Pour calculer le seuil Θ_j, l'algorithme d'entraînement construit l'histogramme cumulatif des images positives $f_{p,j}$ et négatives $f_{n,j}$ en fonction du descripteur D_j et en respectant les pondérations des images d_i^t.

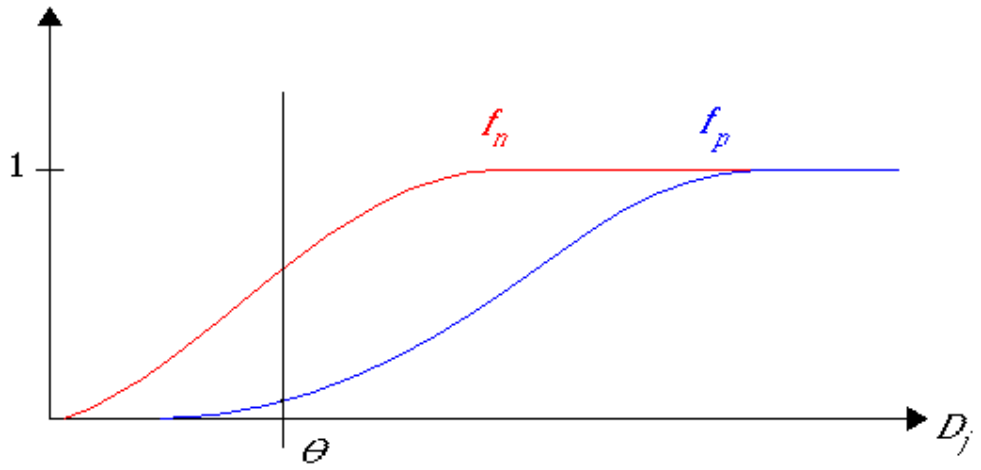

FIG. 3.4 – Image illustrant l'histogramme cumulatif des images positives f_p et négatives f_n.

L'histogramme cumulatif pour les images négatives est donné par :

$$f_{n,j}(\Theta) = \sum_{n=1}^{N} d_n^t \cdot I(\mathbf{x}_n, y_n, \Theta) \tag{3.9}$$

avec

$$I(\mathbf{x}_n, y_n, \Theta) = \begin{cases} 1 & \text{pour } D_j(\mathbf{x}_n) <= \Theta \text{ et } y_n = -1 \\ 0 & \text{sinon} \end{cases} \tag{3.10}$$

L'histogramme cumulatif pour les images positives est donné par :

$$f_{p,j}(\Theta) = \sum_{n=1}^{N} d_n^t \cdot I(\mathbf{x}_n, y_n, \Theta) \tag{3.11}$$

avec

$$I(\mathbf{x}_n, y_n, \Theta) = \begin{cases} 1 & \text{pour } D_j(\mathbf{x}_n) <= \Theta \text{ et } y_n = 1 \\ 0 & \text{sinon} \end{cases} \tag{3.12}$$

Ensuite, il cherche les deux seuils optimaux Θ_j^1 et Θ_j^2 auxquels sont attribués les bits de parités $p_j = 1$ respectivement $p_j = -1$. Θ_j^1 est trouvé en minimisant la fonction

$$c_j^1 = 1 - f_{p,j}(\Theta_j^1) + f_{n,j}(\Theta_j^1) \tag{3.13}$$

et Θ_j^2 en minimisant

$$c_j^2 = 1 - f_{n,j}\left(\Theta_j^2\right) + f_{p,j}\left(\Theta_j^2\right). \tag{3.14}$$

Ensuite, on calcule l'erreur de classification pour les deux classificateurs linéaires $h_j^1\left(\Theta_j^1, p_j^1 = 1\right)$ et $h_j^2\left(\Theta_j^2, p_j^2 = -1\right)$:

$$\varepsilon_j^1 = \sum_{n=1}^{N} d_n^t \cdot I\left(h_j^1(\mathbf{x}_n) \neq y_n\right) \tag{3.15}$$

$$\varepsilon_j^2 = \sum_{n=1}^{N} d_n^t \cdot I\left(h_j^2(\mathbf{x}_n) \neq y_n\right) \tag{3.16}$$

Pour finir, l'algorithme retourne les paramètres du classificateur ayant l'erreur de classification minimale.

$$\Theta_j = \begin{cases} \Theta_j^1 & \text{si} \quad \varepsilon_j^1 < \varepsilon_j^2 \\ \Theta_j^2 & \text{sinon} \end{cases} \tag{3.17}$$

$$p_j = \begin{cases} p_j^1 & \text{si} \quad \varepsilon_j^1 < \varepsilon_j^2 \\ p_j^2 & \text{sinon} \end{cases} \tag{3.18}$$

3.3.3 Définition d'un classificateur faible

Dans ce chapitre, la définition d'un classificateur faible est donnée. Un classificateur faible est défini dans [14] de la façon suivante :

Un classificateur est un classificateur faible pour un ensemble de données S si, donné une pondération quelconque de S, il possède une erreur de classification pondérée strictement inférieure à ½.

Lors du Boosting, une condition indispensable pour que l'algorithme ait une bonne performance est d'utiliser des classificateurs faibles h_j. Dans le contexte de la classification binaire, un classificateur faible est défini par son erreur empirique pondérée $\varepsilon\left(h_j, \mathbf{d}\right)$ strictement inférieure à ½-½γ :

$$\varepsilon\left(h, \mathbf{d}\right) = \sum_{n=1}^{N} d_n I\left(y_n \neq h_j(\mathbf{x}_n)\right) \leq \frac{1}{2} - \frac{1}{2}\gamma, \quad (\gamma > 0) \tag{3.19}$$

avec d_n la pondération de la paire de donnée $\left(\mathbf{x}_n, y_n\right)$, N le nombre de données dans l'ensemble $S = \left\{\left(\mathbf{x}_m, y_n\right)\right\}_{n=1}^{N}$ et γ le paramètre de bord qui quantifie la déviation de la performance du classificateur faible par rapport à un classificateur qui classifie les données au aléatoirement.

3.3.4 Pourquoi des classificateurs linéaires

Dans ce chapitre, il est expliqué pourquoi des classificateurs linéaires ont été choisis comme classificateurs faibles. Comme mentionné dans 3.3.3, le Boosting nécessite un ensemble H de classificateurs faibles pour fonctionner. La condition pour que les classificateurs $h_j \in H$ soient tous des classificateurs faibles est que γ dans l'expression 3.19 soit strictement plus grand que 0 pour tous les classificateurs. Dans [15], le cas a été considéré dans lequel l'ensemble des classificateurs faibles H consistait en classificateurs linéaires

$$h_j(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{3.20}$$

et les données $\mathbf{x}$ appartenaient à l'espace $\mathbf{R}^d$. Il a été montré que pour n'importe quelle pondération $\mathbf{d}$ des données d'apprentissage, l'erreur empirique pondérée $\varepsilon(h_j, \mathbf{d})$ était définit par

$$\varepsilon(h_j, \mathbf{d}) \leq \frac{1}{2} - \frac{c}{N} \tag{3.21}$$

avec c une constante absolue. Ceci veut dire que pour n'importe quelle pondération $\mathbf{d}$ un classificateur linéaire h_j peut être trouvé qui possède une marge $\geq c/N$. La garantie est donc donnée que pour l'ensemble H l'erreur empirique pondérée $\varepsilon(h_j, \mathbf{d})$ est inférieure à ½-½γ. Comme nous utilisons aussi des classificateurs linéaires comme classificateurs simples, nous avons la garantie que nos classificateurs simples sont des classificateurs faibles.

3.4 AdaBoost

3.4.1 Introduction

Dans ce chapitre, la méthode qui permet de combiner plusieurs hypothèses simples pour créer une autre hypothèse plus performante est présentée. La méthode s'appelle AdaBoost (Adaptive Boosting) qui est, comme son nom l'indique, un algorithme de Boosting.
L'idée de base du Boosting est de combiner des « règles » simples (hypothèses simples) pour créer un ensemble dont la performance de chaque élément de l'ensemble est améliorée (« boostée »). L'ensemble composé des hypothèses est défini de la manière suivante :

$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \tag{3.22}$$

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 de l'équation 3.16. 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 classifiés incorrectement et une faible pondération aux autres. Cette technique permet de concentrer la procédure d'apprentissage sur les exemples durs à classifier.
Les deux méthodes de Boosting les plus connues sont probablement AdaBoost et LogitBoost.
Dans le chapitre suivant (3.4.2), les idées de base du Boosting sont présentées. Ensuite, l'algorithme d'AdaBoost est expliqué en détail. Pour finir, il sera montré pourquoi l'erreur d'entraînement tend vers zéro.

3.4.2 Idées de base du Boosting

Dans ce chapitre, les idées de base du Boosting sont présentées pour un problème de classification binaire.
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. Ce problème peut être formulé de la façon suivante : En utilisant une base de données déjà classifiée S_1, il s'agit d'estimer une fonction $f : A \to B$ qui classifie correctement des exemples $(\mathbf{x}, y)$ inconnus. A est l'ensemble de tous les objets et B est l'ensemble des deux classes (labellisées par -1 et 1). Les éléments de la base de données, aussi appelés les données d'apprentissage, S_1 sont générés au hasard et indépendamment l'un de l'autre à partir d'une distribution de probabilité $P(\mathbf{x}, y)$. Lorsque $B = \{-1,1\}$, on parle d'un « hard classifier » et le label assigné à l'entrée $\mathbf{x}$ est donné par $f(\mathbf{x})$.

$$A = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{R}^d\} \tag{3.23}$$

$$B = \{-1,1\} \tag{3.24}$$

$$S_1 = \{(\mathbf{x}_1, y_1),...,(\mathbf{x}_N, y_N) \mid (\mathbf{x}_i, y_i) \in A \times B\} \tag{3.25}$$

La performance d'un classificateur f est définie par le risque $L(f)$, aussi appelé *erreur de généralisation* ou *erreur de test*.

$$L(f) = \int \lambda(f(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{3.26}$$

avec λ une fonction de coût à définir.
Le risque calcule l'erreur de classification en utilisant une base de données déjà classifiée S_2 qui est générée de la même façon que S_1. Il est important à noter que la base de données S_2 se compose d'autres exemples que ceux de la base de données S_1. Les éléments de la base de données S_2 sont aussi appelés données de test.

$$S_2 = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) | (\mathbf{x}_i, y_i) \in A \times B\} \tag{3.27}$$

Pour la classification binaire, la fonction de coût suivante est souvent utilisée :

$$\lambda(f(\mathbf{x}), y) = I(f(\mathbf{x}), y) \tag{3.28}$$

avec

$$I(f(\mathbf{x}), y) = \begin{cases} 1 & \text{si} \quad f(\mathbf{x}) \neq y \\ 0 & \text{sinon} \end{cases} \tag{3.29}$$

Selon le contexte, d'autres fonctions de coût peuvent être utilisées.
Malheureusement, le risque ne peut pas être minimisé directement car la distribution de probabilité $P(\mathbf{x}, y)$ n'est pas connue. Pour cette raison, une fonction similaire doit être estimée qui ressemble le plus possible à la fonction optimale $L(f)$. La fonction doit être estimée à l'aide des informations à disposition, c'est-à-dire sur la base de données déjà classifiée S_1 et les propriétés de la classe des fonctions $\mathbf{F}$ dont la solution f est choisit. Une solution simple, consiste à approximer le risque $L(f)$ par le risque empirique $\hat{L}(f)$:

$$\hat{L}(f) = \frac{1}{N} \sum_{n=1}^{N} \lambda(f(\mathbf{x}_n), y_n) \tag{3.30}$$

À partir du théorème des grands nombres, on peut s'attendre à ce que le risque empirique $\hat{L}(f)$ tend vers le risque $L(f)$ si N tend vers l'infini :

$$L(f) = \lim_{N \to \infty} \hat{L}(f) \tag{3.31}$$

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 $\mathbf{F}$ n'ose pas être trop complexe. Autrement, on peut trouver une fonction f qui possède une erreur de classification sur les données d'apprentissage S_1 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 $\mathbf{F}$) vers $L(f)$.

Dans le cas où la base de données S_1 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 $\mathbf{F}$. 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 y a pu être montré que sous certaines conditions la complexité de la classe des fonctions $\mathbf{F}$ 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 $\mathbf{F}$.

3.4.2 Algorithme

Le but de l'algorithme d'AdaBoost est de sélectionner les coefficients α_t et les hypothèses h_t pour former une hypothèse plus puissante $f_T(\mathbf{x})$.

Entrée : $S_1 = \{(\mathbf{x}_1, y_1),...,(\mathbf{x}_N, y_N)\}$ *avec* N *le nombre d'images d'apprentissage, le nombre d'itérations* T.

Initialisation :

$$d_n^t = \frac{1}{N} \quad \text{pour} \quad n = 1,...,N$$

Pour $t = 1,...,T$ **:**

1) *Entraîner les classificateurs « faibles »* h_t^j *en respectant les données d'entraînement pondérées.*

2) *Calculer l'erreur d'entraînement pondérée pour chaque classificateur « faible »* ε_t^j.

3) *Choisir le classificateur « faible » avec l'erreur d'entraînement pondérée minimale* $\rightarrow$ *hypothèse* h_t *et erreur d'entraînement pondérée* ε_t.

4) *Calculer le poids à attribuer à l'hypothèse* h_t *:*

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

5) *Mettre à jour les poids des données d'entraînement :*

$$d_n^{(t+1)} = \frac{d_n^{(t)} \exp\{-\alpha_t y_n h_t(\mathbf{x}_n)\}}{Z_t}$$

avec Z_t *la constante de normalisation. Elle prend la valeur nécessaire pour que*

$$\sum_{n=1}^{N} d_n^{(t+1)} = 1.$$

On interrompt si

$$\varepsilon_t = 0 \text{ ou } \varepsilon_t \geq \frac{1}{2} \text{ et mettre } T = t-1$$

Sortie : $f_T(\mathbf{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{t=1}^{T}\alpha_t} h_t(\mathbf{x})$

Algorithme 3.2 : Algorithme d'AdaBoost

L'algorithme d'AdaBoost reçoit comme entrée la base de données d'entraînement $S_1 = \{(\mathbf{x}_1, y_1),...,(\mathbf{x}_N, y_N)\}$ et le paramètre T qui indique combien d'hypothèses doivent être combinées au maximum pour construire l'hypothèse combinée plus puissante $H(\mathbf{x})$.
Pour la première itération, les poids d_i^t sont initialisés uniformément :

$$d_n^t = \frac{1}{N} \quad \text{pour} \quad n = 1,...,N \tag{3.32}$$

À chaque itération t, tous les classificateurs faibles $h_t^j \in \mathbf{H}$ sont entraînés (1). **H** est la classe des classificateurs faibles. La façon choisie pour entraîner les classificateurs faibles h_t^j est décrite dans 3.3.2. Une fois les classificateurs faibles h_t^j entraînés, l'erreur d'entraînement ε_t^j pour chaque classificateur faible h_t^j est calculée (2) :

$$\varepsilon_t^j = \sum_{n=1}^{N} d_n^t \cdot I(\mathbf{x}_n, y_n) \tag{3.33}$$

avec

$$I(\mathbf{x}_n, y_n) = \begin{cases} 1 & \text{pour} \quad y_n \neq h_t^j(\mathbf{x}_n) \\ 0 & \text{sinon} \end{cases} \tag{3.34}$$

Parmi tous les classificateurs faibles entraînés, AdaBoost choisit maintenant le meilleur (3). Il s'agit du classificateur qui possède l'erreur d'entraînement minimale.

$$\begin{matrix} h_t^j \rightarrow h_t \\ \varepsilon_t^j \rightarrow \varepsilon_t \end{matrix} \quad \text{si} \quad \varepsilon_t^j < \varepsilon_t^k \quad \text{pour tout } k \neq j \tag{3.35}$$

Après avoir choisi le meilleur classificateur h_t, AdaBoost calcule le poids α_t à attribuer au classificateur h_t (4). Le poids α_t est calculé de façon à minimiser la fonction coût

$$G^{AB}(\alpha) = \sum_{n=1}^{N} \exp\{- y_n (\alpha h_t(\mathbf{x}_n) + f_{t-1}(\mathbf{x}_n))\} \tag{3.36}$$

avec

$$f_{t-1}(\mathbf{x}_n) = \sum_{r=1}^{t-1} \alpha_r h_r(\mathbf{x}_n) \tag{3.37}$$

étant l'hypothèse combinée de l'itération précédente. Il a pu être montré dans [16] que α_t peut être calculé analytiquement. L'expression de α_t est :

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{3.38}$$

Pour finir, AdaBoost actualise la pondération des données d'apprentissage pour ce concentrer sur des exemples difficiles à classifier (5). Les exemples classifiés correctement reçoivent une pondération plus petite et celles classifiées incorrectement une pondération plus grande.

L'itération est interrompue, lorsque l'erreur d'entraînement ε_t du classificateur 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 classificateurs. Si $\varepsilon_t \geq 0{,}5$, alors le classificateur h_t ne satisfait plus aux conditions de classificateur faible. Ceci signifie que h_t classifie moins bien qu'une sélection aléatoire. L'hypothèse f_T ne peut donc plus être améliorée.

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

$$f_T(\mathbf{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{t=1}^{T} \alpha_t} h_t(\mathbf{x}) \tag{3.39}$$

L'hypothèse finale $H(\mathbf{x})$ est donnée par :

$$H(\mathbf{x}) = \text{sign}[f_T(\mathbf{x})] \tag{3.40}$$

3.4.4 La convergence de l'erreur d'entraînement vers zéro

Dans 3.3.4, il a été montré que l'erreur empirique pondérée $\varepsilon(h_j, \mathbf{d})$ de nos classificateurs faibles est inférieure à ½-½γ, $\gamma > 0$. On va montrer maintenant que cette condition suffit pour que l'erreur empirique de l'hypothèse composée f_T converge exponentiellement vers 0 lorsque le nombre d'itérations augmente.
On commence par rappeler que la valeur de la fonction réelle f_T contient de l'information sur la certitude que la donnée $\mathbf{x}$ appartienne à $\text{sign}[f_T(\mathbf{x})]$. La valeur de f_T est définie sur l'intervalle [-1,1]. On définit pour la classification binaire ($y \in \{-1,1\}$) et $f_T \in \mathbf{R}$ la marge de f_T comme :

$$\rho_n = y_n f_T(\mathbf{x}) \tag{3.41}$$

On considère la fonction suivante définie pour $0 \leq \Theta \leq 1/2$,

$$\varphi_\Theta(z) = \begin{cases} 1 & \text{si} \quad z \leq 0 \\ 1 - z/\Theta & \text{si} \quad 0 < z \leq \Theta \\ 0 & \text{sinon} \end{cases} \tag{3.42}$$

L'erreur de marge empirique est définie comme

$$\hat{L}^\Theta(f_T) = \frac{1}{N} \sum_{n=1}^{N} \varphi_\Theta(y_n f_T(\mathbf{x}_n)) \tag{3.43}$$

L'erreur de marge empirique $\hat{L}^{\Theta}(f_T)$ est égale à l'erreur de classification $\hat{L}(f_T)$ pour $\Theta = 0$. La fonction $\hat{L}^{\Theta}(f_T)$ est monotone décroissante en fonction de Θ. L'erreur de marge 0/1 $\tilde{L}^{\Theta}(f_T)$ est définie de la façon

$$\tilde{L}^{\Theta}(f_T) = \frac{1}{N}\sum_{n=1}^{N} I\left(y_n f(\mathbf{x}_n) \le \Theta\right) \tag{3.44}$$

Comme $\varphi_{\Theta}(yf(\mathbf{x})) \le I\left(y_n f(\mathbf{x}_n) \le \Theta\right)$, on peut conclure que $\hat{L}^{\Theta}(f_T) \le \tilde{L}^{\Theta}(f_T)$.

Théorème 1 *On considère AdaBoost comme décrit dans Algorithme 3.2. On suppose qu'à chaque itération t, l'erreur pondérée empirique satisfasse* $\varepsilon\left(h_t, \mathbf{d}^{(t)}\right) \le 1/2 - 1/2\gamma_t$. *Alors, l'erreur de marge empirique de l'hypothèse composée* f_T *obéis à*

$$\hat{L}^{\Theta}(f_T) \le \prod_{t=1}^{T} (1-\gamma_t)^{\frac{1-\Theta}{2}} (1+\gamma_t)^{\frac{1+\Theta}{2}} \tag{3.45}$$

Preuve. La preuve présentée est tirée de [16] pour le cas où $h_t \in \{-1,1\}$. On commence par montrer que pour chaque $\{\alpha_t\}$

$$\tilde{L}^{\Theta}(f_T) \le \exp\left(\Theta\sum_{t=1}^{T}\alpha_t\right)\left(\prod_{t=1}^{T} Z_t\right). \tag{3.46}$$

Par définition

$$\begin{aligned} Z_t &= \sum_{n=1}^{N} d_n^{(t)} e^{-y_n \alpha_t h_t(\mathbf{x}_n)} \\ &= \sum_{n:y_n = h_t(\mathbf{x}_n)} d_n^{(t)} e^{-\alpha_t} + \sum_{n:y_n \ne h_t(\mathbf{x}_n)} d_n^{(t)} e^{\alpha_t} \\ &= (1-\varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}. \end{aligned}$$

De la définition de f_T il suit que

$$yf_T(\mathbf{x}) \le \Theta \;\Rightarrow\; \exp\left(-y\sum_{t=1}^{T}\alpha_t h_t(\mathbf{x}) + \Theta\sum_{t=1}^{T}\alpha_t\right) \ge 1,$$

se qui peut être réécrit comme

$$I\left[Yf_T(\mathbf{X}) \le \Theta\right] \le \exp\left(-y\sum_{t=1}^{T}\alpha_t h_t(\mathbf{x}) + \Theta\sum_{t=1}^{T}\alpha_t\right). \tag{3.47}$$

Il est à remarquer que

$$d_n^{(T+1)} = \frac{d_n^{(T)} \exp\left(-\alpha_T y_n h_T(\mathbf{x}_n)\right)}{Z_T}$$
$$= \frac{\exp\left(-\sum_{t=1}^{T} \alpha_t y_n h_t(\mathbf{x}_n)\right)}{N \prod_{t=1}^{T} Z_T} \tag{3.48}$$

En utilisant (3.47) et (3.48) on trouve que

$$\tilde{L}^{\Theta}(f) = \frac{1}{N} \sum_{n=1}^{N} I\left[y_n f_T(\mathbf{x}_n) \le \Theta\right]$$
$$\le \frac{1}{N} \sum_{n=1}^{N} \left[\exp\left(- y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) + \Theta \sum_{t=1}^{T} \alpha_t \right) \right]$$
$$= \frac{1}{N} \exp\left(\Theta \sum_{t=1}^{T} \alpha_t \right) \sum_{n=1}^{N} \exp\left(- y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) \right)$$
$$= \exp\left(\Theta \sum_{t=1}^{T} \alpha_t \right) \left(\prod_{t=1}^{T} Z_t \right) \underbrace{\sum_{n=1}^{N} d_n^{(T+1)}}_{=1}$$
$$= \exp\left(\Theta \sum_{t=1}^{T} \alpha_t \right) \left(\prod_{t=1}^{T} Z_t \right).$$

Maintenant, on pose $\alpha_t = (1/2)\log((1-\varepsilon_t)/\varepsilon_t)$ (voir algorithme 3.2) ce qui implique que

$$Z_T = 2\sqrt{\varepsilon_t (1-\varepsilon_t)}.$$

En substituant ce résultat dans (3.46) on trouve que

$$\tilde{L}^{\Theta}(f) \le \prod_{t=1}^{T} \sqrt{4\varepsilon_t^{1-\Theta} (1-\varepsilon_t)^{1+\Theta}}$$

ce qui produit le résultat cherché en notant que $\varepsilon_t = 1/2 - \gamma_t / 2$ et $\hat{L}^{\Theta}(f) \le \tilde{L}^{\Theta}(f)$.
Dans le cas spécial où $\Theta = 0$ on trouve que

$$\hat{L}^{\Theta}(f_T) \le e^{-\sum_{t=1}^{T} \gamma_t^2 / 2}, \tag{3.49}$$

d'où on conclut que la condition $\sum_{t=1}^{T} \gamma_t^2 \to \infty$ suffit pour garantir que $\hat{L}(f_T) \to 0$. Par exemple, la condition $\gamma_t \ge c/\sqrt{t}$ suffit. Dans le cas $\gamma_t \ge \gamma_0 > 0$, $\hat{L}^{\Theta}(f_T) \to 0$ pour tout $\Theta \le \gamma_0 / 2$. Ceci signifie que si chaque classificateur est légèrement meilleur qu'un

classificateur qui classifie au hasard alors l'erreur d'entraînement chute exponentiellement vers 0.

3.5 Arbitrage

3.5.1 Introduction

Un classificateur indique normalement pour plusieurs régions dans l'image la présence d'un objet caractéristique. Entre ces détections, il y a des fausses détections qui ne contiennent pas d'objet caractéristique et des détections qui contiennent toutes le même objet caractéristique. Il est donc nécessaire de faire un arbitrage pour trouver la région aillant la plus grande chance de contenir l'objet caractéristique.
Un premier arbitrage qui est utilisé est d'appliquer le détecteur de l'œil droit seulement dans la région supérieure gauche, le détecteur de l'œil gauche seulement dans la région supérieure droite et le détecteur de la bouche seulement dans la région inférieure. Mais cet arbitrage ne suffit pas. Un deuxième arbitrage est donc encore utilisé. Pour le deuxième arbitrage, trois méthodes différentes sont implémentées et testées. Ces trois méthodes sont présentées en bref dans les sections suivantes de ce chapitre.

3.5.2 Première méthode pour arbitrer

La première méthode pour arbitrer consiste à choisir la position avec la valeur de détection maximale comme position la plus probable pour un objet caractéristique. On espère donc que le classificateur obtient la valeur de détection maximale pour la région qui est exactement centrée sur l'objet caractéristique.

3.5.3 Deuxième méthode pour arbitrer

La deuxième méthode pour arbitrer consiste à chercher la région contenant le plus de détections et de choisir la position avec la valeur de détection maximale à l'intérieure de cette région comme position la plus probable pour un objet caractéristique. Cette méthode par de l'idée qu'on a plusieurs détections voisines pour des régions partiellement superposées contenant toutes le même objet caractéristique et que ces régions voisines forment la région contenant le plus de détections.

3.5.4 Troisième méthode pour arbitrer

La troisième méthode pour arbitrer consiste à comparer les positions des détections pour les trois objets caractéristiques avec un modèle géométrique du visage et de choisir les trois positions respectant le mieux le modèle comme positions les plus probables pour les objets caractéristiques. Le modèle géométrique du visage utilisé est assez primitif et consiste en seulement trois points représentant le centre de l'œil gauche c_oeilG, le centre de l'œil droit c_oeilD et le centre de la bouche c_bouche.

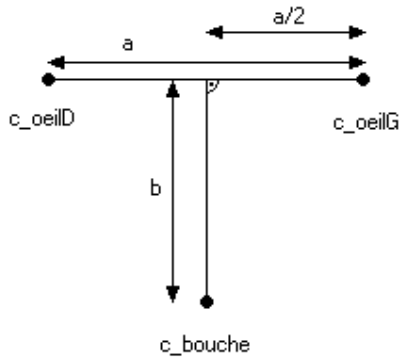

FIG. 3.5 – Image illustrant le modèle du visage utilisé.

a et b sont des constantes auxquelles sont attribuées les valeurs suivantes :

$$a = 14\,[\text{pixel}] \tag{3.50}$$

$$b = 1{,}07 \cdot 14 \approx 15\,[\text{pixel}] \tag{3.51}$$

Ces valeurs ont été trouvées à l'aide de mesures faites sur des visages et sur la base d'essais. Pour pouvoir mesurer la correspondance entre le modèle du visage et les trois points de détections pour les objets caractéristiques la fonction de probabilité de correspondance $p\left(E_{hor}, E_{ver}, E_{eye\_dist}, E_{center}\right)$ est utilisée :

$$p\left(E_{hor}, E_{ver}, E_{eye\_dist}, E_{center}\right) = e^{-\frac{E_{hor}^2}{VAR\_HOR}} \cdot e^{-\frac{E_{ver}^2}{VAR\_VER}} \cdot e^{-\frac{E_{eye\_dist}^2}{VAR\_EYED}} \cdot e^{-\frac{E_{center}^2}{VAR\_CENTER}} \tag{3.52}$$

avec E_{hor}, E_{ver}, E_{eye_dist} les distances d'erreur indiquées dans la figure suivante.

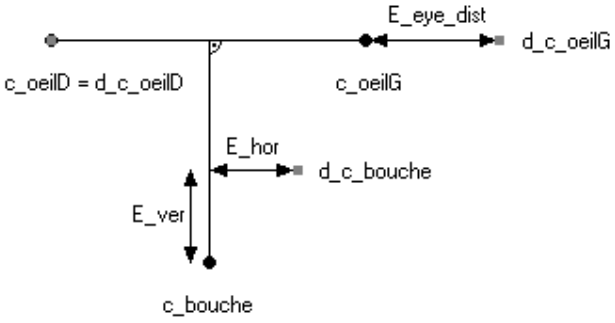

FIG. 3.6 – Image illustrant les distances d'erreur E_{hor}, E_{ver}, E_{eye_dist}.

d_c_oeilD, d_c_oeilG et d_c_bouche sont les centres des détections pour les objets caractéristiques.

E_{center} est une distance d'erreur qui indique de combien le visage formé par les trois détections d_c_oeilD, d_c_oeilG et d_c_bouche est éloigné du centre de l'image contenant le visage. Elle sert à favoriser les ensembles de détections qui correspondent au modèle du visage et qui sont centré dans l'image. Intuitivement, il est correct de favoriser ces détections, car le visage à localiser est normalement aussi centré dans l'image. Définissons $\mathbf{d}_{c\_oeilD}$, $\mathbf{d}_{c\_oeilG}$ et $\mathbf{d}_{c\_bouche}$ comme étant les coordonnées des centres des détections pour l'œil droit, l'œil gauche et la bouche et $\mathbf{c}_{image}$ comme étant la coordonnée du centre de l'image contenant le visage à localiser. Alors E_{center} est défini de la façon suivante :

$$E_{center} = \left\| \mathbf{c}_{center} - \frac{1}{3}\left(\mathbf{d}_{c\_oeilD} + \mathbf{d}_{c\_oeilG} + \mathbf{d}_{c\_bouche}\right) \right\| \tag{3.53}$$

Les valeurs pour les constantes VAR_HOR, VAR_VER, VAR_EYED et VAR_CENTER doivent être trouvées en faisant des essais avec le programme implémenté.

4 Résultats et Expérimentations

4.1 Introduction

Dans ce chapitre 4, les résultats obtenus pour l'apprentissage avec AdaBoost et les performances du programme de localisation du visage sont présentées.
Avant d'expliquer la structure du programme de localisation implémenté, commençons par appeler par convention œil gauche l'œil gauche de la personne et non l'œil situé à gauche sur l'image.
Pour travailler avec des images de tailles 40x30 pixels, un algorithme de normalisation d'images est implémenté dans le programme.
Ensuite, pour détecter les trois objets caractéristiques, trois classificateurs différents sont implémentés :

- un pour l'œil droit,
- un pour l'œil gauche
- et un pour la bouche.

Les classificateurs pour l'œil droit et la bouche sont construit à l'aide de l'algorithme AdaBoost en utilisant des masques visuels pour construire les descripteurs. Comme l'œil gauche est symétrique à l'œil droit, on utilise le classificateur de l'œil droit aussi pour l'œil gauche et réfléchit les masques visuels à l'axe vertical.
Pour localiser les objets caractéristiques, on fait glisser une fenêtre dans l'image du visage (voir FIG. 2.1). À chaque position, la partie de l'image qui est enfermée par la fenêtre est extrait et classifié.
Comme chaque classificateur retourne normalement plusieurs positions pour l'objet caractéristique qu'il doit détecter, des arbitrages sont utilisés à la fin pour trouver la position la plus probable des objets caractéristiques. Le premier arbitrage implémenté consiste à appliquer, à cause de la nature du visage, le classificateur pour l'œil droit seulement dans la région supérieure gauche, le classificateur pour l'œil gauche seulement dans la région supérieure droite et le classificateur pour la bouche seulement dans la partie inférieure de l'image. Cet arbitrage est suivi par un deuxième arbitrage cherchant après la position la plus probable des objets caractéristiques dans les différentes régions. Pour cela, trois méthodes différentes sont implémentées :

- La première méthode consiste à choisir la position avec la valeur de détection maximale comme position la plus probable pour un objet caractéristique.
- La deuxième méthode consiste à chercher la région contenant le plus de détections et de choisir la position avec la valeur de détection maximale à l'intérieure de cette région comme position la plus probable pour un objet caractéristique.
- La troisième méthode consiste à comparer les positions des détections pour les trois objets caractéristiques avec un modèle géométrique du visage et de choisir les trois positions respectant le mieux le modèle comme positions les plus probables pour les objets caractéristiques.

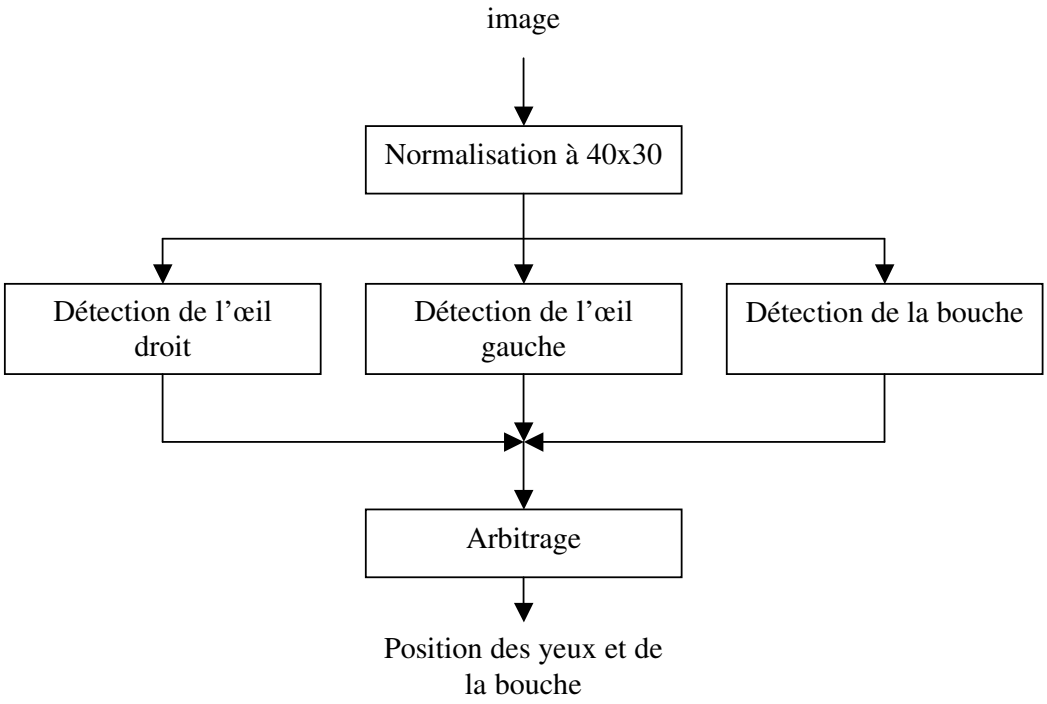

FIG. 4.1 – Structure du programme implémenté.

Le chapitre est subdivisé de la façon suivante : Dans 4.2, la base de données utilisée est présentée. Cette section est suivie par le chapitre présentant des résultats de la performance d'AdaBoost. Pour conclure, les résultats de détection sont présentés.

4.2 Base de données utilisée

La base de données utilisée pour l'entraînement et les tests s'appelle BANCA [17]. BANCA est une base de données standard pour évaluer des algorithmes de localisation. Elle contient des images de 52 personnes différentes vues de face. L'avantage de cette base de données est qu'on connaît la position des yeux dans l'image et que les images des visages ont été prises dans des conditions d'acquisition très variées. Par exemple, les personnes ne regardent pas toujours droit dans la caméra, ont la bouche ouverte ou fermée etc. L'inconvénient de cette base de données est que les personnes sont souvent illuminées du même côté et qu'on ne connaît pas la position de la bouche. D'autres bases de données standards pour tester des algorithmes de localisation sont par exemple XM2VTS [18] ou BioID [19].
Lors de la localisation, on a normalement une image contenant seulement le visage d'une personne comme entrée. Comme la base de données BANCA contient des images de personnes avec leur fond, il faut d'abord extraire les visages des personnes. Pour cela, le même modèle du visage que dans [20] est utilisé pour connaître à partir de la position des yeux la région contenant le visage. Le visage est modélisé par un rectangle et deux points représentant les pupilles des yeux. La position des deux points dans ce rectangle est donnée par des mesures antropologiques reprises de [21]. L'image suivante illustre le modèle du visage.

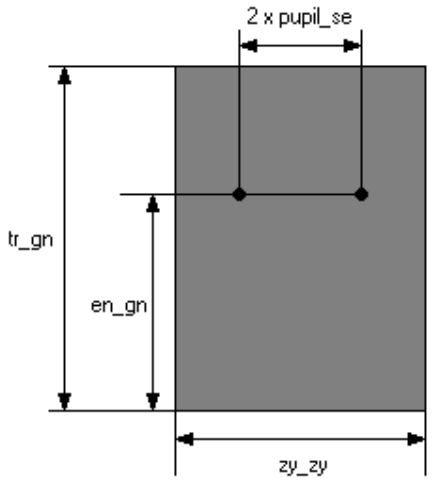

FIG. 4.2 – Modèle du visage construit à partir des mesures antropologiques.

Les constantes $pupil\_se$ (distance entre la pupille et le mieux du visage), en_gn (hauteur de la partie inférieure du visage), tr_gn (hauteur du visage) et zy_zy (largeur du visage) sont reprises de [21] et ont comme valeur :

$$\begin{aligned} pupil\_se &= 33{,}4 \\ en\_gn &= 117{,}7 \\ tr\_gn &= 187{,}2 \\ zy\_zy &= 139{,}1 \end{aligned} \tag{4.1}$$

Avec le modèle du visage présenté, il est possible de calculer à partir de la position des yeux les grandeurs h (hauteur de la région à extraire), w (largeur de la région à extraire) et y_upper (la hauteur de la partie supérieure du visage) qui déterminent la région à extraire d'une image contenant un visage. Les trois grandeurs se calculent de la façon suivante :

$$w = \frac{x\_ee}{2 \cdot pupil\_se} \cdot zy\_zy \tag{4.2}$$

$$h = \frac{w}{zy\_zy} \cdot tr\_gn \tag{4.3}$$

$$y\_upper = h - \frac{w}{zy\_zy} \cdot en\_gn \tag{4.4}$$

avec x_ee la distance entre les yeux mesurée en pixels.

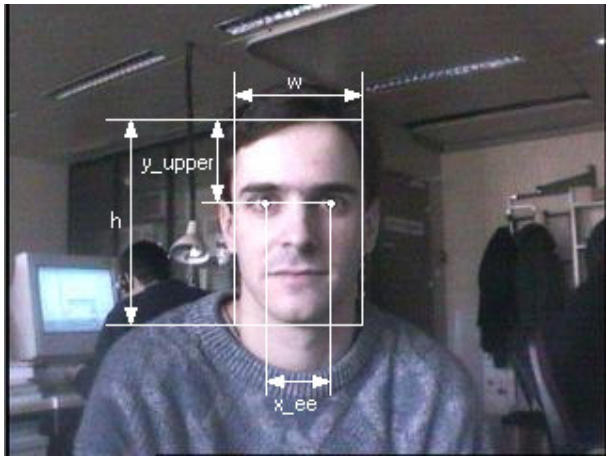

FIG. 4.3 – Région à extraire construite à partir des centres des yeux. L'image est similaire aux images appartenant à la base de données BANCA.

Après avoir extrait les visages des images des 52 personnes de la base de données BANCA, les visages sont repartis en deux groupes G1 et G2 pour entraîner et tester les classificateurs. Chaque groupe contient 1000 images de visages appartenant à 26 personnes différentes. Le groupe G1 est utilisé pour créer les ensembles d'entraînement TRAIN_OD et TRAIN_B qui sont nécessaires pour entraîner le classificateur de l'œil droit et de la bouche. Le groupe G2 est utilisé pour créer les ensembles de test TEST_OD et TEST_B qui servent à tester le classificateur de l'œil droit et de la bouche. Un ensemble est constitué d'images positives et négatives. Les images positives sont les images de l'objet caractéristique à localiser et sont extraites à un endroit précis dans les images du groupe G1 ou G2. Les images négatives sont les images ne représentant pas l'objet caractéristique à localiser et sont extraites à une position choisie aléatoirement dans les images du groupe G1 ou G2. Le tableau suivant montre comment les différents ensembles pour les différents objets caractéristiques sont assemblés.

Objet caractéristique	Œil droit		Bouche	
Ensemble	TRAIN_OD	TEST_OD	TRAIN_B	TEST_B
Groupe	G1	G2	G1	G2
Nombre d'images positives	1000	1000	300	1000
Nombre d'images négatives	2000	2000	500	1200

TAB. 4.1 – Composition des ensembles TRAIN_OD, TRAIN_B, TEST_OD et TEST_B.

Il n'y a pas d'ensemble d'entraînement et de test pour l'œil gauche par-ce qu'on n'a pas entraîné un classificateur pour celui-ci.

4.3 Taille idéale des images

Quelle est la taille idéale des images contentant un visage pour localiser les objets caractéristiques? Pour trouver une réponse à cette question, six classificateurs différents pour l'œil droit sont entraînés pour six tailles d'images différentes :

- 12x9
- 20x15
- 40x30
- 60x45
- 80x60
- 120x90

FIG. 4.4 – Images à la taille 12x9, 20x15, 40x30, 60x45, 80x60 et 120x90.

La figure 4.5 montre l'erreur de classification des différents classificateurs. L'erreur de classification ε_r est calculée en utilisant l'ensemble TEST_OD et est donnée par :

$$\varepsilon_r = \frac{1}{N}\sum_{n=1}^{N} I\left(y_n h(\mathbf{x}_n) \le 0\right)$$

avec N le nombre d'images positives et négatives de l'ensemble TEST_OD.

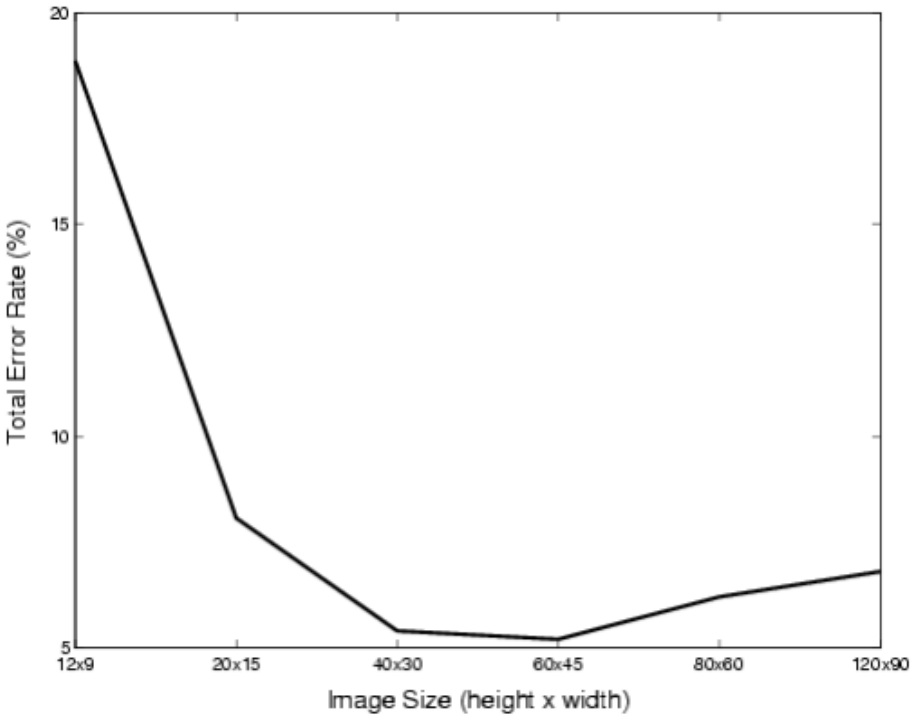

FIG. 4.5 – Erreur de classification de l'œil droit pour les tailles des images du visage 12x9, 20x15, 40x30, 60x45, 80x60 et 120x90.

On voit qu'en augmentant la résolution des images, l'erreur de classification diminue jusqu'à la résolution de 60x45. Intuitivement c'est logique, car en augmentant la résolution on augmente aussi l'information dans l'image ce qui permet de mieux classifier des images. La raison pour laquelle l'erreur de classification remonte à partir de la résolution 60x45 est que les régions extraites pour entraîner le classificateur ont été choisies, par rapport à la taille du visage, plus petites à cause du temps de calcul. Ceci signifie que les images positives contiennent juste l'œil droit et plus l'œil droit plus une petite région autour de l'œil.
Pour diminuer le temps de calcul, il est intéressant d'avoir une résolution aussi faible que possible. C'est la raison pour laquelle on a choisi de travailler avec des images contenant un visage de taille 40x30, vu que l'erreur de classification ne diminue plus significativement à partir de cette taille.

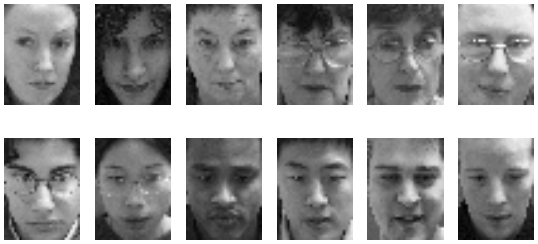
FIG. 4.6 – Images de taille 40x30. Les images appartiennent à la base de données BANCA.

La figure suivante illustre les courbes ROC du classificateur de l'œil droit pour les tailles des images du visage 12x9, 20x15, 40x30, 60x45, 80x60 et 120x90.

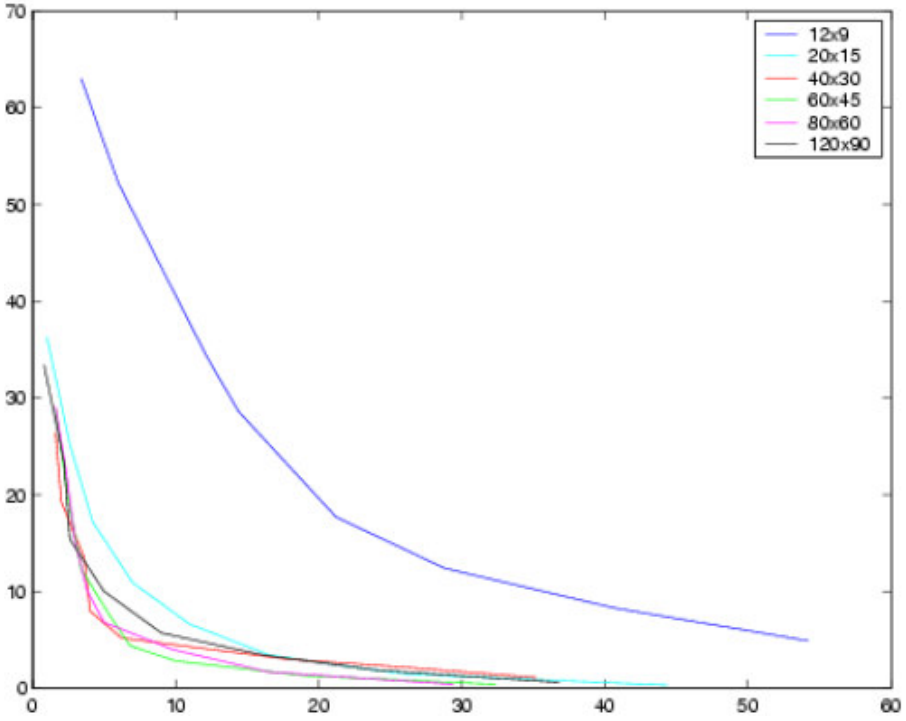

FIG. 4.7 – Courbes ROC du classificateur de l'œil droit pour les tailles des images du visage 12x9, 20x15, 40x30, 60x45, 80x60 et 120x90.

4.4 Résultats de la performance des classificateurs

Comme expliqué dans 1.5, la performance d'un classificateur peut être mesurée à l'aide de plusieurs critères. Ici, l'erreur de classification ε_r est utilisée. La figure suivante montre l'erreur de classification du classificateur entraîné pour l'œil droit. L'erreur de classification est calculée pour l'ensemble TRAIN_OD et TEST_OD. Comme indique dans le tableau 4.1, 1000 images positives et 2000 images négatives ont été utilisées pour l'ensemble TRAIN_OD et 1000 images positives et 2000 images négatives pour l'ensemble TEST_OD.

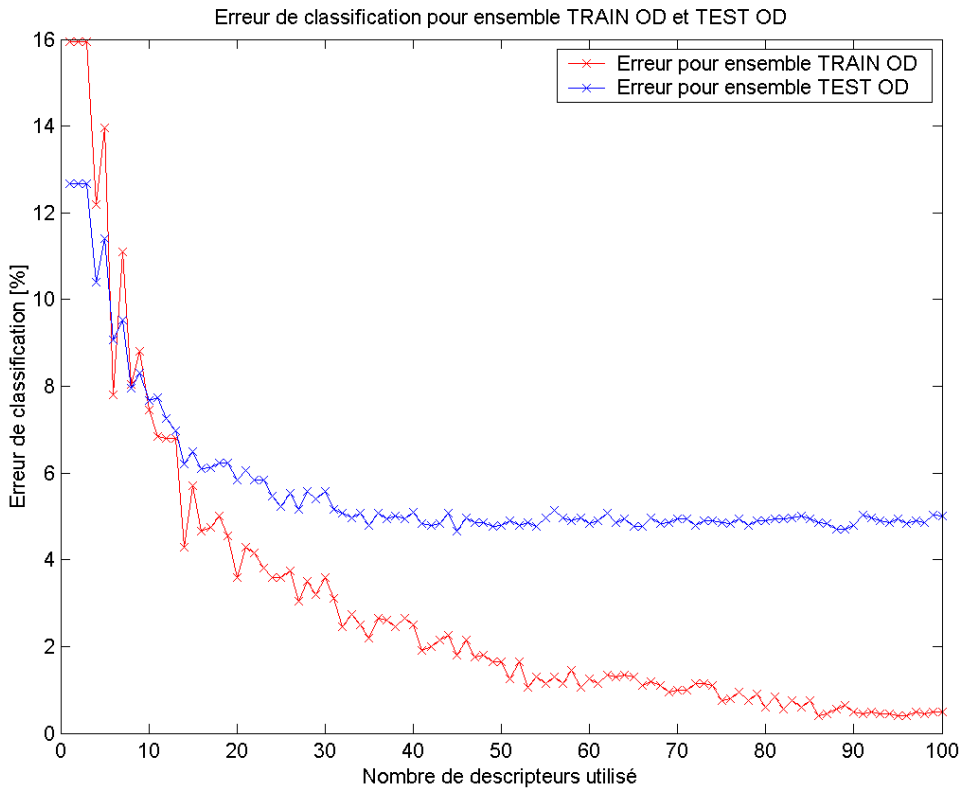

FIG. 4.8 – Erreur de classification pour l'œil droit. Elle est représentée en fonction du nombre de descripteurs utilisés pour l'ensemble TEST_OD et TRAIN_OD.

Il est intéressant de voir comment l'erreur de classification pour l'ensemble TRAIN_OD tend vers zéro lorsque le nombre de descripteurs utilisés augmente. C'est un effet attendu qui a été démontré en théorie (voir chapitre 3.4.4). Cette courbe permet aussi de voir comment AdaBoost sélectionne itérativement des classificateurs faibles pour diminuer continuellement l'erreur de classification ε_r des données d'entraînement TRAIN_OD.

L'erreur de classification ε_r des données TEST_OD montre qu'AdaBoost permet effectivement de booster un classificateur faible. On voit que le meilleur classificateur faible, le classificateur qui est sélectionné le premier par AdaBoost, a une erreur de classification ε_r des données TEST_OD d'environ 13%. AdaBoost permet de trouver un classificateur réduisant l'erreur de classification ε_r des données TEST_OD à environ 5,5%.

La figure suivante montre les courbes ROC pour l'œil droit pour les deux ensembles TRAIN_OD et TEST_OD. Les courbes ROC sont calculées en faisant varier le seuil de décision du classificateur pour l'œil droit.

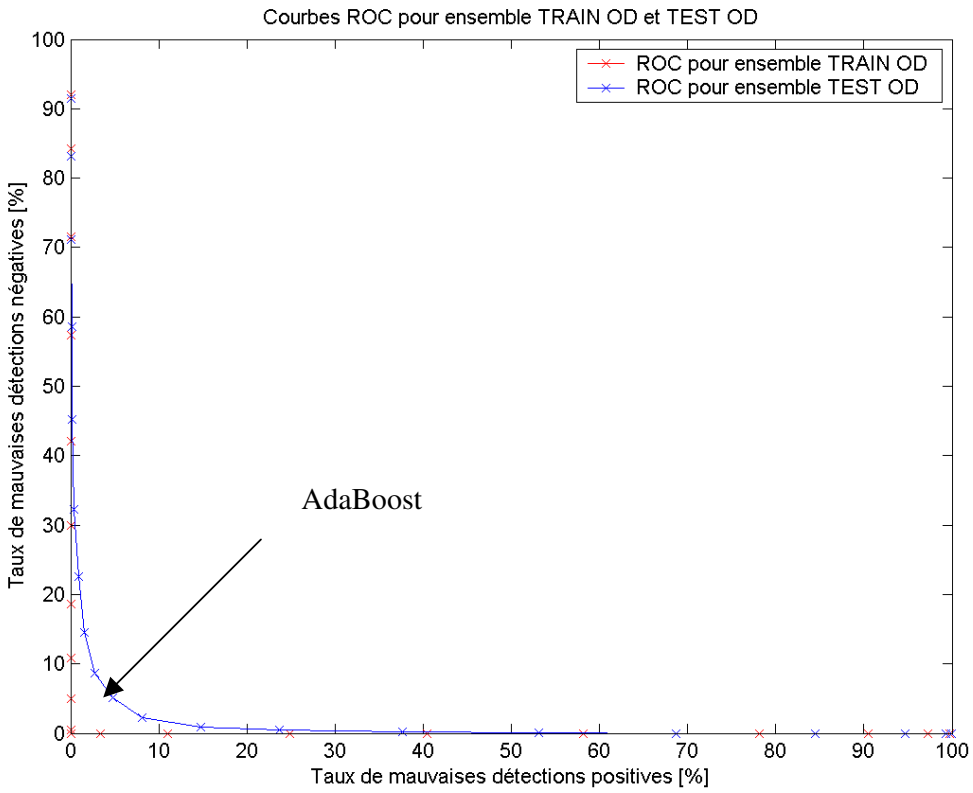

FIG. 4.9 – Courbes ROC pour l'œil droit pour les ensembles TRAIN_OD et TEST_OD.

La courbe ROC pour l'ensemble TEST_OD permet de voir qu'il n'est pas possible de trouver un classificateur pour l'œil droit qui possède en même temps un petit taux de mauvaises détections positives et un petit taux de mauvaises détections négatives. Le classificateur pour l'œil droit crée par AdaBoost peut être trouvé sur la courbe ROC pour l'ensemble TEST_OD à l'endroit indiqué par la flèche. Il s'agit du point le plus proche de l'origine. Ce point a la propriété d'avoir l'erreur de classification minimale.

4.5 Résultats de localisation

Dans cette section, les résultats obtenus pour le programme implémenté sont présentés. Comme décrit dans 4.1, 3 différentes méthodes d'arbitrage ont été testées. Pour chaque méthode les résultats obtenus sont présentés.
Les résultats sont obtenus en appliquant le programme de localisation implémenté sur le groupe d'images G2. Dans la suite du chapitre, des histogrammes de détections seront présentés. Il s'agit d'histogrammes de détections pour le centre des objets caractéristiques. Il y aura aussi des erreurs de détections e_{ij} qui seront présenté. Ces erreurs de détections indiquent le pourcentage des détections $\mathbf{d}$ pour un objet caractéristique qui se trouvent trop éloignées du centre $\mathbf{c}$ connu. Comme on utilise le groupe G2, la position du centre des yeux ($\mathbf{c}_{oeil\_droit}$ et $\mathbf{c}_{oeil\_gauche}$) est connue. La position du centre de la bouche $\mathbf{c}_{bouche}$ est estimée en prenant la moyenne sur un ensemble de mesures faites sur les images. Les centres des objets caractéristiques utilisés ont les coordonnées suivantes :

$$\mathbf{c}_{oeil\_droit} = \begin{pmatrix} 16 \\ 9 \end{pmatrix}$$

$$\mathbf{c}_{oeil\_gauche} = \begin{pmatrix} 16 \\ 20 \end{pmatrix}$$

$$\mathbf{c}_{bouche} = \begin{pmatrix} 32 \\ 14 \end{pmatrix}$$

Une détection $\mathbf{d}$ est classifiée comme correcte si :

$$|\mathbf{c}[0] - \mathbf{d}[0]| \leq i$$

et

$$|\mathbf{c}[1] - \mathbf{d}[1]| \leq j$$

avec i et j les indices de l'erreur de détection e_{ij}.

4.5.1 Localisation avec la première méthode pour arbitrer

La première méthode pour arbitrer cherche après la position la plus probable des objets caractéristiques en choisissant la position avec la valeur de détection maximale comme position la plus probable pour un objet caractéristique.
Commençons par la présentation des histogrammes de détections et les différentes erreurs de détections e_{ij} pour l'œil droit, l'œil gauche et la bouche. On rappelle que les histogrammes indiquent la position du centre des détections.

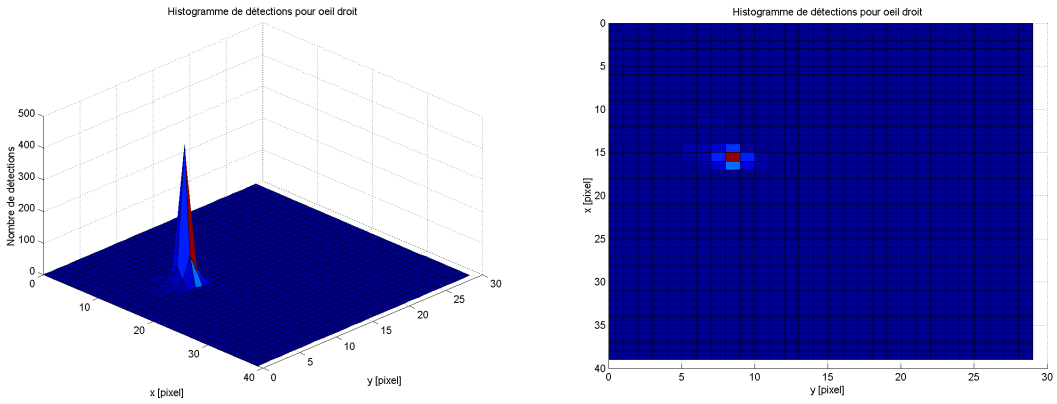

FIG. 4.10 – Histogramme de détections pour l'œil droit. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	96,5
e_{11}	35,8
e_{22}	13,7
e_{33}	8,1
e_{44}	3,9

TAB. 4.2 – Erreur de détections pour l'œil droit.

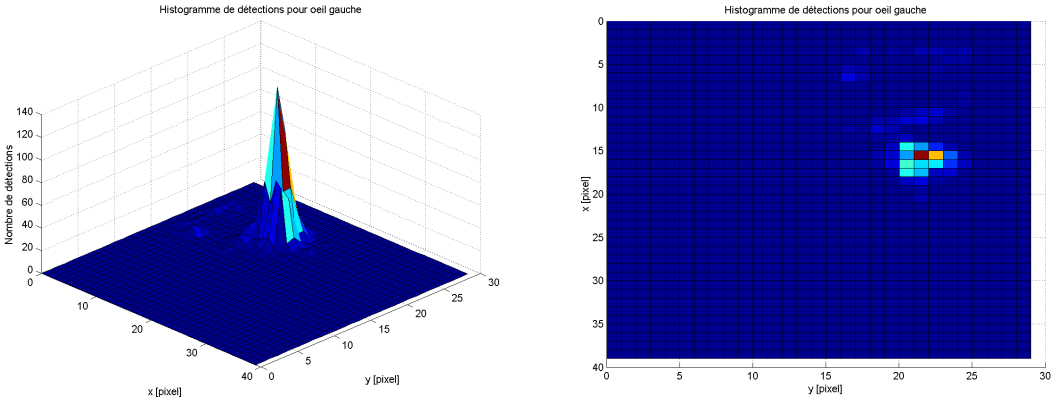

FIG. 4.11 – Histogramme de détections pour l'œil gauche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	94,2
e_{11}	61,1
e_{22}	31,3
e_{33}	22,2
e_{44}	17,4

TAB. 4.3 – Erreur de détections pour l'œil gauche.

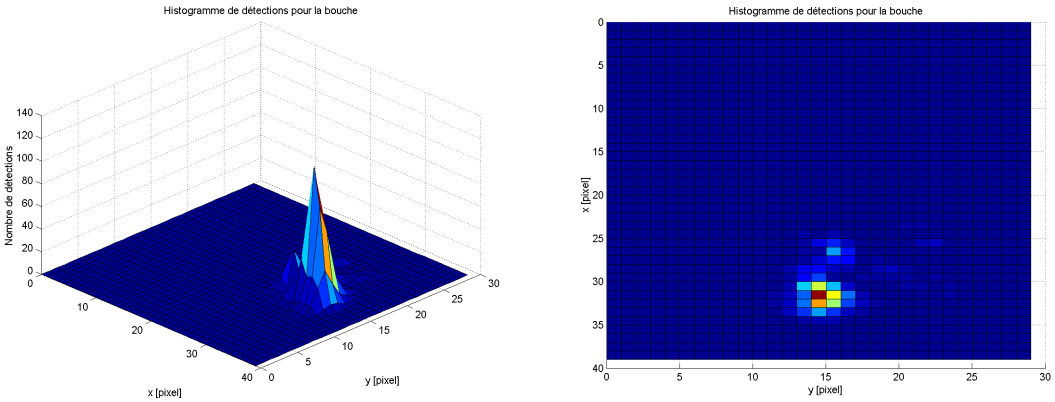

FIG. 4.12 – Histogramme de détections pour la bouche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	91,6
e_{11}	54,5
e_{22}	30,9
e_{33}	24,7
e_{44}	19,3

TAB. 4.4 – Erreur de détections pour la bouche.

On voit que l'œil droit est assez bien détecté. Il y a seulement 8,1% de détections qui sont éloigné plus que 3 pixels du centre $\mathbf{c}_{oeil\_droit}$.

L'œil gauche par contre est détecté très mal. À partir de l'histogramme, on voit qu'on a des mauvaises détections à l'endroit où se trouve le sourcil gauche et dans la région des cheveux. La mauvaise performance du classificateur pour l'œil gauche peut être expliquée par le fait qu'on a simplement reprit le classificateur de l'œil droit et réfléchit les masques visuels à l'axe vertical. Comme les personnes dans la base de données BANCA sont souvent illuminées de gauche, l'œil gauche apparaît souvent différemment que l'œil droit. Ceci est probablement la raison principale pour la mauvaise performance du classificateur de l'œil gauche.

On voit que l'histogramme pour la bouche possède une assez grande variance dans la direction horizontale et horizontale. La raison pour cela est probablement que la bouche ne se trouve pas à une position fixe dans la base de donnée BANCA. Ce qui n'est pas bien visible dans la figure 4.12 est qu'on a pas mal de détections pour la partie inférieure du nez. Apparemment, cette partie ressemble beaucoup à une bouche.

Les images suivantes montrent des exemples de mauvaises et bonnes détections pour l'œil droit, l'œil gauche et la bouche.

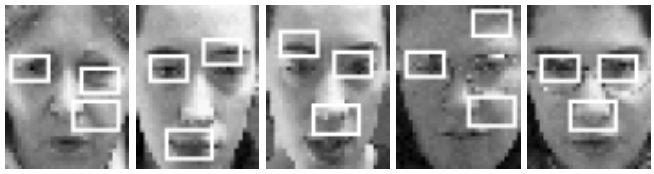

FIG. 4.13 – Mauvaises détections.

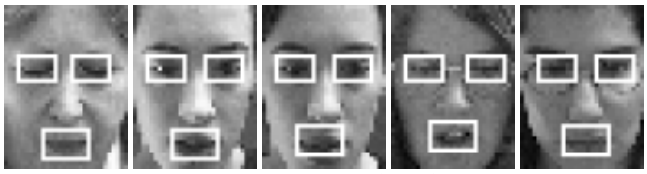

FIG. 4.14 – Bonnes détections.

4.5.2 Localisation avec la deuxième méthode pour arbitrer

La deuxième méthode pour arbitrer cherche après la position la plus probable des objets caractéristiques en cherchant la région contenant le plus de détections et en choisissant la position avec la valeur de détection maximale à l'intérieure de cette région comme position la plus probable pour un objet caractéristique.

Commençons de nouveau par la présentation des histogrammes de détections et les différentes erreurs de détections e_{ij} pour l'œil droit, l'œil gauche et la bouche.

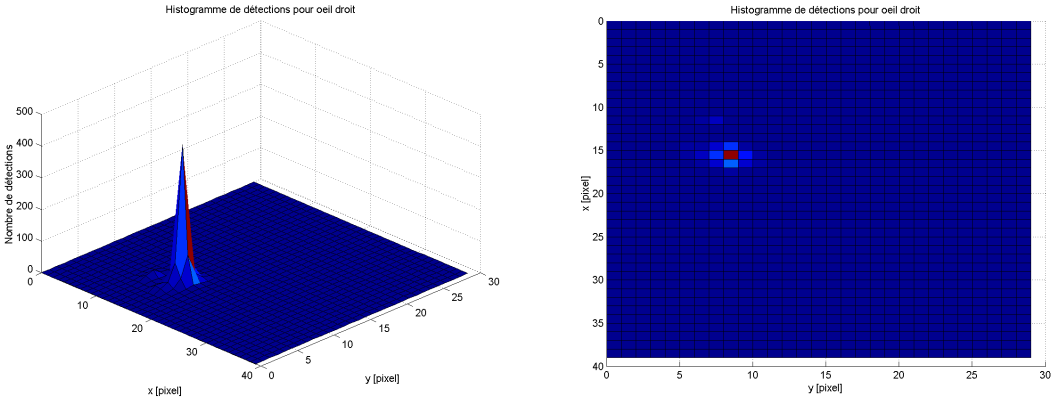

FIG. 4.15 – Histogramme de détections pour l'œil droit. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	97,3
e_{11}	37,8
e_{22}	14,9
e_{33}	10
e_{44}	6,4

TAB. 4.5 – Erreur de détections pour l'œil droit.

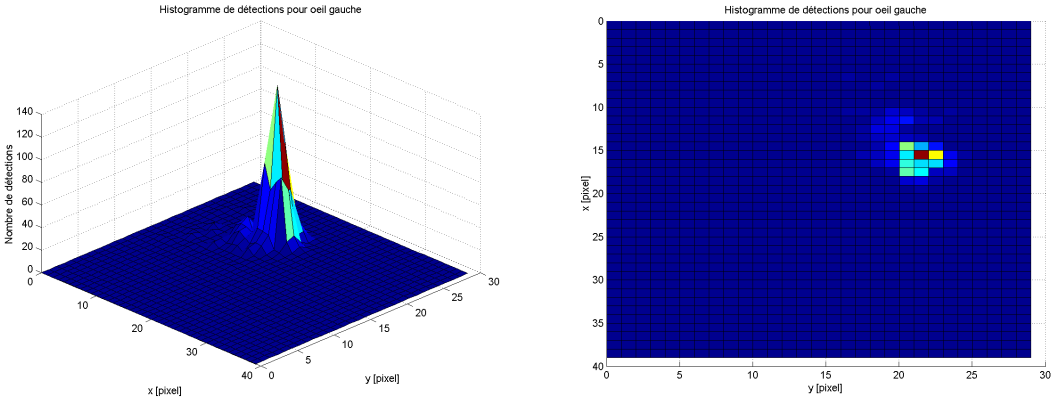

FIG. 4.16 – Histogramme de détections pour l'œil gauche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	94,4
e_{11}	57,8
e_{22}	25,7
e_{33}	17,2
e_{44}	12,8

TAB. 4.6 – Erreur de détections pour l'œil gauche.

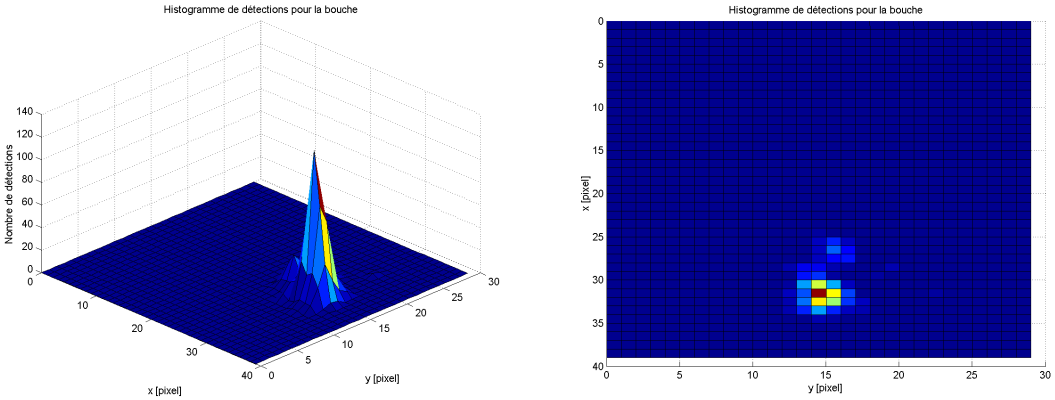

FIG. 4.17 – Histogramme de détections pour la bouche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	91,6
e_{11}	51,3
e_{22}	27,6
e_{33}	21,1
e_{44}	16

TAB. 4.7 – Erreur de détections pour la bouche.

Par rapport à la première méthode, on voit que la performance de détection pour l'œil droit est moins bonne. Les différentes erreurs de détections ont augmenté légèrement. On doit donc conclure que les régions contenant le plus de détections ne sont pas toujours centrées sur le centre connu $\mathbf{c}_{oeil\_droit}$.

La performance de détection pour l'œil gauche est mieux que celle de la première méthode d'arbitrage. On voit à l'aide de l'histogramme, que surtout les détections dans la région des cheveux ont diminué. Probablement, les détections dans cette région sont des détections isolées qui sont filtrées lorsqu'on se concentre sur la région contenant le plus de détections.

La performance de détection pour la bouche est aussi mieux que celle de la première méthode d'arbitrage. Ce sont surtout les détections pour la joue gauche qui ont disparu. La raison doit être la même que celle mentionnée pour l'amélioration de la performance de détection de l'œil gauche : il doit s'agir de détections isolées.

En gros, on peut dire qu'on obtient légèrement un meilleur résultat en utilisant la deuxième méthode d'arbitrage.

La figure suivante montre de nouveau quelques images de bonnes et de mauvaises détections.

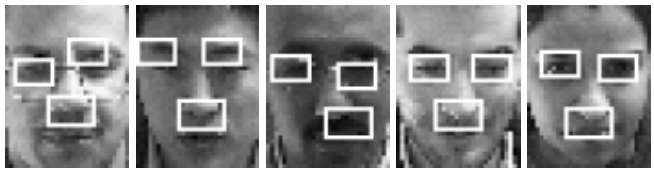

FIG. 4.18 – Mauvaises détections.

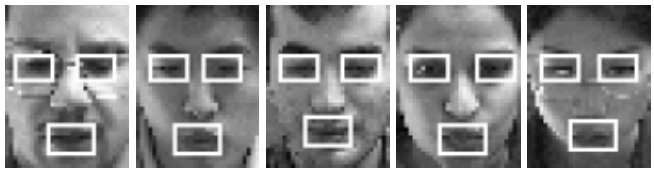

FIG. 4.19 – Bonnes détections.

On voit que les mauvaises détections détectent souvent des objets qui ressemblent aux objets caractéristiques, comme les sourcils ou la partie inférieure du nez.

4.5.3 Localisation avec la troisième méthode pour arbitrer

La troisième méthode pour arbitrer compare les positions des détections pour les trois objets caractéristiques avec un modèle géométrique du visage et choisit les trois positions correspondant le mieux au modèle géométrique du visage comme détections les plus probables. Les résultats illustrés sont obtenus en utilisant les valeurs suivantes pour les variances VAR_HOR, VAR_VER, VAR_EYED et VAR_CENTER :

$$VAR\_HOR = 1$$
$$VAR\_VER = 2{,}25$$
$$VAR\_EYED = 1$$
$$VAR\_CENTER = 9$$

De nouveau les histogrammes de détections et les différentes erreurs de détections e_{ij} pour l'œil droit, l'œil gauche et la bouche sont présentées pour commencer.

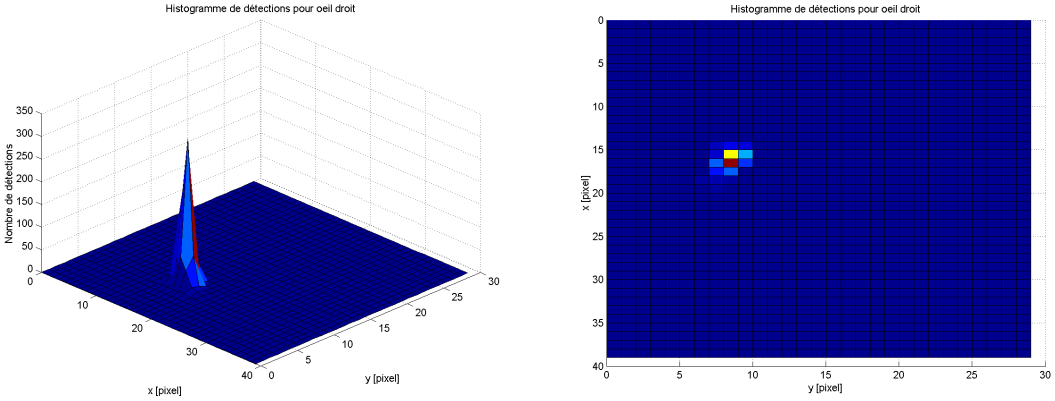

FIG. 4.20 – Histogramme de détections pour l'œil droit. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	94,9
e_{11}	27,2
e_{22}	5
e_{33}	1,3
e_{44}	0,4

TAB. 4.8 – Erreur de détections pour l'œil droit.

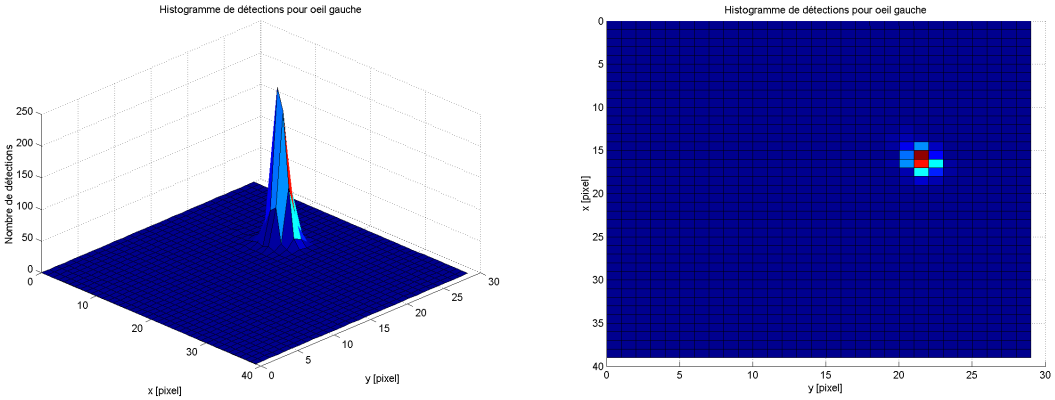

FIG. 4.21 – Histogramme de détections pour l'œil gauche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	93,9
e_{11}	33,7
e_{22}	5,5
e_{33}	1,6
e_{44}	0,5

TAB. 4.9 – Erreur de détections pour l'œil gauche.

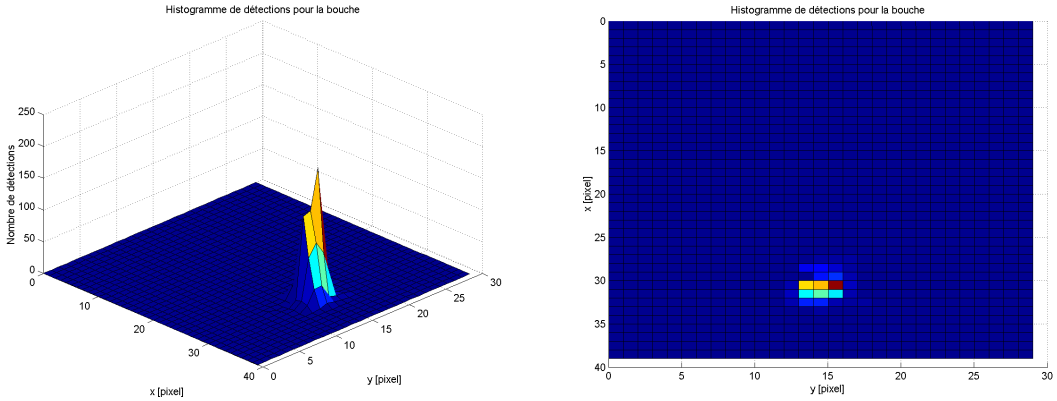

FIG. 4.22 – Histogramme de détections pour la bouche. L'histogramme droit est vu de haut.

Type d'erreur de détections	[%]
e_{00}	97,1
e_{11}	69,2
e_{22}	17,8
e_{33}	10,1
e_{44}	2,8

TAB. 4.10 – Erreur de détections pour la bouche.

Il est intéressant de constater que les erreurs de détections pour les trois objets caractéristiques sont clairement sous ceux obtenues en utilisant une des deux méthodes d'arbitrage précédentes. Ce n'est pas vraiment étonnant, comme la troisième méthode d'arbitrage utilise des connaissances a priori sur la géométrie du visage. À première vue, cette méthode semble être la meilleure. Pourtant, elle à aussi un désavantage assez important. Comme elle modélise des connaissances a priori (comme la distance en pixels entre les yeux), sa performance dépend fortement de la grandeur du visage par rapport à la taille de l'image. Pour un bon fonctionnement de cette méthode d'arbitrage, il est donc important d'avoir un bon programme de détection de visages qui précède ce programme de localisation. Le programme de détection de visages devrait idéalement fournir des images de visages qui ont une relation entre la taille du visage et la taille de l'image constante.

Les figures suivantes illustrent des bonnes et des mauvaises détections faites avec la troisième méthode d'arbitrage.

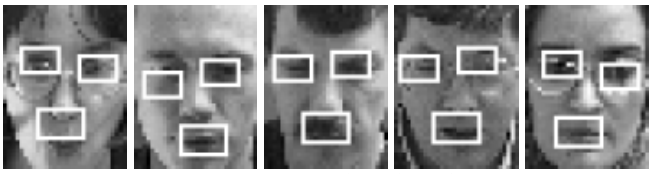

FIG. 4.23 – Mauvaises détections.

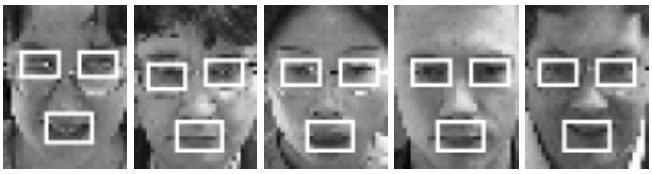

FIG. 4.24 – Bonnes détections.

On voit bien comme les proportions des distances entre les différents objets caractéristiques restent plus ou moins constantes. C'est le cas pour les mauvaises détections comme pour les bonnes détections. Ceci est dû au modèle géométrique du visage qui est utilisé pour la troisième méthode d'arbitrage.

4.5.4 Temps d'exécution du programme

Le programme implémenté a été testé sur un ordinateur équipé d'un processeur Pentium II 300MHz et de 256 MB de RAM. Le tableau suivant montre le temps utilisé du programme implémenté pour traiter une image pour les trois différentes méthodes d'arbitrage.

Méthode d'arbitrage utilisée	[sec/image]
Première méthode	0,58
Deuxième méthode	0,585
Troisième méthode	0,59

TAB. 4.11 – Temps de traitement utilisé par image.

5 Conclusion et travaux futurs

5.1 Conclusion

Dans ce rapport, on a vu qu'il existe beaucoup de méthodes pour faire de la localisation de visages. Principalement, les méthodes peuvent être subdivisées en méthodes basées sur les images et méthodes feature-based (ou géométriques). Selon le contexte, il y a des méthodes qui conviennent plus que d'autres.
La méthode choisie et analysée dans ce rapport est une méthode basée sur les images. Un programme de localisation a été implémenté utilisant un algorithme de Boosting, des masques visuels qui sont une variante d'ondelette de Haar modulé par une Gaussienne et trois méthodes différentes d'arbitrage. Cette méthode possède les avantages suivants :

- Elle permet d'avoir une bonne performance pour des visages vus de face.
- L'utilisation d'ondelettes pour les masques visuels rend la méthode robuste contre la variation de l'illumination lorsque le visage est illuminé de face.
- Des performances acceptables peuvent être atteintes pour une résolution des images assez faible. Ceci permet d'avoir un algorithme assez rapide.
- La méthode peut être adaptée à d'autres applications. Il suffit de changer les données d'apprentissage.

Mais elle a aussi des désavantages :

- Elle n'est pas capable de localiser des visages qui ne sont pas vu de face.
- Il faut avoir une grande base de données avec les positions exactes des objets caractéristiques connues. Autrement, il n'est pas possible d'entraîner un bon classificateur. En plus, il est dur de savoir a priori combien d'images sont nécessaires pour l'entraînement. La base de données devrait consister en images prises dans différentes conditions et situations (illumination de face, du côté droit et gauche, bouche ouverte et fermée, yeux ouverts et fermés, etc.)
- L'utilisation d'une résolution basse ne donne pas une position exacte des objets caractéristiques.
- L'utilisation d'ondelettes pour les masques visuels ne rend la méthode pas robuste contre la variation de l'illumination lorsque le visage est illuminé de côté. Cela a été vu lorsque le classificateur pour l'œil droit a été utilisé pour l'œil gauche en réfléchissant les masques visuels à l'axe vertical. Si l'utilisation d'ondelettes pour les masques visuels rendait la méthode robuste contre la variation de l'illumination lorsque le visage est illuminé de côté, alors on aurait du avoir les mêmes performances de détection pour le classificateur de l'œil droit et de l'œil gauche.

Des tests ont montré que le classificateur pour l'œil droit possède les meilleures performances de détections. La raison pour cela est qu'on avait un ensemble d'images d'entraînement assez bien à disposition pour l'œil droit, ce qui n'était pas le cas pour l'entraînement du classificateur de la bouche. La meilleure performance de détection du classificateur de l'œil droit par rapport à l'œil gauche vient probablement du fait que le classificateur de l'œil droit a été utilisé pour le classificateur de l'œil gauche en réfléchissant les masques visuels à l'axe vertical et que les visages sont souvent illuminés du côté gauche ce qui fait apparaître l'œil droit différemment que l'œil gauche.
Les tests ont aussi montré que la méthode d'arbitrage utilisant des connaissances a priori du visage possède la meilleure performance de localisation. Le désavantage de cette méthode est que la performance de localisation dépend directement de la qualité du programme de

détection de visages précédant le programme de localisation. Cette méthode d'arbitrage peut aussi être vue comme une méthode feature-based.

5.2 Travaux futurs

Le programme implémenté peut être amélioré. Des améliorations possibles sont :

- Entraîner un modèle pour l'œil gauche. Les essais ont montré que le classificateur pour l'œil gauche n'est pas si performant que le classificateur pour l'œil droit. Ce n'est pas très étonnant comme on a utilisé le classificateur de l'œil droit pour l'œil gauche en réfléchissant simplement les masques visuels à l'axe vertical. Si on entraîne un modèle pour l'œil gauche, on devrait avoir de meilleurs résultats de détections.

- Entraîner un meilleur classificateur pour la bouche en utilisant cette fois ci la position exacte de la bouche pour extraire des images d'entraînements.

- Implémenter une fonction qui permet de tourner la fenêtre glissante. Ceci permettra de localier un visage tourné.

- Utiliser d'autres connaissances a priori du visage pour pouvoir faire un arbitrage encore plus performant.

Fabrice Vermont
Lausanne, 4.3.2005

Bibliographie

[1] W. Zhao, R. Chellappa, P. J. Phillips, A. Rosenfeld, *Face Recognition: A Literature Survey*, ACM Computing Surveys, Vol. 35, No. 4, pp. 399-458, December 2003

[2] Erik Hjelmås, Boon Kee Low, *Face Detection: A Survey*, Computer Vision and Image Understanding, Vol. 83, pp. 236-274, 2001

[3] Jerome Friedman, Trevor Hastie, Robert Tibshirani, *Additive Logistic Regression: a Statistical View of Boosting*, Dept. of Statistics, Stanford University, California, August 1998

[4] D. Cristinacce, T. Cootes, *Facial feature detection using AdaBoost with shape constraints*, Dept. Imaging Science and Biomedical Engineering, University of Manchester, Manchester

[5] Andrew Webb, *Statistical Pattern Recognition*, Second Edition, 2002

[6] David C. Lay, *Linear Algebra And Its Applications*, Second Edition, 2000

[7] Ben Kröse, Patrick van der Smagt, *An introduction to Neural Networks*, Eighth Edition, November 1996

[8] Henry A. Rowly, Shumeet Baluja, Takeo Kanade, *Neural Network-Based Face Detection*, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 1996

[9] Henry A. Rowly, Shumeet Baluja, Takeo Kanade, *Neural Network-Based Face Detection*, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, December 1997

[10] Ara V. Nefian, Monson H. Hayes III, *Face Detection And Recognition Using HiddenMarkov Models*, Center of Siganl and Image Processing, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA

[11] Zoubin Ghahramani, Michael Jordan, *Factorial Hidden Markov Models*, Department of Computer Science, University of Toronto, Toronto, Canada and Department of Brain & Cognitive Science, Massachusetts Institute of Technology, Cambridge, USA, 1997

[12] Jean-Philippe Thiran, Support du cours de : *Reconnaissance de formes*, Institut de Traitement des Signaux (ITS), École Polytechnique Fédéreale de Lausanne (EPFL), Switzerland, 2004

[13] Paul Viola, Michael Jones, *Robust Real-time Object Detection*, Second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing, and Sampling, Vancouver, Canada, July 2001

[14] Ron Meir, Gunnar Rätsch, *An Introduction to Boosting and Leveraging*, Department of Electrical Engineering, Technion, Haifa, Israel, The Australian National University, Canberra, Australia

[15] S. Mannor and R. Meir, *On the existence of weak learners and applications to boosting*, Machine Learning, 48(1-3):219-251, 2002

[16] R.E. Schapire, *A brief introduction to boosting*, In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999

[17] E. Bailly-Baillière, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariéthoz, J. Matas, K. Messer, V. Popovici, F. Porée, B. Ruiz, J.-P. Thiran, *The BANCA Database and Evaluation Protocol*, 4th International Conference on Audio- and Video-Based Biometric Person Authentication, AVBPA, 2003

[18] K. Messer, J. Matas, J. Kittler, J. Luettin, G. Maitre, *XM2VTSDB: The Extended M2VTS Database*, Second International Conference on Audio- and Video-Based Biometric Person Authentication, 1999

[19] O. Jesorsky, K. Kirchberg, R.Frischholz, *Robust Face Detection Using the Hausdorff Distance*, Proceedings of Audio- and Video-Based Authentication, p.90-95, 2001

[20] V. Popovici, J.-P. Thiran, Y. Rodriguez, S. Marcel, *On Performance Evaluation of Face Detection and Localization Algorithms*, Swiss Federal Institute of Technology, Signal Processing Institute, CH-1015 Lausanne, Switzerland, Dalle Molle Institute of Perceptual Intelligence, Computer Vision Group, CH-1920 Martigny, Switzerland, 2004

[21] L. G. Farkas, *Anthropometry of the Head and Face*, Raven Press, 1994