TELECOM PARIS

ECOLE DOCTORALE D'INFORMATIQUE TELECOMMUNICATION ET ELECTRONIQUE DE PARIS

THÈSE

présentée pour obtenir le grade de

DOCTEUR DE L'ECOLE NATIONAL SUPÉRIEURE DES TÉLÉCOMMUNICATIONS

Spécialité : Signal et Images

par

Sébastien ONIS

sous la direction du Pr. Jean-Luc DUGELAY
Co-encadré par : Henri Sanson & Christophe Garcia

Titre :

Appariement Robuste de Formes Visuelles Complexes Application à la Détection de Visages

soutenue publiquement le 15 octobre 2009

JURY

Maurice Milgram	UPMC	*Président*
Jean Michel Jolion	Université de Lyon	*Rapporteur*
Frédéric Jury	Université de Caen	*Rapporteur*
Jean-Luc Dugelay	Eurecom	*Directeur de thèse*
Christophe Garcia	Orange Labs	*Examinateur*

Remerciements

Je tiens à remercier en premier lieu Henri Sanson et Christophe Garcia pour m'avoir donné l'opportunité de faire cette thèse au sein l'équipe Tech/Iris à Orange Labs, Rennes. Ils m'ont apporté un soutien précieux ainsi que des conseils avisés.

Je remercie également mon directeur de thèse, Jean-Luc Dugelay pour ses conseils et son accueil lors de mes déplacements à L'Eurecom.

Je remercie les membres du jury pour m'avoir fait l'honneur de s'intéresser à mon travail.

Merci enfin à tous ceux qui m'ont aidés et soutenus pendant ces trois longues années, je pense en particulier à mes collègues et amis à Orange Labs qui ont rendu ces années inoubliables et à ma famille sans qui rien n'aurait été possible.

Table des matières

Table des figures

Liste des tableaux

Acronymes

2D-PCA Two Dimensional Principal Component Analysis / Analyse en Composantes Principales 2D

2D-HMM Two Dimensional Hidden Markov Model / Modèle de Markov Caché 2D

AANN Auto-Associative Neural Networks / Réseaux de Neurones Auto-Associatifs

ANN Artificials Neural Networks / Réseaux de Neurones Artificiels

C-PCA Convolutional Principal Component Analysis / Analyse en Composantes Principales Convolutionnelle

CFF Convolutional Face Finder

CNN Convolutional Neural Networks / Réseaux de Neurones Convolutionnel

EGM Elastic Graph Matching

GMM Gaussian Mixture Model / Mixture de Gaussiènes

HMM Hidden Markov Model / Modèle de Markov Caché

LDA Linear Discriminant Analysis / Analyse Discriminante Linéaire

MLP Multilayer Perceptron / Perceptron Multicouche

P2D-HMM Pseuso Two Dimensional Hidden Markov Model / Pseudo Modèle de Markov Caché 2D

PCA Principal Component Analysis / Analyse en Composantes Principales

SNOW Sparse Network of Winnows

SVM Support Vectors Machines / Séparateurs à Vastes Marges

T-HMM Turbo - Hidden Markov Model / Modèle de Markov Caché - Turbo

CHAPITRE 1

Introduction et objet de l'étude

1.1 Introduction

La détection et la localisation d'objets d'une famille donnée, ou de parties discriminantes, dans les images constitue un outil fondamental pour l'indexation sémantique automatique et la recherche des contenus audiovisuels, ceci pour de multiples applications incluant les services de distribution de contenus, la gestion des contenus personnels, les moteurs de recherche ou encore la vidéo surveillance. Ces vingt dernières années, l'augmentation des moyens informatiques associée à l'avènement de méthodes de classification puissantes tels que l'AdaBoost ou les réseaux de neurones ont permit d'obtenir des systèmes de détection d'objets et en particulier de détection de visages très performants.

Ces systèmes sont capables, à partir d'un grand nombre d'images exemples manuellement annotées représentatives de la classe d'objet à détecter, d'apprendre à distinguer une image de cet objet d'une image n'appartenant pas à cette classe. L'annotation manuelle de ces exemples est un travail long et fastidieux. Nous proposons dans ce document de mettre au point une méthode d'appariement robuste permettant d'obtenir un système de détection capable de fonctionner avec une base d'images exemples de dimension réduite.

Afin d'arriver à ce résultat, nous nous somme inspiré des méthodes de détection d'objets et ou de visages les plus performantes. Nous avons commencé par utiliser une méthode de détection simple mais peu efficace basée sur une mesure de similarité par corrélation. Nous avons ensuite amélioré ce système en y apportant diverses idées inspirées des systèmes de détection de l'état de l'art comme l'utilisation de filtres convolutionnels, ou des traitements d'images permettant de corriger les variations d'illumination. Nous avons ainsi mis au point un système de détection de visages fonctionnel avec très peu d'exemples.

Malgré les résultats encourageants obtenus avec la corrélation, cette mesure de similarité ne permet pas d'atteindre les taux de détection de l'état de l'art. Nous avons alors remplacé la mesure de similarité par corrélation par une mesure de similarité

basée sur des classifieurs bien plus performant tels que les réseaux de neurones. Nous montrons alors que les méthodes mises au point pour la détection par corrélation permettent d'obtenir un système de détection utilisant dix fois moins d'exemples que les systèmes de l'état de l'art mais obtenant des taux de détection comparables.

1.2 Introduction aux systèmes de détection

Cette section décrit les idées communes à l'ensemble des systèmes de détection d'objets dans une image. Bien que de tels systèmes soient basés sur une grande variété de technologies, leur fonctionnement et leur évaluation sont basés sur des méthodes communes que nous nous proposons de décrire ici. Nous commencerons par décrire les difficultés posées par les problèmes de détection et en quoi ce problème se différencie de celui de la reconnaissance. Puis, nous décrirons l'architecture générale d'un système de détection et nous conclurons sur l'évaluation des performances d'un tel système.

1.2.1 Problématique de la détection

Le but d'un système de détection est de décider la présence d'un objet à une position et une échelle donnée dans une image. La première difficulté est que pour détecter un objet, il faut être capable de reconnaître si une image donnée appartient à la classe 'objet' ou 'non objet'. La seconde difficulté est qu'un tel système doit tester la présence d'un objet dans une image à toutes les positions et échelles possibles, ce qui conduit à une complexité de calcul importante. Ainsi, le problème se rapproche de celui de la reconnaissance ou de la classification à deux classes avec la difficulté supplémentaire que la classe 'non objet' est très difficile à représenter puisqu'elle est constituée de l'ensemble des images ne représentant pas l'objet à détecter.

1.2.2 Fonctionnement général des systèmes de détection

Afin de déterminer l'échelle et la position d'un objet (typiquement un visage) dans une image, l'architecture d'un système de détection est divisée en deux parties. Un système de classification capable de déterminer si une image donnée appartient à la classe 'objet' ou 'non objet' et un système permettant d'appliquer le système de classification à toutes les positions et échelles possibles de l'image dans laquelle nous souhaitons détecter un objet. Il existe un grand nombre de systèmes de classification différents que nous décrirons dans la partie état de l'art. Cependant, ces systèmes ont pour point commun de prendre en entrée une image de dimension $h \times l$ fixée et de renvoyer en sortie un score de détection s caractérisant l'appartenance à la classe 'objet' ou 'non objet' (figure : 1.1).

Afin de pouvoir effectuer une détection multi-échelle sur une image test, la méthode généralement utilisée est la suivante : l'image test est successivement sous-échantillonnée d'un facteur de l'ordre de 1.2 conduisant ainsi en l'obtention d'une pyramide d'images dont la plus grande a les dimensions de l'image test et la plus petite celles de la dernière image de la pyramide contenant une image pouvant être traitée par le classifieur. Ainsi, l'objet pourra être détecté quel que soit son échelle

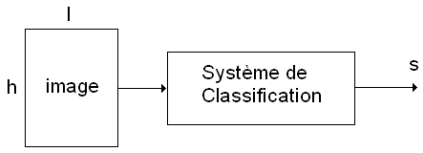


FIGURE 1.1 – Principe d'un système de classification pour la détection : un tel système renvoit pour une image de dimension $h \times l$ un score caractérisant l'appartenance à la classe 'objet' ou 'non objet'. Un seuil θ est ensuite généralement utilisé afin de classer l'image dans la catégorie correspondante.

dans l'image test. La dimension minimum de l'objet détectable correspond à la dimension de l'image traitée par le classifieur, la dimension maximum est celle d'un objet ayant pour hauteur et ou largeur la dimension de l'image test. Afin de déterminer la présence à chaque position et échelle de l'image à tester, le classifieur est utilisé à chaque position possible de l'ensemble des images de la pyramide, *i.e*, chaque pixel des images, résultant ainsi en une pyramide de cartes de scores (figure : 1.2).

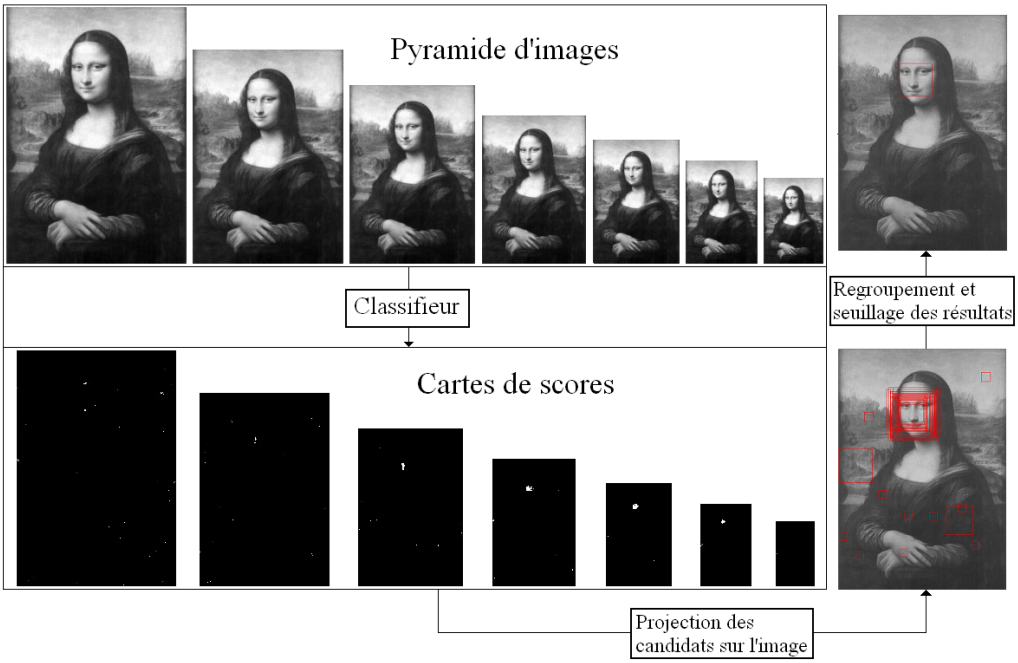


FIGURE 1.2 – Les différentes étapes d'un système de détection : (1) Création d'une pyramide d'images à partir de l'image originale. (2) Extraction des cartes de scores correspondantes à partir d'un classifieur. (3) Projection des candidats détectés sur l'image originale. (4) Regroupement et seuillage des résultats afin d'éviter les recouvrements et de minimiser le nombre de fausses détections.

Cette pyramide de cartes de scores nous donne ainsi une information sur les probabilités que l'image test contienne à une position et une échelle donnée une

image de l'objet recherché. Généralement, plus le score est élevé, plus il est probable que l'image test contienne l'objet à cette position et échelle. Cependant, l'application d'un simple seuillage ne permet généralement pas d'obtenir un positionnement précis de l'objet recherché. En effet, on retrouve souvent autour de cette position plusieurs détections se superposant. Afin d'éviter de retrouver de multiples détections autour de la position réelle de l'objet, on retrouve principalement deux méthodes ou des variantes proches.

Une première stratégie possible consiste à ne garder, pour les différentes détections qui se superposent, uniquement celle ayant le score de détection maximum. La stratégie la plus couramment employée consiste à regrouper les détections proches en position et en échelle afin de les fusionner en une seule détection dont la position et l'échelle est une moyenne éventuellement pondérée de celles du groupe [78]. Le score de détection final est alors égal au nombre de détections fusionnées ou à la somme des scores de ces dernières. Les bonnes détections ayant tendance à regrouper plus de détections proches que les fausses détections, cette méthode apporte généralement une amélioration sensible des résultats en détection de visages [43, 131, 47].

1.2.3 Evaluation d'un système de détection

L'évaluation d'un système de détection est une tâche complexe. Il faut d'abord disposer d'une base de test et d'une base de référence ; la première permettant d'évaluer les performances du système de détection et la seconde d'apprendre les caractéristiques permettant de reconnaître l'objet à détecter. Il convient ensuite de suivre un certain nombre de règles [78] et en particulier, nous devons utiliser une base de référence différente de la base de test utilisée pour évaluer le système ; ceci, afin d'éviter tout problème d'évaluation lié à un sur-apprentissage du classifieur. De plus, la base de test doit être représentative des images sur lesquelles le système sera appliqué. Pour comparer deux systèmes de détection, il est généralement nécessaire de les comparer sur la même base de test, tant il est difficile de trouver deux bases de test représentant la même difficulté. Les visages sont souvent utilisés comme exemple classique d'objet à détecter car on peut trouver des bases de tests standards afin de comparer les systèmes de détection (CMU, Caltech WebFaces) (Annexe : A.1).

Une autre difficulté supplémentaire de l'évaluation est qu'un système de détection peut faire deux types d'erreurs. Il peut ne pas détecter un objet présent (un objet présent dans une image est parfois appelé client), mais il peut aussi, détecter un objet alors que celui-ci n'est pas présent (fausse détection ou imposteur). Ces erreurs sont le plus souvent mesurées par le taux de faux rejet (FRR pour False Rejection Rate) parfois appelé erreur de type I qui est la proportion d'objets non détectés et le taux de fausse acceptation (FAR pour False Acceptance Rate) ou erreur de type II qui correspond à la proportion de fausses détections. Ces deux mesures d'erreurs dont l'importance dépend de l'application que l'on souhaite faire du système de détection sont conflictuelles. En effet, plus le seuil θ, de détection est élevé, plus le risque d'effectuer une fausse détection est faible et plus le risque de ne pas détecter un client est important. Ainsi, le seuil de détection θ permet de minimiser un type d'erreur au dépend de l'autre (figure : 1.3). L'importance de l'erreur de type I par rapport

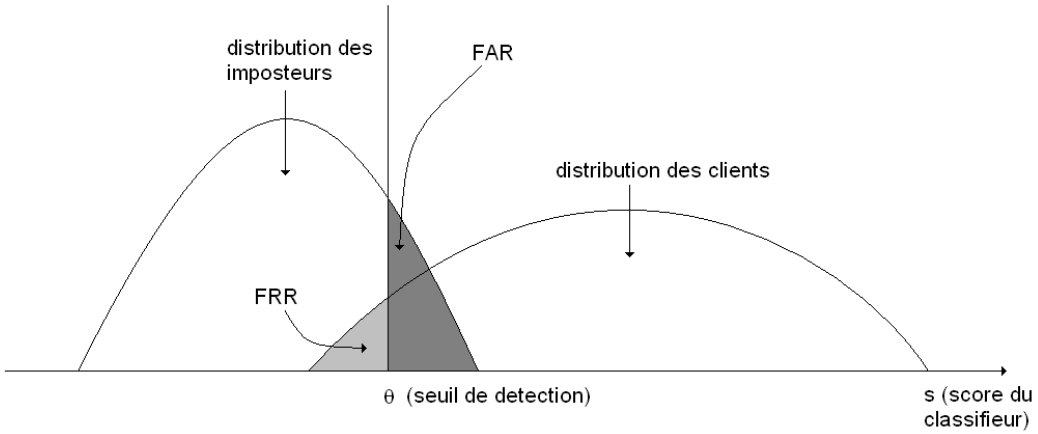
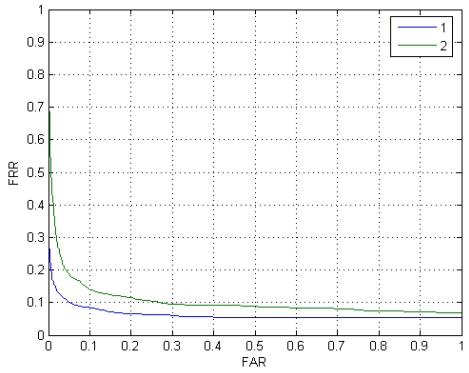


FIGURE 1.3 – Exemple de distribution des clients et imposteurs par rapport au score de détection. Le seuil de détection θ permet de trouver un compromis entre la proportion de fausses détections (FAR) et la proportion de clients non détectés (FRR) ou encore entre la Précision et le Rappel.

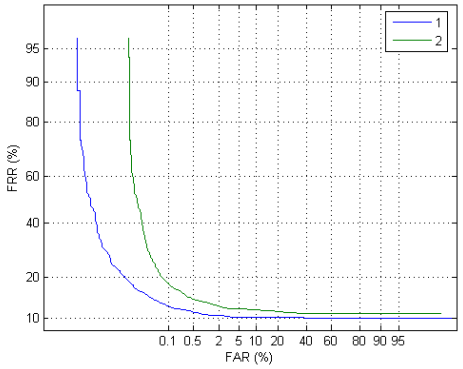
à l'erreur de type II dépend de l'utilisation pratique de notre système de détection. Ainsi, même si deux systèmes sont évalués sur la même base de test, il reste difficile de les comparer.

Afin d'évaluer un système de détection, ses performances peuvent être représentées sous forme de courbes ROC (Receiver Operating Charateristic). Cette courbe est une fonction paramétrique du seuil de détection θ du FAR contre le FRR (figure : 1.4a). Ainsi, plus une courbe s'approche du coin inferieur gauche, meilleur est le système de détection. Une autre méthode de représentation des performances de mesure d'un système de détection est la courbe DET (Detection Error Trade-Off) (figure : 1.4b). La courbe DET est une courbe ROC mais avec une échelle qui suit une loi normale en abscisse et en ordonnée. Ainsi, si la répartition d'imposteurs et de clients suit une distribution Gaussienne, alors la courbe DET est linéaire. Cette représentation est peu utilisée en détection d'objets car la distribution des imposteurs et des clients ne suit généralement pas une loi normale. On la retrouve plus souvent dans des systèmes de traitement du son, comme la reconnaissance de locuteurs où le but est de détecter un locuteur parmi un certain nombre de locuteurs possibles à partir des caractéristiques de sa voix [4, 80]. Enfin, une autre possibilité de représentation des performances est l'utilisation des courbes Rappel/Précision que nous privilégierons dans ce document (figure : 1.4c). La Précision est le nombre de bonne détections divisé par le nombre totale de détection et le Rappel est le nombre de bonnes détections divisé par le nombre total d'objets à détecter. Ainsi, plus une courbe se rapproche du coin supérieur droit, meilleur sera le système de détection. Pour évaluer un système sous forme de chiffres et non de courbes, on donne parfois le taux d'erreur égal (ERR pour Equal Error Rate) qui correspond au point pour lequel le FAR est égal au FRR. En détection de visages, on retrouve souvent pour caractériser le système, la valeur du Rappel pour un nombre fixe d'erreurs avec une

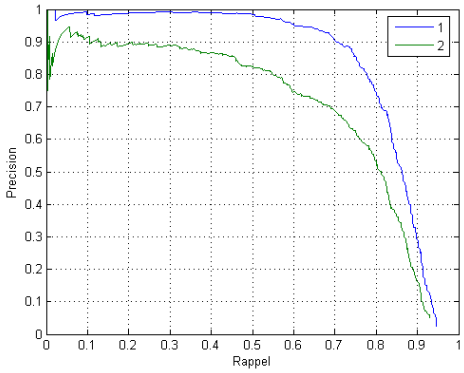
base d'images donnée (le plus souvent CMU).



(a) Courbes ROC



(b) Courbes DET



(c) Courbes Rappel Précision

FIGURE 1.4 – Exemples de courbes mesurant la performance de deux systèmes de détection de visages sur une même base d'images de test. A l'aide de ces représentations, il est aisé de remarquer que le système de détection 1 est meilleur que le système 2.

1.3 Historique de la détection d'objets

L'évolution des méthodes de détection d'objets dans une image est particulièrement liée à la détection de visages. Les visages ont constitué un exemple d'objets particulièrement intéressants pour les systèmes de détection car ils combinent des variations de forme, de luminosité et de couleur. De plus, la détection de visages a de nombreuses applications commerciales puisqu'elle est particulièrement liée à la reconnaissance de visages. En effet, avant de reconnaître une personne à partir de son visage, il est d'abord nécessaire de précisément détecter ce dernier. Enfin, il existe des bases reconnues d'images de visages tels que CMU qui permettent de comparer les résultats des différents systèmes de détection. Ainsi, la majorité des détecteurs de

visages peut être considérée comme des détecteurs d'objets spécialisés ; la détection de visages n'étant qu'un exemple particulier comportant des difficultés communes à un grand nombre d'objets différents. Ceci est particulièrement vrai avec l'avènement des systèmes d'apprentissage. En effet, un système fonctionnel pour la détection de visages, basé sur les méthodes d'apprentissages peut l'être pour d'autres objets similaires, simplement en changeant la base de données d'apprentissage et en adaptant éventuellement la structure et certains paramètres du système. C'est par exemple le cas du détecteur d'objet de Viola-Jones [131] appliqué à la base à la détection de visages mais utilisé couramment pour la détection d'autres objets tels que les mains [60] ou les voitures [22]. Nous pouvons aussi parler des réseaux de neurones convolutionnels [43] appliqués à la détection de visages par Christophe Garcia puis dans d'autres domaines tels que la détection de logos [31], la détection de texte ou d'espacement entre les lettres [112]. Ainsi nous associerons souvent dans ce document la détection d'objets à la détection de visages.

1.3.1 Méthodes basées descripteurs

Les premiers efforts en détection de visages ou d'objets complexes datent des années 1970. Ces systèmes étaient principalement basés sur des méthodes entropomêtriques [113]. Ces techniques ne pouvaient s'appliquer qu'à des visages parfaitement de face sans arrière plan complexe, typiquement, une photo d'identité. Les limitations des moyens informatiques ne permettaient pas l'utilisation des nombreux paramètres nécessaires à la prise en compte de l'ensemble des variations que peut subir une image de visage. Ainsi, le moindre changement dans les conditions de prise de vue des images demandait un complet re-paramétrage du système de détection. A partir des années 1990, l'augmentation des moyens informatiques alliée aux perspectives d'utilisation commerciales, notamment pour la reconnaissance de visages [20] relancèrent les recherches sur la détection. Dans un premier temps, les recherches ont porté sur l'extraction de descripteurs plus robustes permettant de distinguer un visage du reste du monde. Par exemple, la couleur et le grain de la peau ou la géométrie du visage [16, 128]. La détection de visages est alors effectuée en manipulant des distances, des aires, ou des angles liés aux différents descripteurs extraits de l'image. Ainsi, ces méthodes font appel aux connaissances que l'on a sur l'objet à détecter ; on extrait des informations telles que la position de certains contours et d'autres informations géométriques caractéristiques de la classe d'objet à détecter, puis on utilise la connaissance à priori que l'on a de l'objet afin de le différencier du reste du monde. Ces méthodes étant principalement basées sur les descripteurs utilisés, elles sont dites basées sur les descripteurs. Elles sont très dépendantes de l'objet à détecter. Les descripteurs et la manière d'utiliser l'information fournie pour détecter par exemple un visage seront très différents de ceux utilisés pour détecter une voiture. Ils devront être choisis par rapport à nos connaissances à priori sur l'objet à détecter. Ainsi, les méthodes basées descripteurs sont généralement figées sur une classe donnée d'objets et sont difficilement adaptables à une autre classe.

1.3.2 Méthodes basées Images

La disponibilité d'une plus grande puissance de calcul, ainsi que les avancées dans les systèmes d'apprentissage permirent, à partir des années 1990, l'essor de systèmes dit basée images. Dans ces méthodes, l'utilisateur ou concepteur du système n'apporte que peu, voire aucune information sur la classe d'objet à détecter. C'est le système de détection lui même qui à partir d'un certain nombre d'images exemples extrait ces informations afin de classer une image donnée dans la catégorie souhaitée *i.e*, 'objet' ou 'non objet'.

Un des premiers systèmes de détection de visages basé images fonctionnel à été introduit par Turk et Pentland [126] en 1991. L'idée était de déterminer un sous espace de Projection à partir de la méthode de l'Analyse en Composantes Principales (PCA). Sirovich et Kirby [58] avaient montré en 1980 que cette méthode pouvait efficacement représenter des visages en les projetant dans un sous-espace orthogonal calculé à partir d'une base d'images exemples. La méthode de Turk et Pentland consistait donc à projeter une image à tester dans le sous-espace des vecteurs propres. Puis, à comparer l'énergie de l'image projetée avec l'énergie de l'image originale. Si la différence d'énergie est grande, alors il est probable que l'image testée ne soit pas un visage. Au contraire, si la différence d'énergie est réduite, alors il est probable que l'image originale soit un visage. Dans cette méthode, le système de détection détermine lui même les descripteurs caractéristiques de l'objet à détecter. Ainsi, rien n'interdit d'utiliser ce système sur d'autres objets que des visages en changeant simplement la base d'images exemples. La seule condition est que la classe 'objet' doit être convenablement représentée par un nombre réduit de vecteurs propres de la PCA.

L'avènement de systèmes d'apprentissage complexes et en particulier, les réseaux de neurones, a par la suite permis une nette amélioration des taux de détection et de la robustesse des systèmes de détection d'objets. Les premières approches basées sur les réseaux de neurones utilisaient des MLP (Multilayer Perceptron), la forme la plus basique de réseau de neurones [17, 57, 101]. Les résultats furent encourageants sur des bases de test simples. En 1998, Sung et Poggio [125] mirent au point le premier système de détection de visages réellement fonctionnel sur des bases d'images complexes telles que CMU. Il combine des méthodes de clustering (k-Mean) et la PCA afin de minimiser la dimension des données d'entrée. Ces dernières sont ensuite classifiées à l'aide d'un MLP. Ce système montra la supériorité de l'approche basée images sur l'approche basée descripteurs. En effet, non seulement il était, en terme de taux de détection, meilleur que les méthodes basées descripteurs, mais un tel système pouvait être adapté à d'autres objets simplement en changeant la base d'images d'apprentissage. Il faut cependant noter que la méthode est particulièrement adaptée à la détection de visages et que l'efficacité du système est donc très dépendante de la classe d'objet à détecter.

La même année, Rowley *et al* [47] publièrent une méthode utilisant un simple réseau de neurones (MLP) directement appliqué à des images en Niveaux de Gris prétraités de façon à minimiser les variations de luminosité. Ce système obtint des résultats encore supérieurs à ceux de Sung et Poggio et démontra l'efficacité des réseaux de neurones appliqués aux problèmes de détection. Par la suite, Rowley *et*

al ajoutèrent à leur système de détection de visages un réseau de neurones capable de déterminer l'angle de rotation des visages présent dans l'image afin de rendre la détection indépendante à la rotation [48].

En 2000, une approche utilisant un réseau de neurones différent d'un MLP nommée SNOW et ne comportant que deux neurones fut mise au point [109]. Cette méthode montra des résultats au niveau des autres systèmes de détection de visages malgré la relative simplicité du classifieur. En dépit des résultats très prometteurs, on ne retrouve aujourd'hui que peu de développement de cette méthode.

En 2001, Viola et Jones [131] proposèrent un système de détection d'objets basé sur une cascade de classifieurs fonctionnant selon le principe de l'AdaBoost. Très performant en détection de visages et aisément adaptable à une large variété d'objets différents, ce système est en plus très rapide (capable de fonctionner en temps réel sur des vidéos). Il constitue encore aujourd'hui le système le plus couramment utilisé pour effectuer une détection d'objets. On le trouve notamment directement implémenté dans des bibliothèques de traitement d'image telles que OPENCV.

En 2004 Garcia et Delakis [43] utilisèrent un réseau de neurones convolutionnels afin de créer un système de détection de visages considéré aujourd'hui comme le plus performant, aussi bien en terme de taux de détection que de robustesse. De plus, cette méthode est la première à ne pas utiliser de prétraitements d'images complexes, nécessaires à l'ensemble des autres systèmes de détection. Ce système a ensuite été utilisé pour d'autres objets, tel que des logos ou du texte [31] ou encore en reconnaissance de caractères [112] avec des résultats supérieurs à l'état de l'art.

On remarque avec cet historique que la problématique de la détection d'objets s'est déplacée de l'extraction de descripteurs complexes permettant de distinguer une classe d'objet, vers la mise au point de classifieurs utilisant directement l'ensemble des informations des images afin d'en extraire l'information utile. Ainsi, nous sommes progressivement passés de systèmes très spécialisés, extrayant quelques informations précises sur la classe d'objet à détecter, vers des systèmes tels que les réseaux convolutionnels qui utilisent directement l'image en Niveaux de Gris sans aucun prétraitement d'image.

CHAPITRE 2

Etat de l'art des méthodes d'apprentissage et de classification en reconnaissance d'objets

2.1 Introduction

La reconnaissance de formes visuelles, tels que des objets ou des entités assimilables, a fait l'objet de recherches actives ayant conduit à de nombreuses approches. Les systèmes les plus performants sont basés sur des méthodes d'apprentissage et de classification. Ils permettent, à partir de données brutes représentatives de la variabilité de la classe d'objet à reconnaître, d'en extraire les informations caractéristiques. Ce chapitre présente les méthodes de classification et d'apprentissage les plus souvent utilisées en détection et reconnaissance de formes.

Nous diviserons ce chapitre en deux parties constituées d'une part, par les approches génératives consistant à modéliser la classe d'objet à détecter et ne requérant donc que des exemples de cette classe et d'autre part, par les approches discriminatives consistant à modéliser finement la frontière de cette classe par rapport au 'reste du monde', et exigeant à la fois des exemples de la classe à détecter, mais aussi des contre-exemples.

Ces méthodes sont utilisées dans de nombreux domaines pour un grand nombre d'applications, allant de la reconnaissance de la parole, à la compression ou la classification d'images. En ce qui concerne la détection et la reconnaissance d'objets, certaines de ces méthodes sont utilisées pour extraire les descripteurs caractéristiques des données de référence, d'autres utilisent ces descripteurs afin d'effectuer une classification correcte des données d'entrée. L'utilisation d'une technique de classification plutôt qu'une autre est fortement dépendante des contraintes liées à la tâche que nous souhaitons effectuer. Une large part des systèmes de détection ou de reconnaissance actuels utilisent d'ailleurs une combinaison de plusieurs des méthodes présentées ci-dessous.

2.2 Méthodes Génératives

En détection d'objets, les classifieurs doivent distinguer la classe 'objet' de la classe 'non objet'. Si la classe objet peut être convenablement décrite à partir d'un nombre suffisant d'images exemples, ce n'est pas le cas de la classe 'non objet' qui correspond à l'ensemble des images qui ne représentent pas l'objet à détecter. L'intérêt des méthodes génératives est que nous n'avons pas besoin de représenter la classe 'non objet'.

2.2.1 Plus proches voisins

En classification d'images, les images utilisées sont généralement redimensionnées de manière à ce que les systèmes de classification travaillent toujours sur des données de même dimension. Ainsi les images sont représentées sous forme vectorielle $\mathbf{x} = (x_1, \ldots, x_d)^T$ de dimension $d = h \times l$, h et l représentant respectivement la hauteur et la largeur en pixels des images à classer.

La méthode de classification par plus proches voisins est non spécifique au traitement d'image et peut être utilisée pour classer toute donnée représentable sous formes de vecteurs. Elle consiste à classer les données d'entrée en fonction de leur distance à l'exemple le plus proche de la base de référence. Soit $D^N = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ un ensemble de N échantillons exemples labélisés, représentatifs des données à classer. Dans le cadre de la détection d'objets, chaque échantillon exemple $\mathbf{x_i}$ est un vecteur représentant une image de l'objet à détecter. Le principe de la méthode des plus proches voisins est d'assigner à tout échantillon à classer $\mathbf{y}$ la catégorie ω_i de l'échantillon de D^N le plus proche, ou dans le cas de la détection d'objets, à classer l'échantillon $\mathbf{y}$ dans la catégorie 'objet' si la distance à l'échantillon le plus proche est inférieure à un seuil donné et dans la catégorie 'non objet' sinon.

Cette méthode est simple à mettre en œuvre. De plus, lorsque le nombre N d'échantillons exemples tend vers l'infini, l'erreur de classification est toujours inférieure à deux fois le minimum possible défini par les lois de Bayes [30], ce qui assure d'excellents résultats à condition de disposer de suffisamment d'exemples. Enfin cette méthode ne nécessite aucune phase d'apprentissage. Cependant chaque échantillon de test $\mathbf{y}$ présenté doit être comparé à l'ensemble des échantillons exemples, ainsi la complexité de ce classifieur est directement proportionnel à N le nombre d'échantillons d'apprentissage.

2.2.1.1 Mesure de distance

La méthode des plus proches voisins implique nécessairement l'utilisation d'une notion de distance entre deux échantillons $\mathbf{x}$ et $\mathbf{y}$.

D'un point de vue mathématique, une distance D doit respecter les quatre règles suivantes :

1. définie positive : $D(\mathbf{x}, \mathbf{y}) \geq 0$
2. réflectivité : $D(\mathbf{x}, \mathbf{y}) = 0$ si et seulement si $\mathbf{x} = \mathbf{y}$
3. symetrie : $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$

4. inégalité triangulaire : $D(\mathbf{x},\mathbf{y}) + D(\mathbf{y},\mathbf{z}) \geq D(\mathbf{x},\mathbf{z})$

La mesure de distance la plus couramment utilisée est la distance euclidienne :

$$D(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{2.1}$$

Ou encore la distance de Minkowski, généralisation de la distance euclidienne :

$$D(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^k \right)^{\frac{1}{k}} \tag{2.2}$$

Cependant d'un point de vue pratique, la notion de distance utilisée par la méthode des plus proches voisins ne doit pas nécessairement être une distance mathématique. Ainsi nous pouvons utiliser différentes mesures de similarité afin de définir la distance entre deux échantillons. Les deux échantillons les plus proches devenant alors les plus similaires. Des systèmes de détection ou de reconnaissance basés sur la méthode des plus proches voisins peuvent utiliser des mesures de similarité mieux adaptées au traitement d'image telles que la corrélation normée centrée, utilisée dans [75] pour un système de détection de formes et par Wakahara *et al* [132] dans le cadre d'un système de reconnaissance de caractères :

$$S(\mathbf{x},\mathbf{y}) = \frac{1}{\sigma_x \sigma_y} \left(\mathbf{x}^T \mathbf{y} - m_x m_y \right) \tag{2.3}$$

m_x et m_y étant les moyennes respectives des vecteurs $\mathbf{x}$ et $\mathbf{y}$

$$m_x = \sum_{i=1}^{d} x_i \quad m_y = \sum_{i=1}^{d} y_i \tag{2.4}$$

σ_x et σ_y les variances respectives des vecteurs $\mathbf{x}$ et $\mathbf{y}$

$$\sigma_x = \left(\mathbf{x}^T \mathbf{x} - m_x^2 \right)^{\frac{1}{2}} \quad \sigma_y = \left(\mathbf{y}^T \mathbf{y} - m_y^2 \right)^{\frac{1}{2}} \tag{2.5}$$

2.2.2 Analyse en Composantes Principales

En traitement du signal et plus particulièrement en traitement d'image, la dimension des données utilisées pour décrire une information est souvent très grande. L'espace représentant les images de 100×100 pixels est déjà de dimension $d = 10000$. Lorsque nous souhaitons classifier ces données, les temps de calculs ainsi que le nombre d'exemples nécessaires à la représentation de ces données sont directement liés à leur dimension. De plus le volume de données associé à l'ajout de dimensions

supplémentaire dans un espace mathématique augmente exponentiellement avec la dimension. Ce problème nommé par Belman 'malédiction de la dimension' rend complexe l'analyse statistique de problèmes de grande dimension et nécessite l'ajout d'informations a priori dans le but de contraindre le système d'apprentissage.

Une des approches pour réduire la dimension de ces données est de projeter les vecteurs $\mathbf{x} = (x_1, \ldots, x_d)^T$ représentant chaque pixel des images de référence, dans une base de dimension plus réduite représentée par la matrice W permettant toujours de convenablement décrire l'ensemble des échantillons exemples. Ainsi, chaque vecteur $\mathbf{x}$ de dimension d est projeté dans un espace de dimension réduite k par la matrice de projection W^T. Chaque donnée d'entrée est alors représentée par un vecteur $\mathbf{s} = (s_1, \ldots, s_k)^T$ avec $k \leq d$.

$$\mathbf{s} = W^T \mathbf{x} \tag{2.6}$$

Cette approche est appelée méthode de projection linéaire et est très utilisée en traitement d'image pour son efficacité et sa simplicité.

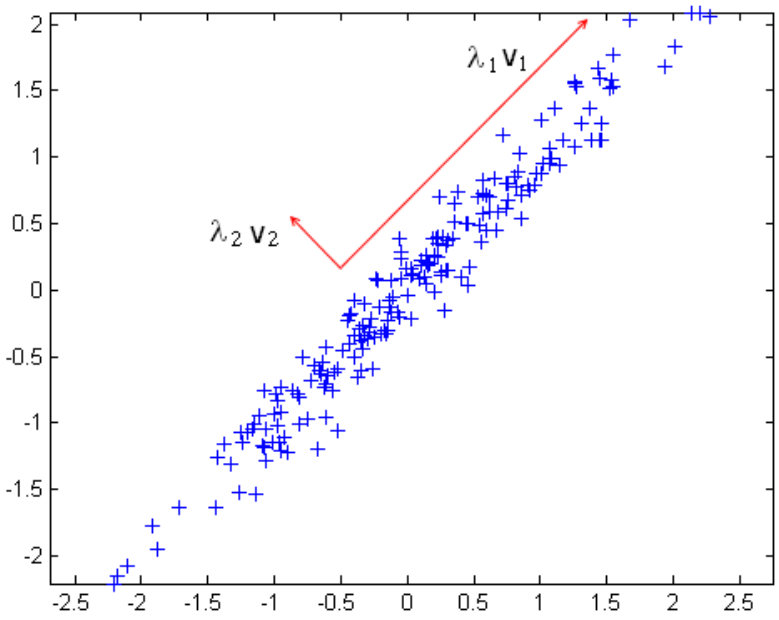


FIGURE 2.1 – Analyse en Composantes Principales d'une distribution gaussienne de points centrée sur 0. Nous remarquons que l'essentiel des informations sur la distribution des points peut être conservé en projetant l'ensemble des points sur le vecteur propre $\mathbf{v_1}$ correspondant à la valeur propre λ_1 la plus grande de la matrice de covariance

Il y a généralement une grande corrélation, non seulement entre les images d'une même classe mais aussi entre les pixels voisins d'une même image. Ainsi en traitement d'image, une des approches les plus classique pour déterminer un espace de

projection représenté par la matrice W est l'Analyse en Composantes Principales (PCA pour Principal Component Analysis). Aussi connue sous le nom de transformée de Karhunen-Loève (KLT) ou de transformée de Hotelling [49], la technique de l'Analyse en Composantes Principales est attribuée à K. Pearson [97]. Cette méthode consiste à décrire un vecteur comme la combinaison linéaire de k vecteurs orthogonaux entre eux (figure : 2.1). Ces vecteurs sont calculés de façon à minimiser la distance $L2$ entre l'ensemble des vecteurs reconstruits et les vecteurs de référence. Autrement dit, la PCA définit la matrice de projection W^T de dimension $k \times d$ telle que $E = \sum_{i=1}^{N} \|\mathbf{x_i} - W\mathbf{s_i}\|$ soit minimum.

Soit, $\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ un ensemble de N vecteurs centrés, *i.e*, $\sum_{i=1}^{N} \mathbf{x_i} = \mathbf{0}$. La matrice de projection W définie par l'Analyse en Composantes Principales est alors formée des k vecteurs propres correspondant aux plus grandes valeurs propres de la matrice de covariance :

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x_i} {\mathbf{x_i}}^T \tag{2.7}$$

La matrice Σ étant symétrique, on peut, par décomposition en valeurs propres, écrire :

$$\Sigma = \begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \cdots & \mathbf{v_d} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & \cdots & \ddots & \cdots \\ 0 & \cdots & 0 & \lambda_d \end{pmatrix} \begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \cdots & \mathbf{v_d} \end{pmatrix}^T \tag{2.8}$$

L'ensemble des vecteurs $\{\mathbf{v_1}, \cdots, \mathbf{v_d}\}$ forme une base orthonormée. *i.e*, si $i \neq j$ ${\mathbf{v_i}}^T \mathbf{v_j} = 0$ et $\forall\, i \in [1, d]\,, \|\mathbf{v_i}\| = 1$.

Si les valeurs propres sont classées par ordre décroissant, *i.e*, $\forall i \in [1, d-1]\,, \lambda_i \geq \lambda_{i+1}$. Alors la matrice de projection W de la PCA s'écrit :

$$W = (\mathbf{v_1}, \cdots, \mathbf{v_k}) \tag{2.9}$$

La valeur de k dépend fortement de l'application et des données sur lesquelles la PCA est utilisée. Plus k est petit, plus l'erreur de reconstruction sera grande. Ainsi, la dimension de l'espace de projection est un compromis entre le besoin de compression des données de départ et la perte d'information due à cette compression.

Cette méthode est particulièrement utilisée en reconnaissance et détection de visages. En effet Kirby et Sirovich ont montré que les images de visages pouvaient être représentées par la combinaison linéaire d'un faible nombre de vecteurs propres [58]. Ainsi dans [126], Turk et Pentland décrivent un système de reconnaissance de visages basé sur une réduction de la dimension des images de visages par l'utilisation de la PCA. Chaque visage à reconnaître est alors projeté dans le sous-espace défini par la matrice W. La classification s'effectue ensuite par la méthode des plus proches

voisins, *i.e*, chaque image de visage présentée au système de classification est associée au visage le plus proche dans le sous-espace des vecteurs propres. Dans [125] la méthode de la PCA est associée à une méthode de clustering (k-Mean) par Sung et Poggio afin de réduire la dimension des données d'entrée d'un Classifieur. Chaque imagette de 19×19 (soit 361 pixels) est projetée dans douze sous-espaces de 75 vecteurs propres. Chaque image est ensuite caractérisée par différentes distances à ces sous-espaces afin d'obtenir un vecteur caractéristique de dimension 24 qui sera classée par un Perceptron Multicouche (MLP Multi Layer Perceptron).

Pour résumer, La PCA est souvent utilisée en traitement d'image afin de représenter les images à classifier avec un minimum de descripteurs. Les images ainsi représentées pourront alors être classées dans la catégorie appropriée plus rapidement et avec moins d'exemples. Cependant la PCA est bien plus efficace pour représenter l'ensemble des variations de textures et de luminosité des objets que pour modéliser les déformations et les variations d'angle de prise de vue des images. La méthode des Modèle d'Apparence Actifs décrite ci-dessous tente de résoudre ce problème.

2.2.3 Modèles d'Apparence Actifs

La technique des Modèles d'Apparences Actifs (AAM pour Active Appearence Models) a été introduite par Cootes, Ewards et Taylor [23] dans le but de représenter avec un minimum de descripteurs des objets déformables tels que des visages. Cette méthode consiste à décrire un objet comme une combinaison linéaire de textures déformées selon une méthode basée sur les modèles actifs de forme [24]. La méthode des Modèles d'Apparence Actifs consiste à déplacer des points d'intérêt selon les principaux modes de déformation associés à l'objet. Ces modes de déformation sont déterminés par la méthode de la PCA appliquée a un grand nombre d'exemples manuellement annotés (figure :2.2).

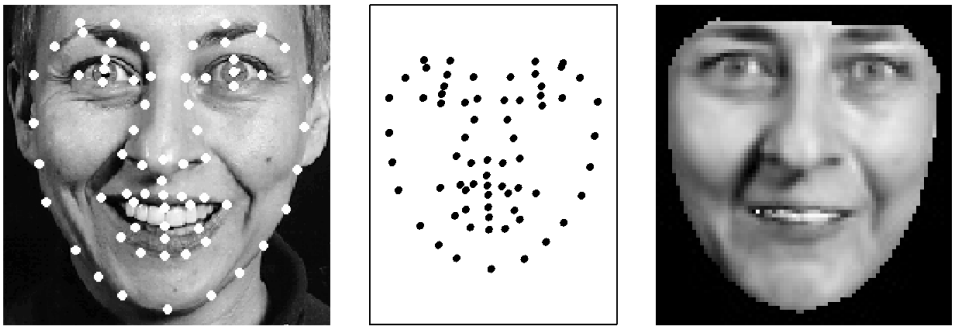


FIGURE 2.2 – Méthode des Modèles d'Apparence Actifs appliqués à un visage dans [23], à gauche, un exemple de visage utilisé avec les points d'intérêt annotés correspondant, au centre, les positions moyennes des points d'intérêt pour l'ensemble des images exemples et à droite, l'image du visage exemple reconstituée à partir de la position des points d'intérêt définis au centre

Les coordonnées $\mathbf{x}$ des points d'intérêt dans une image sont définies par l'équation

suivante :

$$\mathbf{x} = \bar{\mathbf{x}} + P_s \mathbf{b_s} \tag{2.10}$$

$\bar{\mathbf{x}}$ correspond aux coordonnées moyennes des point d'intérêt sur l'ensemble des images de référence, *i.e*, si $\mathbf{x_i}$ représente les coordonnées de l'ensemble des points d'intérêt associés à la i^{ieme} image de référence alors :

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x_i} \tag{2.11}$$

P_s est le sous-espace formé par les k_s premiers vecteurs propres de la matrice de covariance des points d'intérêt. $\mathbf{b_s}$ est donc un vecteur de dimension k_s représentant la position relative des points d'intérêt par rapport à leur position moyenne.

De même, il est possible de représenter avec l'aide de la méthode de la PCA la texture $\mathbf{g}$ qui sera déformée de façon à correspondre au mieux à l'objet à représenter. $\mathbf{g}$ correspond à l'image de l'objet à représenter avec les points d'intérêt dans la position définie par $\bar{\mathbf{x}}$ (image de droite dans la figure : 2.2).

$$\mathbf{g} = \bar{\mathbf{g}} + P_g \mathbf{b_g} \tag{2.12}$$

La forme et la texture associées aux objets à représenter sont généralement corrélées. C'est pourquoi, la méthode des AAM effectue une troisième PCA appliquée à la concaténation des vecteurs b_s et b_g. Ainsi, la méthode des Modèles d'Apparence Actifs décrit à la fois la forme et la texture des objets à représenter par un unique vecteur c.

$$\mathbf{x} = \bar{\mathbf{x}} + Q_s \mathbf{c} \tag{2.13}$$

$$\mathbf{g} = \bar{\mathbf{g}} + Q_g \mathbf{c} \tag{2.14}$$

Connaissant le vecteur $\mathbf{c}$ la méthode des AAM permet de synthétiser l'objet représenté en déformant la texture $\mathbf{g}$ selon la position des points d'intérêt $\mathbf{x}$. Les matrices Q_s et Q_g sont calculées à partir d'une base de référence manuellement annotée et représentent les déformations et variations de textures associées aux objets que nous souhaitons représenter.

2.2.3.1 Décrire une image avec les Modèles d'Apparence Actifs

Les matrices Q_s et Q_g étant à présent connues, le problème est maintenant de déterminer l'image synthétisée qui se rapproche le plus de l'image que nous souhaitons représenter, *i.e*, trouver le vecteur $\mathbf{c}$ qui minimise la distance entre la texture modélisée représentée par le vecteur $\mathbf{g_m}(\mathbf{x}) = \bar{\mathbf{g}} + Q_g \mathbf{c}$ et la texture $\mathbf{g_s}(\mathbf{x})$ correspondant à l'image déformée que nous souhaitons décrire. La méthode utilisée est une optimisation itérative basée sur la descente par gradient.

Soit $\mathbf{r}(\mathbf{c}) = \mathbf{g_s} - \mathbf{g_m}$, alors décrire une image par la méthode des Modèles d'Apparence Actifs revient à minimiser la fonction :

$$E(\mathbf{c}) = \mathbf{r}(\mathbf{c})^T \mathbf{r}(\mathbf{c}) = \|\mathbf{r}(\mathbf{c})\| \tag{2.15}$$

Ainsi nous cherchons $\mathbf{c}$ tel que $\mathbf{r}(\mathbf{c}) = 0$. Ce problème est résolu par la méthode de Gauss Newton. Un développement limité au premier ordre nous donne :

$$\mathbf{r}(\mathbf{c} + \delta\mathbf{c}) = \mathbf{r}(\mathbf{c}) + \frac{\partial \mathbf{r}}{\partial \mathbf{c}} \delta\mathbf{c} \tag{2.16}$$

$$\frac{\partial \mathbf{r}}{\partial \mathbf{c}} = \begin{pmatrix} (\nabla_{\mathbf{c}} r_i)^T \\ \vdots \end{pmatrix} = \begin{pmatrix} \frac{\partial r_i}{\partial c_j} & \cdots \\ \vdots & \ddots \end{pmatrix} \tag{2.17}$$

r_i et c_j représentant respectivement les éléments des vecteurs $\mathbf{r}$ et $\mathbf{c}$.
Cette équation est résolue par l'utilisation de la matrice pseudo inverse R :

$$\delta\mathbf{c} = R\mathbf{r}(\mathbf{c}) \tag{2.18}$$

$$R = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}}^T \frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{c}}^T \tag{2.19}$$

L'approximation linéaire est ensuite itérativement répétée jusqu'à convergence, *i.e*, nous réestimons une nouvelle valeur de $\mathbf{c}$ à partir de la valeur précédente jusqu'a ce que $E(\mathbf{c})$ soit minimum.

$$\mathbf{c_{i+1}} = \mathbf{c_i} + R_i \mathbf{r}(\mathbf{c_i}) \tag{2.20}$$

Cette méthode est particulièrement utilisée pour localiser précisément des visages ou des éléments des visages (yeux, bouche, nez), déterminer l'orientation d'un visage ou estimer la forme d'un visage dans une pose donnée à partir d'images du visage dans une pose différente [25]. En reconnaissance de visages, elle permet d'effectuer la reconnaissance en minimisant les variations dues aux angles de prise de vue et aux déformations inhérentes à un visage [11, 33]. Cette méthode est aussi utilisée pour la compression puisqu'elle permet de représenter une catégorie d'objets avec peu de descripteurs. Les Modèles d'Apparence Actifs sont aussi utilisés en détection [34], un objet étant détecté s'il est suffisamment bien représenté par la méthode des AAM, *i.e*, si la différence entre l'image réelle de cet objet et l'image modélisée est suffisamment petite, le seuil de détection étant fixé de manière pratique. Cette méthode est très efficace pour représenter et décrire les petits objets déformables tels que des visages, car elle modélise aussi bien les textures de l'objet que les déformations possibles associées. Cependant, elle nécessite un procédé d'optimisation qui la rend bien moins rapide qu'une simple PCA. De plus, il est nécessaire d'annoter manuellement de nombreux points d'intérêt sur plusieurs centaines d'images exemples.

2.2.4 Maximum de Vraisemblance

La méthode du Maximum de Vraisemblance consiste à représenter une classe d'exemples donnés comme la densité de probabilité qu'un échantillon représenté par le vecteur $\mathbf{x}$ appartienne à la classe à modéliser. Cette méthode suppose que nous connaissons la forme de la densité de probabilité recherchée à l'exception d'un ensemble de paramètres représenté par le vecteur $\boldsymbol{\theta}$, ainsi la probabilité qu'un vecteur $\mathbf{x}$ appartienne à la classe que nous souhaitons représenter s'écrit :

$$P(\mathbf{x}|\boldsymbol{\theta}) \tag{2.21}$$

La méthode du Maximum de Vraisemblance consiste à déterminer les paramètres $\boldsymbol{\theta}$ qui maximisent la vraisemblance que la densité de probabilité $P(\mathbf{x}|\boldsymbol{\theta})$ génère l'ensemble $D^N = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ d'échantillons exemples labellisés, *i.e* : trouver $\boldsymbol{\theta}$ tel que $P(D|\boldsymbol{\theta})$ soit maximum. Si les échantillons $\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ sont indépendants, nous pouvons écrire :

$$P(D|\boldsymbol{\theta}) = \prod_{i=1}^{N} P(\mathbf{x_i}|\boldsymbol{\theta}) \tag{2.22}$$

Résoudre ce problème analytiquement consiste à trouver les valeurs de $\boldsymbol{\theta}$ pour lesquels le gradient de $P(D|\boldsymbol{\theta})$ est nul. Si on définit le logarithme de la vraisemblance par la fonction $l(\boldsymbol{\theta}) = \ln P(D|\boldsymbol{\theta})$, alors :

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln P(\mathbf{x_i}|\boldsymbol{\theta}) \tag{2.23}$$

La valeurs de $\boldsymbol{\theta}$ maximisant la vraisemblance se déduit alors en résolvant l'équation suivante :

$$\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \ln P(\mathbf{x_i}|\boldsymbol{\theta}) = 0 \tag{2.24}$$

Par exemple, si on définit $P(\mathbf{x}|\boldsymbol{\theta})$ comme une loi normale :

$$P(\mathbf{x}|\boldsymbol{\theta}) = P(x|m, \sigma) = \frac{1}{\sigma(2\pi)^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} \tag{2.25}$$

$\boldsymbol{\theta}$ est un vecteur à deux dimensions contenant la moyenne m et la variance σ de la loi normale. Par la méthode du Maximum de Vraisemblance, nous trouvons des résultats intuitivement satisfaisants. En effet les valeurs estimées m et σ correspondent respectivement à la moyenne et à l'écart type des échantillons exemples D.

Une loi normale est cependant généralement trop simple pour modéliser des classes complexes d'échantillons telles que des images d'objets. La méthode des mélanges de Gaussiennes (GMM Gaussian Mixture Models) a été introduite dans le but de pouvoir représenter des densités de probabilité complexes et ainsi de pouvoir s'adapter à un grand nombre de problèmes de modélisation et de décision.

2.2.4.1 Mélange de Gaussiennes

La méthode des Mélanges de Gaussienne (GMM) consiste à représenter l'ensemble des échantillons exemples comme une densité de probabilité formée par la somme pondérée de k Gaussiennes *i.e* :

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^{k} a_i G(\boldsymbol{\mu_i}, \Sigma_i) \tag{2.26}$$

$$G(\boldsymbol{\mu_i}, \Sigma_i) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_i|^{\frac{1}{2}}} e^{(\mathbf{x}-\boldsymbol{\mu_i})^T \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu_i})} \tag{2.27}$$

$$\sum_{i=1}^{k} a_i = 1 \tag{2.28}$$

Les paramètres $\boldsymbol{\theta} = \{a_1, \boldsymbol{\mu_1}, \Sigma_1, \ldots, a_k, \boldsymbol{\mu_k}, \Sigma_k\}$ sont calculés de façon à maximiser la vraisemblance des échantillons $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ *i.e*, trouver $\boldsymbol{\theta}$ tel que $P(D|\boldsymbol{\theta})$ soit maximum :

$$P(D|\boldsymbol{\theta}) = \prod_{i=1}^{N} P(\mathbf{x_i}|\boldsymbol{\theta}) \tag{2.29}$$

Cette méthode à été utilisée avec succès notamment dans le domaine de la reconnaissance de locuteurs indépendante du texte prononcé [104]. La voix est découpée en segments d'environs 20 ms, et un vecteur $\mathbf{x}$ caractéristique de ce segment en est extrait, généralement en utilisant la méthode de la LPC (Linear Predictive Coding). Un Mélange de Gaussiennes est ensuite utilisé afin de modéliser la probabilité pour une personne donnée d'émettre un son caractérisé par un vecteur $\mathbf{x}$. Ainsi, si nous disposons d'un échantillon de voix d'une personne, il devient possible de déterminer la probabilité que cette voix corresponde à celle modélisée par le Mélange de Gaussiennes.

Dans le domaine du traitement d'image, les Mélanges de Gaussiennes ont de multiples applications, on peut citer notamment la classification d'images [99]. La méthode des GMM est utilisée afin de modéliser la probabilité de présence de certaines textures et couleurs dans différentes catégories d'images. Par exemple, une classe d'images représentant la mer aura de fortes chances de voir apparaître la couleur bleue et certaines textures particulières, ce qui sera modélisé par le Mélange de Gaussiennes. Si nous devons ensuite classer une image présentant une grande proportion de bleu,

elle aura alors une probabilité importante d'être classée dans cette catégorie. Dans [85], Moghaddam et Pentland présentent une méthode basée sur les GMM permettant de déterminer la probabilité $P(\mathbf{x}|\Omega)$ qu'une image $\mathbf{x}$ représente un objet de la classe Ω et l'appliquent à la détection de visages et de mains dans une image.

Une des difficultés posée par la méthode des Mélanges de Gaussiennes est que l'expression (2.29) n'est pas une fonction linéaire des paramètres $\boldsymbol{\theta}$, ainsi une maximisation directe est impossible en pratique. Une des méthodes les plus classique permettant de résoudre ce problème est un algorithme itératif appelé algorithme EM pour Expectation Maximization [29].

2.2.4.2 Algorithme EM

L'idée de base de l'algorithme EM est d'utiliser les paramètres $\boldsymbol{\theta^t}$ à la t^{ieme} itération afin d'estimer de nouvelles valeurs des paramètres $\boldsymbol{\theta^{t+1}}$ telles que $P(D|\boldsymbol{\theta^{t+1}}) \geq P(D|\boldsymbol{\theta^t})$.
Considérons un ensemble d'échantillons $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$. Supposons que des descripteurs soient manquants, un échantillon peut alors s'écrire sous la forme $\mathbf{x_i} = (\mathbf{x_{ig}}, \mathbf{x_{ib}})$ avec $\mathbf{x_{ig}}$ les 'bons' (good) descripteurs et $\mathbf{x_{ib}}$ les 'mauvais' (bad). On notera D_g l'ensemble des 'bons' descripteurs et D_b l'ensemble des 'mauvais'(descripteurs manquants)

Soit $Q(\boldsymbol{\theta}; \boldsymbol{\theta^t})$, l'espérance du logarithme de la vraisemblance de $P(D_g, D_b)$ connaissant une précédente estimation $\boldsymbol{\theta}^t$ des paramètres de la loi de probabilité de $P(\mathbf{x})$ et les 'bons' descripteur D_g. Alors si on note :

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta^t}) = E_{D_b}[\ln P(D_g, D_b|\boldsymbol{\theta})|D_g; \boldsymbol{\theta^t}] \tag{2.30}$$

Algorithm 1 Expectation Maximum

1: initialisation : $\boldsymbol{\theta}^0, T, t = 0$
2: **repeat**
3: $\quad Q(\boldsymbol{\theta}; \boldsymbol{\theta^t})$
4: $\quad \boldsymbol{\theta^{t+1}} = \arg\max Q(\boldsymbol{\theta}; \boldsymbol{\theta^t})$
5: **until** $Q(\boldsymbol{\theta^{t+1}}; \boldsymbol{\theta^t}) - Q(\boldsymbol{\theta^t}; \boldsymbol{\theta^{t-1}}) \leq T$
6: **return** $\boldsymbol{\theta} = \boldsymbol{\theta^{t+1}}$

Cet algorithme garantit que la vraisemblance des échantillons D_g augmente à chaque itération. C'est pourquoi il est très souvent utilisé dans les problèmes de maximisation de vraisemblance.

Dans le cas du problème du GMM, si on note, $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ l'ensemble des échantillons exemples. On définit $z_i \in \{1, \ldots, k\}$ l'information manquante déterminant de quelle Gaussienne du mélange est issue l'échantillon $\mathbf{x_i}$. $\boldsymbol{\theta}$ est l'ensemble des paramètres à estimer du Mélange de Gaussiennes. Alors la fonction $Q(\boldsymbol{\theta}; \boldsymbol{\theta^t})$ est déterminée comme suit :

$$
\begin{aligned}
P(D_g, D_b|\boldsymbol{\theta}) &= P(z_i = j, \mathbf{x_i}|\boldsymbol{\theta}) \qquad (2.31)\\
Q(\boldsymbol{\theta}; \boldsymbol{\theta^t}) &= E_z[\ln P(D_g, D_b; \boldsymbol{\theta})|D_g; \boldsymbol{\theta^t}] \\
&= \sum_{i=1}^{N} E_z\left[\ln P(\mathbf{x_i}, z|\boldsymbol{\theta})|\mathbf{x_i}, \boldsymbol{\theta^t}\right] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{k} P(z_i = j|\mathbf{x_i}, \boldsymbol{\theta^t}) \ln P(\mathbf{x_i}, z_i = j|\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{k} P(z_i = j|\mathbf{x_i}, \boldsymbol{\theta^t}) \ln \left(P(\mathbf{x_i}|z_i = j, \boldsymbol{\theta}) P(z_i = j|\boldsymbol{\theta})\right)
\end{aligned}
$$

Avec :

$$P(\mathbf{x_i}|z_i = j, \boldsymbol{\theta}) = G(\mathbf{x_i}; \boldsymbol{\mu_j}, \Sigma_j) \qquad (2.32)$$

$$P(z_i = j|\boldsymbol{\theta}) = a_j \qquad (2.33)$$

$$P(z_i = j|\mathbf{x_i}, \boldsymbol{\theta^t}) = \frac{a_j^t G(\mathbf{x_i}; \boldsymbol{\mu_j^t}, \Sigma_j^t)}{\sum_{i=1}^{k} a_i^t G(\mathbf{x_i}; \boldsymbol{\mu_j^t}, \Sigma_j^t)} \qquad (2.34)$$

La nouvelle estimation des paramètres $\boldsymbol{\mu_i}$ et Σ_i peut alors être effectuée de manière analytique par le calcul de la dérivée de $Q(\boldsymbol{\theta}; \boldsymbol{\theta^t})$. Et dans le cas des paramètres a_i, par la méthode du Lagrangien afin de respecter la contrainte $\sum_{i=1}^{k} a_i = 1$.

La méthode des Mélanges de Gaussiennes est efficace pour modéliser les distributions statistiques d'échantillons exemples et représenter ainsi, des images, ou un signal quelconque par la probabilité d'apparition d'un certain nombre d'éléments caractéristiques. Cependant un tel modèle ne tient pas compte d'éventuelles relations temporelles ou spatiales entre les différents échantillons. La section suivante présente une approche toujours basée sur la maximisation de la vraisemblance mais permettant de tenir compte des transformations locales d'un signal en modélisant celui-ci comme une séquence d'événements.

2.2.5 Modèles de Markov Cachés

La méthode des Modèles de Markov cachés (HMM pour Hidden Markov Models) a été introduite par Rabinet [102]. Cette méthode de modélisation permet de représenter un signal, ou plus généralement une information, comme une séquence d'états successifs. L'état du système à l'instant t dépendant directement de l'état à l'instant $t-1$. Ainsi, cette méthode est particulièrement efficace pour représenter des informations présentant un aspect temporel. Elle est notamment très utilisée en traitement de la parole ou elle permet de modéliser la succession temporelle des phonèmes qui composent les mots. Bien que les images ne présentent à la base pas d'aspect temporel, celles ci peuvent être représentées comme une succession unidimensionnelle d'observations, modélisables par un Modèle de Markov Caché. Le plus souvent les

images sont divisées en sous-régions, chaque sous-région de l'image correspondant à une observation [115], la méthode des HMM permet alors de modéliser quelles successions de sous-régions sont les plus probables pour la classe d'image que nous avons modélisée. Une autre solution consiste à utiliser des versions modifiées de la méthode des HMM telle que les 2D-HMM ou encore les pseudo 2D-HMM afin de tenir compte de la bidimensionnalité des images.

2.2.5.1 Architecture d'un Modèle de Markov

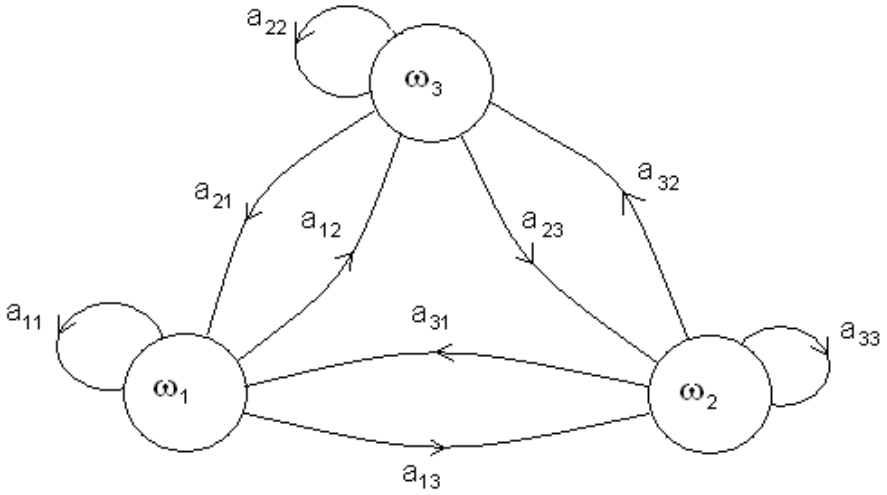


FIGURE 2.3 – Modèle de Markov basique. Les noeuds représentent les états que peut prendre le système à un instant t. Les probabilités de transition a_{ij} sont représentées par les liens entre les noeuds.

Un Modèle de Markov (figure : 2.3) est défini ainsi :
Considérons un ensemble de T états successifs $\omega^T = \{\omega(1), \omega(2), \ldots, \omega(T)\}$, $\omega(t)$ représentant l'état du système à l'instant t. A chaque instant t, un Modèle de Markov ne peut se trouver que dans un état ω_i donné parmi un nombre fini N d'états. Ainsi une séquence particulière de longueur six pour un Modèle de Markov à trois états pourrait être $\omega^6 = \{\omega_1, \omega_3, \omega_3, \omega_2, \omega_1, \omega_2\}$. Dans un Modèle de Markov du premier ordre, la probabilité de se trouver dans l'état ω_j à l'instant t dépend directement de l'état ω_i du système à l'instant $t-1$. Ainsi pour décrire un Modèle de Markov, nous devons disposer des probabilités $P(\omega_j|\omega_i) = a_{ij}$, d'obtenir l'événement ω_j à l'instant t connaissant l'état ω_i à $t-1$. De plus nous devons connaître les probabilités $P(\omega_i(0)) = \pi_i$ correspondant à la probabilité que l'état initial du système soit ω_i. Un Modèle de Markov ainsi défini permet de connaître la probabilité $P(\omega^T)$ de n'importe quelle séquence d'état et ainsi de savoir si la séquence donnée d'événements correspond bien au signal, mot, ou objet que nous avons modélisés.

L'état ω_i d'un système à modéliser n'est pas toujours observable. En effet, il est courant qu'un signal ou une information à modéliser dépende d'un certain nombre d'états non directement observables. Ainsi une voix est un signal sonore observable qui dépend d'un certain nombre d'états cachés (position de la langue, vitesse de vibration des cordes vocales, forme de la cavité nasale...). Le Modèle de Markov Caché est une généralisation du Modèle de Markov qui permet de tenir compte des états cachés du système à modéliser.

2.2.5.2 Architecture d'un Modèle de Markov Caché

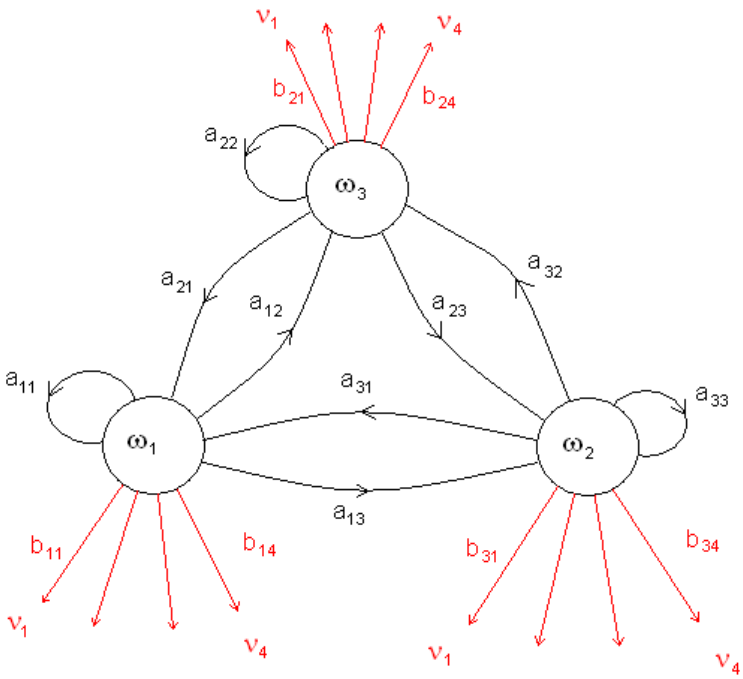


FIGURE 2.4 – Représentation d'un Modèle de Markov Caché ou Automate probabiliste à nombre d'états finis. Les nœuds représentent les états cachés que peut prendre le système à un instant t. Les probabilités de transition a_{ij} sont représentées par les liens entre ces nœuds. Les probabilités b_{jk} des états observables ν_k dépendent directement des états cachés ω_j et sont représentées par les flèches dirigées vers l'extérieur

Comme pour un Modèle de Markov basique, nous continuons de supposer que le système à l'instant t ne peut se trouver que dans un seul état $\omega(t)$, cependant nous ajoutons l'hypothèse que cet état $\omega(t)$ n'est pas observable (caché) et que pour chaque état $\omega(t)$ il y a une probabilité donnée d'émettre un état visible $\nu_k(t)$ que nous pouvons observer. Ainsi la probabilité b_{jk} d'observer l'état ν_k à l'instant t dépend uniquement de l'état caché $\omega(t)$ du système $b_{jk} = P(\nu_k|\omega_j)$. Comme pour le Modèle

de Markov nous définissons une séquence particulière d'états visibles d'un HMM par $V^T = \{\nu(1), \nu(2), \ldots, \nu(t), \ldots, \nu(T)\}$.

Pour résumer un Modèle de Markov Caché est défini par $\lambda = \{W, V, A, B, \Pi\}$

- $W = \{\omega_1, \ldots, \omega_N\}$ L'ensemble des N états cachés possibles
- $V = \{\nu_1, \ldots, \nu_L\}$ L'ensemble des L états visibles possibles
- $A = \{a_{ij}\}_{i,j=1\ldots N}$ Les probabilité de transition d'un état $\omega_i(t-1)$ vers l'état $\omega_j(t)$ ($a_{ij} = P(\omega_j|\omega_i)$ et $\sum_{j=1}^{N} a_{ij} = 1$).
- $B = \{b_{jk}\}_{j=1\ldots N,k=1\ldots L}$ Les probabilités d'observer l'état ν_k si le système est dans l'état caché ω_j ($b_{jk} = P(\nu_k|\omega_j)$ et $\sum_{k=1}^{L} b_{jk} = 1$).
- $\Pi = \{\pi_1, \ldots, \pi_N\}$ Les probabilités que l'état initial du système soit ω_i ($\pi_i = P(w_i(0))$)

La structure d'un Modèle de Markov Caché étant à présent définie, nous allons nous focaliser sur les trois principaux problèmes posés par un tel système :

Le problème d'évaluation Supposons que nous avons un HMM complètement défini dont nous connaissons l'ensemble des paramètres $\lambda = \{W, V, A, B, \Pi\}$. Quelle est la probabilité qu'une séquence d'états visibles V^T soit générée ?

Le problème de décodage Supposons comme pour le problème précédent que nous disposons d'un HMM complet ainsi que d'une séquence d'observation V^T. Quelle est la séquence d'états cachés ω^T la plus probable ayant conduit à ces observations ?

Le problème d'apprentissage Supposons à présent que nous disposions seulement de la structure du HMM c'est à dire son nombre d'états cachés et visibles. Comment déterminer les probabilités a_{ij} et b_{jk} à partir d'un ensemble d'observations V^T ?

2.2.5.3 Evaluation d'un HMM

La probabilité qu'un Modèle de Markov Caché produise une séquence V^T est :

$$P(V^T) = \sum_{r=1}^{r_{max}} P(V^T|\omega_r^T)\, P(\omega_r^T) \tag{2.35}$$

r indexe une séquence particulière ω^T, ainsi, si N est le nombre d'états cachés, le nombre de séquences ω^T possibles est $r_{max} = N^T$. La probabilité d'obtenir une séquence d'états visibles donnée est la somme des probabilités d'obtenir cette séquence pour toutes les combinaisons d'états cachés ω^T possibles pondérés par la probabilité d'obtenir une telle combinaison. Cette probabilité peut être calculée directement à partir des probabilités a_{ij} et b_{jk} et des conditions initiales π_i. La complexité d'un tel calcul étant en $O(N^T T)$, il devient donc très rapidement impossible à effectuer en pratique. Il existe cependant un algorithme de calcul en $O(N^2 T)$:

Posons $\alpha_j(t) = P(V^t, \omega_j(t))$, la probabilité d'avoir t états visibles donnés et l'état caché $\omega_j(t)$ à l'instant t. Nous pouvons alors définir une relation de récurrence pour le calcul de $\alpha_j(t)$:

$$\alpha_j(t) = b_{jk} \sum_{i=1}^{N} \alpha_i(t-1)\, a_{ij} \tag{2.36}$$

$$\tag{2.37}$$

Alors :

$$P(V^t) = \sum_{j=1}^{N} \alpha_j(t) \tag{2.38}$$

Si nous connaissons l'état du système à $t = 0$ on peut alors écrire un algorithme en $O(N^2T)$ permettant de calculer $P(V^T)$:

Algorithm 2 HMM Forward

1: initialisation : a_{ij},b_{jk},V^T,$\alpha_j(0)$
2: **for** $t = 1$ to T **do**
3: $\quad \alpha_j(t) = b_{jk} \sum_{i=1}^{N} \alpha_i(t-1)\, a_{ij}$
4: **end for**
5: **return** $P(V^T) = \sum_{j=1}^{N} \alpha_j(T)$

2.2.5.4 Décodage d'un HMM

Le décodage consiste à trouver ω_r^* tel que $P(\omega_r^*|V^T)$ soit maximum. Comme pour le problème précédent, ceci peut être résolu en énumérant toutes les séquences ω_r possibles, mais la complexité d'une telle méthode est en $O(N^T T)$. D'après, le théorème de Bayes, il est possible de reformuler ce problème ainsi :

$$P(\omega_r|V^T) = \frac{P(V^T|\omega_r)P(\omega_r)}{P(V^T)} \tag{2.39}$$

La probabilité $P(V^T)$ se basant uniquement sur une séquence d'observations fixées, elle ne dépend donc pas de la séquence d'états cachés ω_r. On peut donc formuler le problème de maximisation comme suit :

$$\omega_r^* = \arg\max_{\omega_r} P(V^T|\omega_r)P(\omega_r) \tag{2.40}$$

Par un résonnement récursif similaire à celui pour l'évaluation, l'algorithme de Viterbi [140] permet alors d'effectuer ce calcul avec une complexité en $O(N^2T)$.

2.2.5.5 Méthode d'apprentissage

Le but de l'apprentissage d'un HMM est de déterminer les probabilités a_{ij} et b_{jk} qui permettent de représenter au mieux l'ensemble des exemples d'apprentissage, *i.e*, maximiser la vraisemblance d'obtenir l'ensemble des échantillons exemples. Bien qu'il n'existe pas de méthode permettant d'obtenir les paramètres a_{ij} et b_{jk} conduisant à une représentation optimale de l'ensemble de référence, de nombreuses techniques permettent d'obtenir une bonne solution à ce problème. La technique la plus couramment utilisée est une extension de l'algorithme EM et est connue sous le nom d'algorithme de Baum-Welch [56], ou bien encore d'algorithme 'Forward/Backward'.

Les paramètres $\hat{\mathsf{a}}_{ij}$ et $\hat{\mathsf{b}}_{jk}$ sont estimés itérativement en utilisant leur estimation à l'itération précédente et l'ensemble des observation d'entraînement. Si on pose, $\gamma_{ij}(t)$ la probabilité que le modèle soit dans l'état caché ω_i à l'instant $t-1$ et ω_j à l'instant t connaissant une séquence d'états visibles V^T :

$$\gamma_{ij}(t) = P(\omega_i(t-1), \omega_j(t)|V^T) \tag{2.41}$$

Nous pouvons alors aisément démontrer que :

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(V^T)} \tag{2.42}$$

Avec $\beta_i(t) = P(V^{t+1,T}|\omega_i(t))$ la probabilité d'avoir les états visibles $V^{t+1,T}$ donnés entre les instants $t+1$ et T et l'état caché $\omega_i(t)$ à l'instant t. $\beta_i(t) = P(V^{t+1,T}|\omega_i(t))$ se calcule de manière similaire à $\alpha_j(t)$ par une simple relation de récurrence :

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1)\, a_{ij}\, b_{jk} \tag{2.43}$$

Alors :

$$\hat{\mathsf{a}}_{ij} = \frac{Esperance\ du\ nombre\ de\ transitions\ de\ l'evenement\ \omega_i\ vers\ \omega_j}{Esperance\ du\ nombre\ d'evenements\ \omega_i} \tag{2.44}$$

$$= \hat{\mathsf{a}}_{ij} = \frac{\sum_{t=1}^{T}\gamma_{ij}(t)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\gamma_{ik}(t)} \tag{2.45}$$

$$\hat{\mathsf{b}}_{jk} = \frac{Esperance\ du\ nombre\ d'evenements\ \omega_i\ conduisant\ a\ l'observation\ de\ \nu_k}{Esperance\ du\ nombre\ d'evenements\ \omega_i}$$

$$= \hat{\mathsf{b}}_{jk} = \frac{\sum_{t=1,\nu(t)=\nu_k}^{T}\gamma_{ij}(t)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\gamma_{ik}(t)} \tag{2.46}$$

Ainsi, à chaque itération, les paramètres a_{ij} et b_{jk} vont évoluer de façon à rendre plus probable l'obtention de l'ensemble des observations d'entraînement par le HMM.

2.2.5.6 HMM appliqué à l'analyse d'images

Les Modèles de Markov Cachés sont des systèmes monodimensionnels initialement utilisés pour le traitement du son et plus particulièrement, la reconnaissance de paroles. Nous nous proposons dans cette section de décrire brièvement les approches utilisées pour adapter les HMM aux données bidimensionnelles que sont les images.

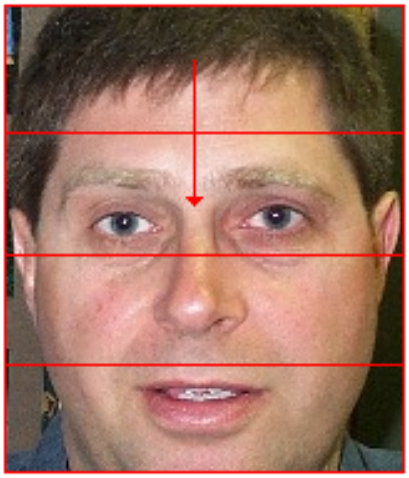


FIGURE 2.5 – Image de visage découpée en bandes horizontales afin de pouvoir être modélisée par un HMM.

Les premières solutions ont consisté à décrire une image comme une succession de sous-régions pouvant éventuellement se recouvrir. Dans [115] des images de visages sont divisées en bandes horizontales du haut vers le bas (figure : 2.5). Les états observables sont les descripteurs de ces sous-régions (front, yeux, nez, bouche). Le HMM décrit ainsi la probabilité d'apparition d'une sous-région (bande verticale) en fonction de la sous-région qui la précède. Ainsi un visage est reconnu si la succession des différentes bandes verticales est suffisamment probable (les yeux devraient être suivis du nez et ainsi de suite).

Un tel modèle ne tient cependant pas compte de l'aspect bidimensionnel des images, et donc de la corrélation entre plusieurs sous-régions voisines. Les Modèles de Markov Cachés bidimensionnels (2D-HMM) permettent d'en tenir compte et de modéliser la probabilité d'apparition d'une sous-région donnée en fonctions des sous-régions voisines. Cependant, la complexité des algorithmes de Baum-Welsh et Viterbi pour un 2D-HMM est exponentielle, ce qui rend cette méthode difficilement utilisable en pratique.

La méthode des Pseudo 2D-HMM (P2D-HMM) a été introduite pour palier à la complexité des 2D-HMM, tout en tenant compte de l'aspect bidimensionnel des images. Elle a été introduite par Agazzi *et al* dans le cadre de la reconnaissance de caractère [1, 62]. Elle utilise la notion de super-états. Ces derniers forment une séquence verticale de bandes d'images. Chaque super-état contient un HMM monodimensionnel permettant de modéliser une bande horizontale de l'image (figure :

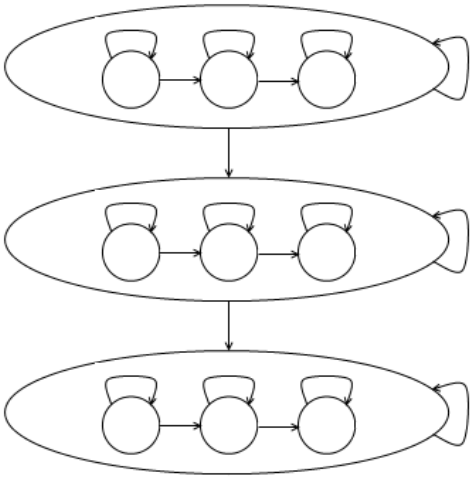


FIGURE 2.6 – Illustration d'un Pseudo 2D-HMM, l'image est modélisée verticalement par des super-états, eux même formés par un HMM classique permettant la modélisation horizontale d'une bande d'image.

2.6).

Dans [100] la méthode des 2D-HMM est approximée par une combinaison de HMM monodimensionnels verticaux et horizontaux. La méthode est appelée Turbo HMM (T-HMM) car elle utilise une technique de décodage directement empruntée aux turbo codes. Dans ces travaux, le T-HMM est utilisé afin de modéliser des visages. Ces visages sont divisés en sous-régions pouvant subir des déformations apprises par le T-HMM selon le principe des EGM (Elastic Graph Matching) [63]. Comme pour les AAM, cette méthode de modélisation tient compte de l'aspect élastique des visages et s'avère très efficace pour les représenter, donnant ainsi des résultats intéressants en reconnaissance de visages.

2.3 Méthodes discriminatives

Les systèmes de détection d'objets les plus performants sont tous basés sur des méthodes de classification discriminatives. Nous nous proposons dans cette section de présenter les méthodes de classification discriminatives les plus courantes ainsi que la façon dont elles sont utilisées en détection d'objets.

Contrairement aux méthodes génératives qui cherchent à modéliser une classe donnée à partir d'un ensemble d'exemples représentatifs, les méthodes discriminatives tentent de modéliser finement les frontières entre différentes classes et exigent à la fois des exemples de la classe à détecter, mais aussi des contre-exemples (classe 'non objet'). L'ensemble de la classe 'non objet' ne pouvant en pratique pas être représentée par un ensemble d'échantillons exemples, seuls les échantillons proches de la frontière à modéliser sont utilisés. Nous commencerons dans cette section par décrire la méthode de 'BootStrapping' utilisée dans la quasi totalité des systèmes

de détection afin de sélectionner les échantillons de la classe 'non objet'. Nous décrirons ensuite une des premières méthodes discriminatives utilisées, *i.e*, L'Analyse Discriminante Linéaire (LDA Linear Discriminant Analysis) qui modélise la frontière entre deux classes par un hyperplan. Finalement nous exposerons les trois méthodes les plus utilisées aujourd'hui, à savoir, AdaBoost, les Séparateurs à Vastes Marges (SVM) et les Réseaux de Neurones (ANN Artificials Neural Networks).

2.3.1 BootStrapping

La méthode de BootStrapping dont le nom désigne aussi une méthode dévaluation des systèmes de classification, a été introduite dans les systèmes de détection par Sung et Poggio dans [125]. Il est en pratique impossible de réunir un ensemble d'échantillons suffisent pour être représentatifs de la classe 'non objet'. En effet chaque imagette de n'importe quelle dimension, et à n'importe quelle position dans une image est un exemple valide pour la classe 'non objet'. La méthode de Sung et Poggio consiste à se focaliser sur les exemples proches de la frontière avec la classe 'objet' en accumulant itérativement les échantillons mal classés par le classifieur discriminatif :

1. Réunir un petit nombre d'échantillons exemples de la classe 'non objet' et un ensemble d'échantillons représentatif de la classe 'objet'
2. Entraîner le classifieur discriminatif à partir de la base d'échantillons exemples courante.
3. Utiliser le système de détection d'objets sur une base d'images ne contenant pas la classe à détecter. Ajouter les fausses détections à la classe 'non objet'.
4. Retourner à l'étape 2 jusqu'à accumulation d'un nombre suffisant d'exemples

A chaque itération, la base d'échantillons de la classe 'non objet' est agrandie par les 'non objet' précédemment mal classés, ce qui permet au classifieurs de corriger les erreurs précédentes. Cette Méthode est utilisée dans la majorité des systèmes de détection, avec parfois, quelques variations. Ainsi dans [43] un BootStrapping est utilisé afin de collecter des images de 'non visage', avec la particularité que le seuil de détection permettant de déterminer si une détection est une fausse alarme diminue avec le nombre d'itérations. Ainsi le système se concentre sur les erreurs les plus significatives lorsque ce dernier commet encore un grand nombre d'erreurs pour ensuite tenir compte des erreurs moins significatives lorsque le système devient plus performant.

2.3.2 Analyse Discriminante Linéaire

L'Analyse Discriminante Linéaire (LDA pour Linear Discriminant Analysis) a été introduite par Fisher [38] en 1936 et a été généralisée plus tard sous le nom d'Analyse Discriminante de Fisher. Le but de la LDA est de déterminer les directions dans lesquelles les données à classer sont le plus aisément séparables. Tout comme la PCA, la LDA est une méthode de projection linéaire dans un sous-espace. Cependant contrairement à la PCA qui détermine le sous-espace le plus adapté à la représentation

des données, la LDA détermine le sous-espace qui permet la meilleure séparation des données à classer (figure : 2.7).

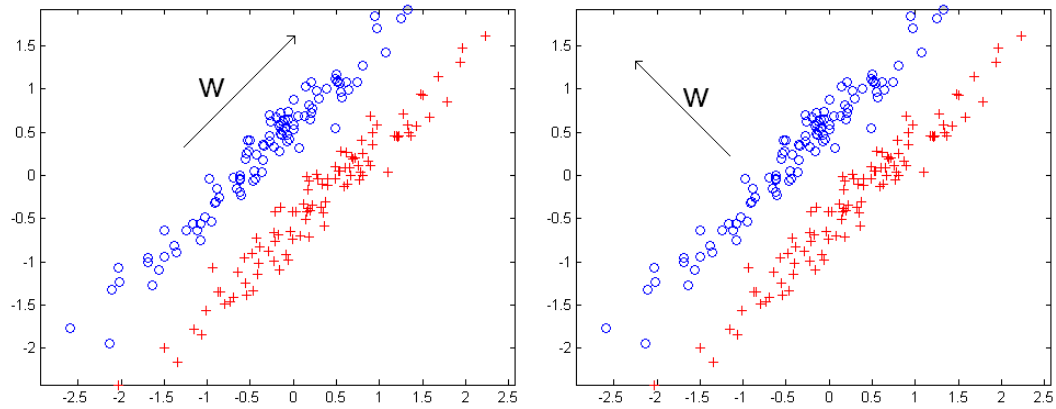


FIGURE 2.7 – Projection du même ensemble de points sur deux lignes différentes dans la direction **w**. La figure sur la gauche montre la direction du premier vecteur propre permettant de représenter l'ensemble des points par la méthode de la PCA. A droite la direction donnée par la LDA permettant la meilleure séparation des échantillons.

Soit un ensemble d'échantillons $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ représentatifs des données à classer dans c classes distinctes $\{D_1, \ldots, D_c\}$. Contrairement à la méthode de la PCA, la dimension du sous-espace de projection est fixée par le nombre de classes à discriminer. Ainsi la dimension de l'espace de projection est égale au nombre de classes à discriminer moins un $(c-1)$. L'espace de projection W est déterminé en maximisant le critère de Fisher [38, 9] :

$$J(W) = \frac{variance\ inter-classe}{variance\ intra-classe} = \frac{\left|W^T \Sigma_B W\right|}{\left|W^T \Sigma_W W\right|} \tag{2.47}$$

Avec Σ_W la somme des matrices de covariance de chaque classe D_i comprenant N_i échantillons :

$$\Sigma_W = \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m_i}) (\mathbf{x} - \mathbf{m_i})^T \tag{2.48}$$

$$\mathbf{m_i} = \frac{1}{N_i} \sum_{\mathbf{x} \in D_i} \mathbf{x} \tag{2.49}$$

Et Σ_B la matrice de covariance des moyennes de chaque classe pondéré par leur nombre d'échantillons :

$$\Sigma_B = \frac{1}{N}\sum_{i=1}^{c} N_i \left(\mathbf{m_i} - \mathbf{m}\right)\left(\mathbf{m_i} - \mathbf{m}\right)^T \tag{2.50}$$

$$\mathbf{m} = \frac{1}{N}\sum_{\mathbf{x}\in D} \mathbf{x} \tag{2.51}$$

Ainsi, le critère de Fisher est maximum lorsque la variance des échantillons de chaque classe est minimum et que la variance des moyennes est maximum, *i.e*, ce critère maximise la distance entre chaque classe tout en minimisant 'l'espace' qu'elles occupent.

La solution à ce problème de maximisation est obtenue en calculant les vecteurs propres $\mathbf{v_i}$ de la matrice $\Sigma_W^{-1}\Sigma_B$. En effet, les colonnes de la matrice W qui maximisent J correspondent aux vecteurs propres $\mathbf{v_i}$.

Le vecteur propre avec la plus grande valeur propre est connu comme le discriminant de Fisher. La projection sur cet axe est donc une mesure permettant de déterminer à quelle classe appartient un échantillon donné. Cette méthode permet donc en fixant un seuil, de déterminer un hyperplan frontière entre les classes que nous souhaitons séparer. Cependant un simple hyperplan est généralement insuffisant pour convenablement séparer les images d'un 'objet' à détecter du 'reste du monde' [123]. En détection, la méthode est plus souvent utilisée afin de déterminer un sous-espace de projection limitant la dimension des données à classer. Dans [139] Yang *et al* utilisent la LDA afin d'effectuer une détection de visages. Les classes 'visage' et 'non visage' sont divisées en c sous classes plus aisément linéairement séparables. La LDA est ensuite utilisée afin de déterminer un espace de projection de dimension $c-1$ qui permet une meilleure séparation des échantillons. En reconnaissance de visages, la projection dans un sous-espace déterminé par la LDA a montré des résultats supérieurs à l'utilisation de la PCA dans la très grande majorité des cas [81].

L'Analyse Discriminante Linéaire a donc pour avantage la simplicité et la linéarité. Cependant, en détection et en reconnaissance d'objets, les classes ne sont généralement pas linéairement séparables et la LDA s'avère insuffisante pour effectuer une classification efficace. Les Séparateurs à Vaste Marge, que nous nous proposons de décrire ci-dessous permettent de modéliser des frontières plus complexes et sont souvent utilisés dans les problèmes de reconnaissance de formes.

2.3.3 Séparateurs à Vastes Marges (SVM)

La méthode des Séparateurs à Vastes Marges [12, 130, 26, 18] est une technique de classification très générale permettant de déterminer une frontière séparant deux classes. Cette frontière est définie par le principe de la minimisation structurelle du risque formulée par Vapnik [129]. Soit $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ un ensemble de N échantillons exemples labellisés par $z_i = \{1, -1\}$ en fonction de la classe correspondante de l'échantillon $\mathbf{x_i}$. Le but de cette méthode est de trouver un hyperplan de séparation qui maximise la marge entre les échantillons les plus proches de chaque classe (figure : 2.8).

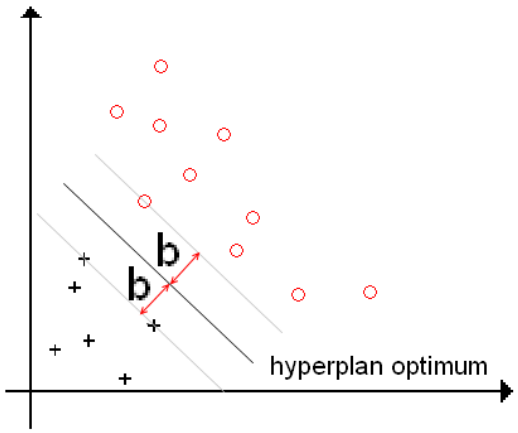


FIGURE 2.8 – Entraîner un SVM consiste à trouver l'hyperplan qui maximise la distance avec les échantillons les plus proches de chaque classe. Les vecteurs supports correspondent aux échantillons à la distance b de l'hyperplan de séparation des deux classes. Dans le cas ci-dessus, il y a trois vecteurs supports

Soit :

$$\mathbf{a} = \begin{bmatrix} \omega_0 \\ \boldsymbol{\omega} \end{bmatrix} \quad \mathbf{y_i} = \begin{bmatrix} 1 \\ \mathbf{x_i} \end{bmatrix} \tag{2.52}$$

Alors la distance d_i entre un échantillon $\mathbf{x_i}$ et l'hyperplan séparateur défini par le vecteur $\mathbf{a}$ s'écrit :

$$d_i = \frac{z_i \mathbf{a}^T \mathbf{y}_i}{\|\boldsymbol{\omega}\|} \tag{2.53}$$

Ainsi, trouver l'hyperplan qui maximise la distance aux échantillons les plus proches de chaque classe, revient à trouver le vecteur $\mathbf{a}$ maximisant b avec $\forall\, i,\ d_i \geq b$. L'hyperplan de séparation étant inchangé par un changement d'échelle de $\boldsymbol{\omega}$, on peut définir que $b\,\|\omega\| = 1$. Le problème revient alors à minimiser $\|\boldsymbol{\omega}\|$ sous la contrainte suivante :

$$z_i \mathbf{a}^T \mathbf{y}_i \geq 1 \tag{2.54}$$

L'utilisation de la méthode du multiplicateur de Lagrange [130] nous permet alors de formuler le problème comme la maximisation du Lagrangien :

$$L(\boldsymbol{\alpha}) = \sum_{I=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j z_i z_j \mathbf{y}_j^T \mathbf{y}_i \tag{2.55}$$

Sous les contraintes :

$$\forall\, i \in [1, N], \alpha_i \geq 0 \tag{2.56}$$

$$\sum_{i=1}^{N} \alpha_i z_i = 0 \tag{2.57}$$

Ce problème de maximisation est en général résolu par des méthodes d'optimisation quadratiques. La direction de l'hyperplan séparateur définie par la méthode des SVM est alors représentée par le vecteur $\boldsymbol{\omega}$:

$$\boldsymbol{\omega} = \sum_{i=1}^{N} \alpha_i z_i \mathbf{x}_i \tag{2.58}$$

Notons que tous les α_i sont nuls exceptés ceux correspondants aux vecteurs supports. Connaissant $\boldsymbol{\omega}$ et les échantillons correspondants aux vecteurs supports, on déduit aisément ω_0 de l'équation (2.53). Finalement les nouveaux échantillons pourront être classifiés en déterminant de quel coté de l'hyperplan séparateur ils se trouvent :

$$z = \mathrm{sgn}(\mathbf{a}^T \mathbf{y}) \tag{2.59}$$

Dans la pratique, il est cependant courant que le problème ne soit pas complètement linéairement séparable. L'équation (2.54) n'a alors plus de solution. Afin de résoudre cette difficulté la contrainte formulée par l'équation (2.54) est remplacée par l'équation suivante :

$$z_i \mathbf{a}^T \mathbf{y}_i \geq 1 - \tau \tag{2.60}$$

$$\tau \geq 0 \tag{2.61}$$

La condition de l'équation 2.57 devient alors :

$$\forall\, i \in [1, N], 0 \leq \alpha_i \leq \tau \tag{2.62}$$

$$\sum_{i=1}^{N} \alpha_i z_i = 0 \tag{2.63}$$

2.3.3.1 SVM non linéaire

En traitement d'image, et plus particulièrement en détection d'objets, les frontières entre les classes sont généralement trop complexes pour pouvoir être modélisées efficacement par un simple hyperplan. Afin de remédier au problème de l'absence de séparateur linéaire, Boser *et al* [12] ont proposé l'utilisation de fonctions noyaux permettant de reconsidérer le problème dans un espace de dimension supérieure. Dans

ce nouvel espace, il est alors probable qu'il existe un séparateur linéaire. L'idée des fonctions noyaux a été introduite par Mercer en 1909 [83] alors que leur utilité dans le contexte des systèmes d'apprentissage a été montrée en 1964 par Aizermann, Bravermann et Rozoener [3].

Les fonctions noyaux permettent de transformer un produit scalaire dans un espace de grande dimension, ce qui est coûteux, en une simple évaluation ponctuelle d'une fonction. Ainsi les équations (2.52) deviennent :

$$\mathbf{a} = \begin{bmatrix} \omega_0 \\ \boldsymbol{\omega} \end{bmatrix} \quad \mathbf{y_i} = \begin{bmatrix} 1 \\ \mathbf{g}(\mathbf{x_i}) \end{bmatrix} \tag{2.64}$$

Le problème n'est alors plus linéaire par rapport à $\mathbf{x}$ mais par rapport à $\mathbf{g}(\mathbf{x})$ sa projection dans un espace de dimension supérieure. Le problème est alors résolu en utilisant la même procédure que pour le cas linéaire, les échantillons $\mathbf{x}$ étant remplacés par leur projection $\mathbf{g}(\mathbf{x})$. Le problème d'optimisation revient alors à maximiser :

$$L(\boldsymbol{\alpha}) = \sum_{I=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j z_i z_j \mathbf{g}(\mathbf{x}_j)^T \mathbf{g}(\mathbf{x}_i) \tag{2.65}$$

Sous les contraintes :

$$\forall\, i \;\in\; [1, N]\,, \alpha_i \geq 0 \tag{2.66}$$

$$\sum_{i=1}^{N} \alpha_i z_i = 0 \tag{2.67}$$

Le problème d'une telle méthode est qu'elle implique un produit scalaire entre vecteurs dans l'espace de redescription, de dimension élevée, ce qui peut s'avérer très coûteux en termes de calculs. Pour résoudre ce problème, on utilise une astuce connue sous le nom de 'Kernel Trick', qui consiste à utiliser une fonction noyau $\mathbf{g}(.)$, qui vérifie :

$$K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{g}(\mathbf{x_i})^T \mathbf{g}(\mathbf{x_j}) \tag{2.68}$$

Ainsi, il n'est plus nécessaire pour résoudre le problème, ni de calculer un produit scalaire dans un espace de grande dimension, ni de connaître explicitement la fonction $\mathbf{g}(.)$, puisque seul nous est nécessaire le résultat du produit scalaire $K(\mathbf{x_i}, \mathbf{x_j})$.

En pratique, même si nous n'avons pas besoin de connaître la fonction $\mathbf{g}(.)$, nous devons nous assurer que pour la fonction $K(\mathbf{x_i}, \mathbf{x_j})$ choisie, il existe toujours une fonction $\mathbf{g}(.)$ vérifiant l'équation (2.68). Ceci est assuré par la conditions de Mercers [130]. Les noyaux usuels respectant ces conditions employés avec les SVM sont :

- Noyaux polinomiaux :

$$K(\mathbf{x_i}, \mathbf{x_j}) = \left(\mathbf{x_i}^T \mathbf{x_j} + 1\right)^n$$

- Noyaux Gaussiens :

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2}$$

- Noyaux Tangeante Hyperbolique :

$$K(\mathbf{x_i}, \mathbf{x_j}) = \tanh\left(\kappa(\mathbf{x_i}^T \mathbf{x_j}) - \tau\right)$$

2.3.3.2 SVM avec peu d'exemples d'apprentissage

De par la minimisation du risque structurel, les SVM permettent une bonne généralisation. Afin de gérer le faible nombre d'exemples, les principales techniques consistent à minimiser la dimension des vecteurs à classer [15]. En particulier, on peut citer la méthode RFE (Recursive Feature Elimination) [46] qui consiste à itérativement éliminer lors de l'apprentissage les descripteurs les plus proches de zéro ou encore la méthode MRS (Minimum Reference Set) consistant à garder le nombre minimum de descripteurs permettant de correctement classifier l'ensemble des exemples [21]. Il est aussi possible soit d'utiliser un SVM linéaire pour éviter le sur-apprentissage, soit d'adapter la fonction kernel au problème que l'on souhaite traiter.

Depuis leur développement dans les années 1990, les SVM ont été très largement utilisés dans de très nombreux domaines, y compris en traitement d'image pour des système de détection [116, 92]. Cependant, les résultats les plus remarquables en détection sont généralement basés sur les méthodes de classification que nous allons présenter dans les deux sections suivantes, *i.e*, AdaBoost [131] et les Réseaux de Neurones [43].

2.3.4 Boosting

L'idée du Boosting est de combiner plusieurs classifieurs dit 'faibles' (peu efficaces) afin d'obtenir un classifieur 'fort', *i.e*, performant. En pratique, un classifieur 'faible' doit simplement classer mieux que le hasard. Une méthode simple pour effectuer un Boosting est la suivante (figure : 2.9) :

Considérons que nous disposons de trois classifieurs 'faibles' pour un problème de classification dans deux catégories. Soit $D = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ l'ensemble des échantillons exemples. Un premier classifieur C_1 est entraîné avec une sélection D_1 de $N_1 = N/3$ échantillons de D. Un second classifieur C_2 est entrainé avec une sélection D_2 de $N_2 = N/3$ échantillons nous apportant le maximum d'information par rapport aux échantillons D_1. Concrètement la moitié seulement des échantillons de D_2 doivent être bien classés par C_1. Finalement le troisième classifieur C_3 est entrainé à partir d'une sélection d'échantillons correspondant aux échantillons de D pour lesquels les classifieurs C_1 et C_2 ne donnent pas la même classification. Ainsi D_3 représente l'ensemble des échantillons mal représentés par la combinaison des classifieurs C_1 et C_2.

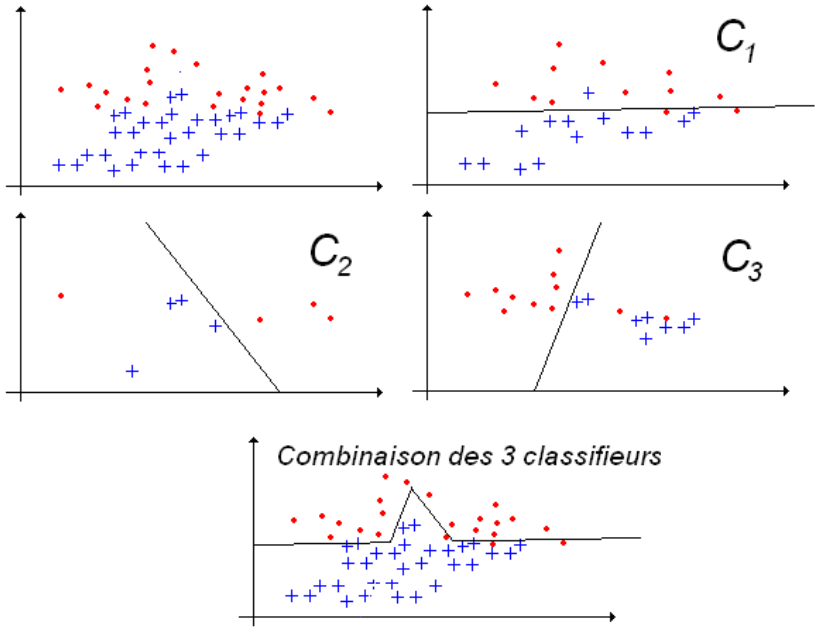


FIGURE 2.9 – Exemple de méthode de Boosting utilisant trois classifieurs linéaires dans un espace de dimension deux. Nous obtenons par la combinaison des trois classifieurs, une frontière et un classement bien plus précis qu'avec un seul classifieur linéaire.

Il existe de nombreuses variations sur les méthodes de Boosting, un des algorithmes les plus utilisés s'appelle AdaBoost et permet de combiner les classifieurs faibles jusqu'à obtenir l'erreur de classification désirée pour les échantillons d'entraînement.

2.3.4.1 AdaBoost

AdaBoost (contraction de Adaptative Boosting) est une méthode de Boosting introduite par Freund et Schapire [39]. Les performances et la simplicité de cette technique font qu'elle est utilisée dans un grand nombre de problèmes de classification et notamment pour le détecteur d'objets de Viola-Jones [131]. Comme pour la méthode de boosting basique, l'idée de cette méthode est de focaliser les classifieurs 'faibles' sur les exemples les plus difficiles à classer.

Bien qu'il existe des extensions de la méthode AdaBoost pour le cas des problèmes multi-classes [40, 117], nous nous focaliserons ici sur le problème de classification entre deux classes. Dans cette configuration, le classifieur 'fort' est constitué de k_{max} classifieurs 'faibles' binaires successifs $C_k = \{-1, +1\}$. Chaque échantillon exemple $\mathbf{x_i}$ associé a la classe $z_i = \{-1, +1\}$ reçoit pour chaque classifieur 'faible' C_k un poids w_i^k qui détermine sa probabilité d'être sélectionné pour entraîner le classifieur. Chaque échantillon mal classé par le classifieur C_k verra son poids associé augmenter pour le

classifieur suivant C_{k+1} et diminuer dans le cas ou il serait bien classé.

$$w_i^{k+1} = \frac{w_i^k}{W_{k+1}} \times e^{-z_i C_k(\mathbf{x_i})\alpha_k} \tag{2.69}$$

$$W_{k+1} = \sum_{i=1}^{N} w_i^{k+1} \tag{2.70}$$

$$\alpha_k = \frac{1}{2}\ln\left(\frac{1-E_k}{E_k}\right) \tag{2.71}$$

$$E_k = \sum_{z_i \neq C_k(\mathbf{x_i})} w_i^k \tag{2.72}$$

Ainsi, les exemples souvent mal classés voient leur poids augmenter par rapport à ceux souvent bien classés. De plus, chaque classifieur faible se voit associé un poids α_k lié a son erreur de classification. Le score final $s(\mathbf{x})$ donné par l'association de l'ensemble des classifieurs se calcule ainsi :

$$s(\mathbf{x}) = \sum_{k=1}^{k_{max}} \alpha_k C_k(\mathbf{x}) \tag{2.73}$$

Le score $s(\mathbf{x})$ associé à l'exemple $\mathbf{x}$ est la somme pondérée du score des classifieurs $C_k(\mathbf{x})$. Le poids associé à chaque classifieur dépend directement du taux d'erreur E_k du classifieur. Si le classifieur est parfait $E_k = 0$ et α_k tend vers l'infini. Si le classifieur a un taux d'erreur $E_k = 0.5$ (taux de détection du hasard) alors, $\alpha_k = 0$, son poids sera nul dans l'association de classifieurs. Ainsi la méthode d'AdaBoost maximise l'importance des 'meilleurs' classifieurs 'faibles'.

Algorithm 3 AdaBoost

1: p, q nombre d'échantillons associés à la classe $z_i = \{-1, +1\}$
2: $\forall i = 1 \dots N$, $w_i^1 = \frac{1}{p}$ si $z_i = -1$, $w_i^1 = \frac{1}{q}$ sinon
3: **for** $k = 1$ to k_{max} **do**
4: Entraîner le classifieur 'faible' C_k avec les poids w_k^i et les échantillons exemples $\mathbf{x_i}$
5: $E_k = \sum_{z_i \neq C_k(\mathbf{x_i})} w_i^k$
6: $\alpha_k = \frac{1}{2}\ln\left(\frac{1-E_k}{E_k}\right)$
7: $w_i^{k+1} = \frac{w_i^k}{W_{k+1}} \times e^{-z_i C_k(\mathbf{x_i})\alpha_k}$
8: **end for**
9: **return** $s(\mathbf{x}) = \text{sgn}\left(\sum_{k=1}^{k_{max}} \alpha_k C_k(\mathbf{x})\right)$

Cette méthode de Boosting est particulièrement intéressante car nous pouvons choisir le nombre k_{max} de classifieurs de manière à atteindre le taux d'erreur souhaité sur les échantillons exemples. De plus, il est démontré [40] que le taux d'erreur

E associé au classifieur 'fort' décroît exponentiellement avec le nombre de classifieurs 'faibles' utilisés. En effet, si on écrit l'erreur de classification du k^{ieme} classifieur 'faible' sous la forme $E_k = \frac{1}{2} - \gamma_k$, la variable γ_k supérieure à 0 représentant l'amélioration du taux d'erreur du classifieur par rapport au hasard. Alors on peut écrire :

$$E = \prod_{k=1}^{k_{max}} \left(1 - 4\gamma_k^2\right)^{\frac{1}{2}} \leq e^{\left(-2\sum_{k=1}^{k_{max}} \gamma_k^2\right)} \tag{2.74}$$

Une valeur trop grande de k_{max}, outre des problèmes de complexité de calcul peut, en théorie conduire à un sur-apprentissage, conduisant à une bonne classification sur les images exemples mais une très mauvaise généralisation du problème dans le cas pratique. Les nombreuses expérimentations sur AdaBoost ont cependant montré que le sur-apprentissage est en pratique très rare avec cette méthode. Les propriétés de généralisation d'AdaBoost en font donc une méthode de classification très efficace en pratique. Elle est très utilisée pour des problèmes de détection 'd'objets réels', tels que des piétons [70] ou même des formes plus théoriques tels que des croisements de lignes dans une image [77]. Enfin le détecteur d'objets de Viola-Jones [131] est l'un des plus utilisé aujourd'hui et est directement basé sur la méthode de classification AdaBoost.

2.3.4.2 Détecteur d'objet de Viola-Jones

Le détecteur d'objet de Viola-Jones constitue aujourd'hui une référence pour les systèmes de détection d'objets. Il permet d'effectuer une détection robuste et en temps réel des objets dans une image. Pour cela il s'appuie sur trois idées fondamentales.

Les Descripteurs : Ils sont basés sur les fonctions de Haar utilisées par Papageorgiou *et al* [95]. Concrètement, ces descripteurs se divisent en trois types différents consistant en sommer les valeurs des pixels dans les rectangles blancs et soustraire ceux dans les rectangles grisés (figure : 2.10). Les dimensions et la position de ces descripteurs pouvant varier dans l'image de l'objet que l'on souhaite détecter, nous arrivons pour une image d'objet de dimension 24×24 pixels à un nombre de descripteurs possibles de 45396.

AdaBoost avec sélection des descripteurs : Le nombre de descripteurs possible pour une image de 24×24 pixels est très grand. Cependant les expérimentations montrent qu'un petit nombre de ces descripteurs est suffisant pour entraîner un classifieur efficace. Afin de sélectionner les meilleurs descripteurs permettant de décrire un objet donné, le système de Viola-Jones utilise un algorithme basé sur la méthode AdaBoost. Les classifieurs faibles C_k sont de simples hyperplans séparateurs utilisant un seul descripteur f_j, un seuil θ_j et une parité $p_j = \{1, -1\}$ permettant de déterminer la direction de la normale à l'hyperplan séparateur constituant la frontière :

$$C_k(\mathbf{x}) = \text{sgn}(p_j f_j(\mathbf{x}) - \theta_j) \tag{2.75}$$

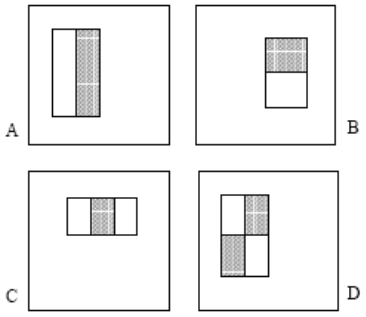


FIGURE 2.10 – Exemples de descripteurs utilisés par le détecteur d'objets de Viola-Jones. La somme des pixels dans les zones grisées est soustraite de la somme des pixels dans les zones blanches. Ces descripteurs se divisent en trois catégories, ceux formés de deux régions (A et B), ceux formés de trois régions comme par exemple C et enfin ceux formés de quatre régions comme D

Dans cette variante de la méthode AdaBoost, chaque classifieur C_k formant le classifieur 'fort' est associé au descripteur f_j permettant de minimiser le taux d'erreur sur l'ensemble des échantillons exemples $\mathbf{x}_i$ pondérés par les poids w_i^k (Algorithme : 4). Ainsi chaque classifieur C_k utilise le descripteur f_j (figure : 2.11) permettant de différencier au mieux les échantillons mal classés par les classifieurs précédents $(C_1 \dots C_{k-1})$.

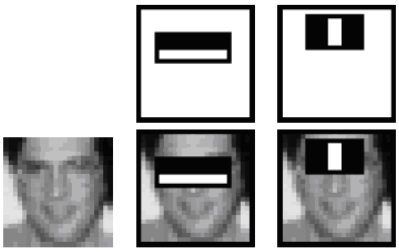


FIGURE 2.11 – Deux premiers/meilleurs descripteurs utilisés pour la détection de visages par l'algorithme de Viola-Jones. Les yeux semblent être la partie du visage permettant la meilleure discrimination par rapport au 'reste du monde'.

Cascade de classifieurs : La méthode expliquée ci-dessus permet d'obtenir un classifieur efficace, cependant, le nombre de classifieurs faibles et donc de descripteurs nécessaires pour obtenir un taux de détection acceptable peut être important (plusieurs centaines pour un détecteur de visages), ce qui ne permet pas une détection en temps réel. Afin de rendre l'algorithme de détection plus rapide, le système de Viola-Jones utilise une cascade de classifieurs (figure : 2.12). Chaque classifieur élimine un certain nombre de 'non objet' tout en sélectionnant pratiquement 100% des objets à détecter. Les premiers classifieurs de la cascade sont basés sur un pe-

Algorithm 4 AdaBoost Viola-Jones

1: p, q nombre d'échantillons associés à la classe $z_i = \{-1, +1\}$
2: $\forall i = 1 \dots N$, $w_i^1 = \frac{1}{p}$ si $z_i = -1$, $w_i^1 = \frac{1}{q}$ sinon
3: **for** $k = 1$ to k_{max} **do**
4: Entraîner l'ensemble des classifieurs h_j^k associé aux descripteurs f_j avec les poids w_k^i et les échantillons exemples $\mathbf{x_i}$
5: Choisir le classifieur $h_j^k = C_k$ avec le plus faible taux d'erreur E_k
6: $E_k = \sum_{z_i \neq C_k(\mathbf{x_i})} w_i^k$
7: $\alpha_k = \frac{1}{2} \ln \left(\frac{1-E_k}{E_k} \right)$
8: $w_i^{k+1} = \frac{w_i^k}{W_{k+1}} \times e^{-z_i C_k(\mathbf{x_i}) \alpha_k}$
9: **end for**
10: **return** $s(\mathbf{x}) = \text{sgn} \left(\sum_{k=1}^{k_{max}} \alpha_k C_k(\mathbf{x}) \right)$

tit nombre de descripteurs et sont donc très rapides mais peu discriminatifs, puis chaque classifieur successif de la cascade devient de plus en plus complexe, permettant une réduction importante des fausses détections. Ainsi le dernier classifieur qui utilise plusieurs centaines de descripteurs, n'est utilisé que sur un nombre très réduit d'imagettes à tester, limitant au minimum le nombre d'opérations requises pour la détection.

Afin de collecter des échantillons de 'non objet' pertinents, cette méthode utilise une variante de la méthode de BootStrapping adaptée à la cascade de classifieurs. Le premier classifieur est entraîné avec p_1 'non objet' choisies aléatoirement (p_1=10000 pour le détecteur de visage). Le second classifieur est entraîné avec p_2 images de fausses détections acquise par la méthode de BootStrapping appliqué au premier classifieur. Une nouvelle itération permet d'entraîner le classifieur suivant à partir de nouvelles fausses détections effectuées par la cascade formée des deux premiers classifieurs. Ainsi chaque classifieur de la cascade se spécialise sur les erreurs faites par les classifieurs précédents.

Les trois idées exposées ci-dessus permettent d'obtenir un système de détection rapide et robuste et font de ce système de détection un des plus utilisés actuellement.

2.3.4.3 Méthodes dérivées du détecteur d'objets de Viola-Jones

La méthode de détection de Viola-Jones a permis, grâce à l'utilisation de descripteurs basés sur les fonctions de Haar, de la méthode de classification AdaBoost et d'une cascade de classifieurs, d'obtenir un système de détection d'objets à la fois robuste, rapide et performant. Ces dernières années, de nombreuses méthodes de détections se sont inspirées du détecteur de Viola-Jones et les dernières évolutions aussi bien, en terme de taux de détection que de complexité de calcul sont presque toutes dérivées de cette méthode. Les améliorations apportées à l'algorithme de Viola-Jones portent aussi bien sur la méthode AdaBoost [124, 118, 50, 69], les descripteurs utili-

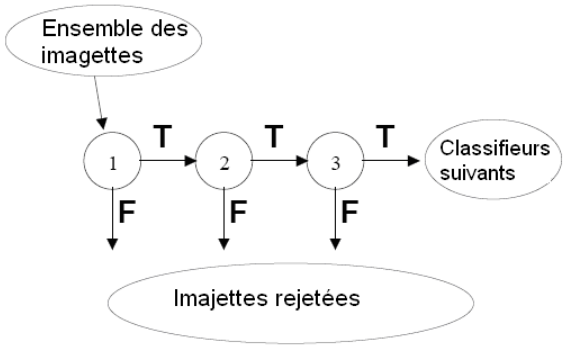


FIGURE 2.12 – Schéma d'une cascade de classifieurs. Chaque imagette présentée au système est classée successivement par chacun des classifieurs. Si un classifieur classe l'imagette dans la catégorie 'non objet' les calculs s'arrêtent et l'imagette est classée dans cette catégorie. Sinon elle est classée dans la catégorie 'objet'. Un système de détection d'objets traitant beaucoup moins d'échantillons 'objet' que 'non objet', ce système permet de grandement diminuer les temps de calculs.

sées [71, 135, 119] et la forme de la cascade de classifieurs [14, 50, 136, 134].

AdaBoost : Parmi les améliorations apportées à la méthode AdaBoost, on notera en particulier la méthode du FloatBoost [69] qui consiste à ajouter une phase de vérification de l'utilité de l'ensemble des classifieurs faibles et si possible de supprimer les classifieurs faibles inutiles. Cela permet de diminuer le nombre de classifieurs faibles utilisés et donc d'augmenter la vitesse de calcul, sans pour autant diminuer le taux de détection. Une autre amélioration nommée RealAdaBoost [118] utilisé par Huang *et al* dans [50] consiste à utiliser des classifieurs faibles donnant non plus une réponse binaire $\{0, 1\}$ mais une réponse appartenant à $[0, 1]$ caractérisant ainsi la confiance que l'on peut avoir dans le résultat du classifieur faible. Cette méthode permet de minimiser le nombre de classifieurs faibles comparé à la méthode de Viola-Jones.

Les descripteurs : Les fonctions de Haar utilisées par Viola-Jones permettent de décrire efficacement les objets à détecter de manière extrêmement rapide grâce à l'utilisation de la méthode des images intégrales. Les descripteurs utilisés restent généralement toujours basés sur les fonctions de Haar. La principale évolution est basée sur l'utilisation de la méthode des 'features centrics evaluation' par Shneiderman [119] et par Yan *et al* [135]. Cette méthode consiste à utiliser chaque descripteur, non pas à une position de l'imagette à classifier mais à toutes les positions possibles. Ainsi, un seul descripteur apporte plus d'informations que la méthode classique 'window centric evaluation' ce qui permet une nette réduction du nombre de classifieurs faibles nécessaires ainsi qu'une amélioration des taux de détection.

Cascade de classifieurs : Les évolutions les plus notables sont dues aux progrès fait sur la cascade de classifieurs. Dans la méthode de Viola-Jones, les classifieurs forts de chaque étage de la cascade de classifieurs sont indépendants les uns des autres. Ainsi, dans la méthode de Viola-Jones, chaque étage de la cascade de classifieurs n'utilise pas l'information apporté par les étages précédents. La méthode des 'Boosting Chain Learning' utilisée par Xiao *et al* dans [134], par Bourdev et Brandt dans [14] et par Huang *et al* dans [50] permet d'utiliser le score des classifieurs forts des étages précédents de la cascade pour entrainer le classifieur fort suivant.

Ces méthodes dérivées de Viola-Jones forment aujourd'hui les systèmes de détection parmi les plus rapides et performants. Une alternative à AdaBoost est basée sur les réseaux de neurones qui constituent aujourd'hui grâce à la variété des techniques possibles, une des méthodes de classification les plus performantes, y compris pour les problèmes de détection. La section suivante présentera les principes des réseaux de neurones, et les différents types de réseaux de neurones utilisés pour les problèmes de détection.

2.3.5 Réseaux de Neurones

Les réseaux de neurones constituent aujourd'hui une des techniques de classification les plus populaires et les plus performantes, y compris pour les problèmes de reconnaissance de formes visuelles et de détection. Cette méthode, inspirée par le fonctionnement du cerveau humain consiste à interconnecter par des synapses, des neurones symbolisés par des fonctions mathématiques simples φ, ceci dans le but d'effectuer des tâches de classification complexes. A chaque connexion à un neurone est associée un poids w_i symbolisant l'importance de l'information x_i associée à cette connexion. La réponse y du neurone est alors le résultat de la fonction φ de la somme des signaux d'entrée $\mathbf{x} = (x_1 \dots x_d)^T$ pondérés par les poids $\mathbf{w} = (w_1 \dots w_d)^T$ et un biais additionel b (figure : 2.13). Ainsi, $y = \varphi(\mathbf{w} \cdot \mathbf{x} + b)$.

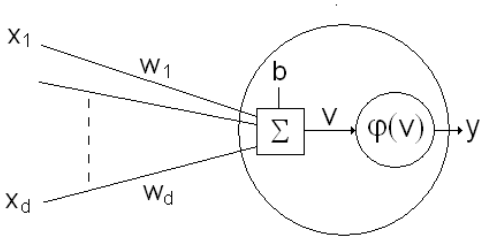


FIGURE 2.13 – Schéma d'un neurone. Les entrées $\mathbf{x}$ sont pondérées par les poids $\mathbf{w}$ et sommées au biais b donnant le résultat v. la réponse du neurone $\mathbf{y}$ est alors donnée par la fonction $\varphi(v)$.

Entraîner un réseau de neurones consiste à trouver l'ensemble des poids $\mathbf{w}$ et le biais b de chaque neurone qui minimise l'erreur E de sortie du réseau de neurone. Dans la très grande majorité des cas, E représente l'erreur quadratique, *i.e*, si $\mathbf{z_i} = \mathbf{g}(\mathbf{x_i})$ est la sortie du réseau de neurones pour le i^{ieme} échantillon exemple et $\mathbf{t_i}$ les valeurs objectifs souhaitées pour l'échantillon, alors :

$$E = \sum_i (z_i - t_i)^2 \tag{2.76}$$

Il existe un très grand nombre d'architectures possibles pour un réseau de neurones. Cette dernière est caractérisée par les fonctions φ associées à chaque neurone, le nombre de neurones du réseau et les connexions entre ces neurones. Les architectures les plus utilisées en reconnaissance de formes et détection d'objets sont présentées dans les sections suivantes.

2.3.6 Perceptron

Inspiré par les travaux sur la théorie cognitive de Friedrich Hayek et Donald Hebb, le Perceptron a été introduit en 1958 par Rosenblatt [106]. Il constitue le modèle de réseau de neurones le plus simple puisqu'il n'est constitué que d'un seul neurone.

Ainsi la sortie du système est la réponse du neurone induite par la somme pondérée par $\mathbf{w}$ des éléments de l'entrée $\mathbf{x}$:

$$net = \mathbf{w} \cdot \mathbf{x} + b \tag{2.77}$$

$$y = \varphi(net) \tag{2.78}$$

La fonction d'activation φ utilisée est une simple fonction heaviside $H(net) = 0$ si $net < 0$ sinon, $H(net) = 1$. Ainsi le perceptron est un simple hyperplan séparateur, l'échantillon $\mathbf{x}$ est classé dans la catégorie '0' ou '1' en fonction de sa position par rapport à l'hyperplan définie par $\mathbf{w}$ et b. L'utilisation d'un algorithme d'optimisation basé sur la minimisation du nombre d'échantillons exemples mal classées [107] permet de traiter tout problème de classification à deux classes. Un simple séparateur linéaire est cependant généralement insuffisant pour traiter des problèmes de classification complexes tels que la détection d'objets dans une image. De plus, Minsky et Papert [84] ont montré que le perceptron est inefficace pour des problèmes simples tels que celui du OU-exclusif où il est impossible de séparer par une droite les points $(0,0)$ et $(1,1)$ des points $(0,1)$ et $(1,0)$ (figure : 2.14). Le Perceptron Multicouche utilise plusieurs neurones interconnectés afin de pouvoir modéliser des frontières plus complexes permettant de traiter des problèmes de classification plus difficiles ainsi que les problèmes de plus de deux classes.

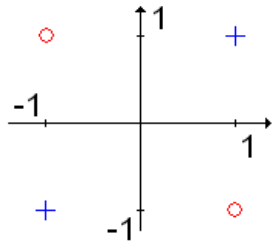


FIGURE 2.14 – Problème de classification OU-Exclusif. Malgré la simplicité du problème qui ne comporte que 2×2 échantillons bidimensionnels à classer, ce problème n'est pas linéairement séparable.

2.3.7 Perceptron Multicouche

Le Perceptron Multicouche (MLP pour Multilayer Perceptron) est la forme de réseau de neurones la plus couramment utilisée. Un tel réseau est constitué d'un minimum de trois couches de neurones. Les neurones d'une couche donnée ne sont connectés qu'aux neurones de la couche suivante (figure : 2.15), ainsi l'activation des neurones est propagée à travers les différentes couches, de l'entrée vers la sortie. Disposant d'une méthode d'apprentissage simple et efficace [107], Le MLP est en plus capable d'approximer n'importe quelle fonction décision et donc n'importe quelle forme de frontière, à condition de disposer de suffisamment de neurones cachés

(neurones non situés sur la première ou la dernière couche du réseau). En détection d'objets, le MLP dispose généralement d'un seul neurone de sortie dont la valeur de sortie $z = g(\mathbf{x})$ permet de définir si un échantillon $\mathbf{x}$ appartient ou pas à la catégorie 'objet'. Un tel système est entraîné de façon à ce que $z = 1$ si l'échantillon d'entrée appartient à la catégorie 'objet' et $z = -1$ dans le cas contraire, la frontière séparant les deux catégorie correspond aux valeurs de $\mathbf{x}$ telles que $z = g(\mathbf{x}) = 0$ (figure : 2.22). Enfin, la souplesse du MLP permet aussi de traiter le cas multi-catégories en faisant correspondre une catégorie à chaque neurone de sortie.

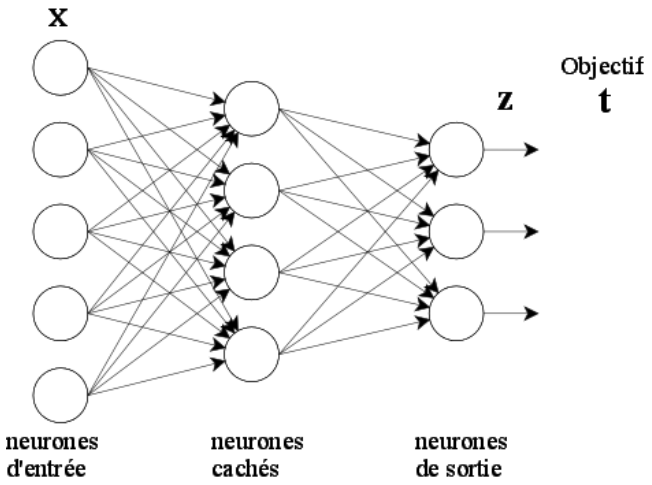


FIGURE 2.15 – Perceptron Multicouche. Il est constitué d'au minimum trois couches de neurones. Une couche d'entrée contenant le vecteur $\mathbf{x}$ à classer, une ou plusieurs couches cachées, et une couche de sortie dont le vecteur résultat $\mathbf{z} = (z_1, \ldots, z_c)$ permet de classer l'échantillon correspondant au vecteur $\mathbf{x}$. Dans cet exemple le MLP est dit entièrement connecté car chaque neurone est relié à l'ensemble des neurones de la couche précédente.

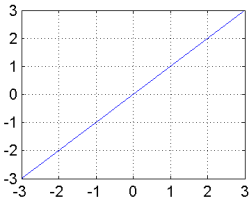
En ce qui concerne les fonctions d'activation φ des neurones du MLP, la seule contrainte théorique est induite par l'algorithme d'apprentissage de Rétropropagation (ou Backpropagation) utilisé. En effet ce dernier est basé sur la méthode de descente par gradient et impose donc que les fonctions d'activation $\varphi(\mathbf{x}, \mathbf{w}, b)$ soient continues et dérivables par rapport à $\mathbf{w}$ et b. En pratique, on retrouve le plus souvent les trois fonctions suivantes (figure : 2.16) :

$$\varphi(net) = net \qquad \text{linéaire} \tag{2.79}$$

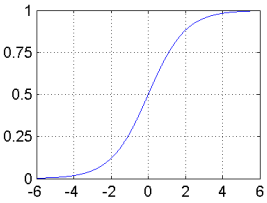
$$\varphi(net) = \frac{1}{1 + e^{-net}} \qquad \text{sigmoid} \tag{2.80}$$

$$\varphi(net) = tanh(net) = \frac{1 - e^{-net}}{1 + e^{-net}} \qquad \text{tangeante hyperbolique} \tag{2.81}$$

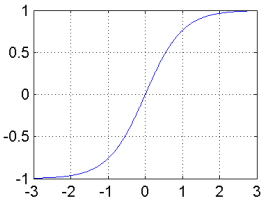
$$net = \mathbf{w} \cdot \mathbf{x} + b \tag{2.82}$$



(a) Linéaire



(b) Sigmoid



(c) Tangeante Hyperbolique

FIGURE 2.16 – fonctions d'activation utilisées dans les réseaux de neurones

2.3.7.1 Perceptron Multicouche appliqué à la détection

Le MLP est très largement utilisé en traitement d'image, pour des tâches aussi variées que l'interpolation [51], la reconnaissance de visages [91, 35, 36], la reconnaissance de mots [19] ou encore la classification d'images [82]. En ce qui concerne le problème de la détection, ont peut citer la méthode mise au point par Sung et Poggio [125] appliquée à la détection de visages. Le système commence par déterminer à l'aide d'un algorithme de clustering dérivé de la méthode du k-Mean, douze clusters caractérisés par six images de visages et six de 'non visage' caractéristiques de la base d'exemples. Pour chaque imagette, ce système extrait deux distances à chacun des douze clusters constituant ainsi pour chaque entrée un vecteur de dimension 24 (figure : 2.17). La première distance D_1 est la distance de Mahalanobis dans un sous-espace de dimension 75 déterminé par la PCA (Σ_i étant la matrice de covariance des exemples associés au i^{ieme} cluster) :

$$D_1(\mathbf{x}, \boldsymbol{\mu_i}) = \frac{1}{2}\left(d\ln(2\pi) + \ln|\Sigma_i| + (\mathbf{x} - \boldsymbol{\mu_i})^T\Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu_i})\right) \tag{2.83}$$

La seconde distance D_2 est la distance euclidienne entre une imagette représentée par le vecteur $\mathbf{x}$ et sa projection dans le sous-espace d'un cluster.

Les vecteurs de dimension 24 ainsi obtenus sont ensuite classés à l'aide d'un perceptron à trois couches entièrement connecté possédant 24 neurones 'cachés' et un neurone de sortie. La fonction d'activation choisie est la fonction Sigmoid. Le MLP est entraîné de façon à renvoyer 1 si l'échantillon d'entrée est un visage et 0 sinon.

Une autre méthode utilisant avec succès un Perceptron Multicouche en détection est celle de Rowley *et al* [47]. Cette méthode a la particularité d'utiliser un Perceptron Multicouche avec comme entrée un vecteur $\mathbf{x}$ correspondant à une image de 20×20 pixels en Niveaux de Gris, prétraitée de façon à corriger les problèmes de luminosité. Un Perceptron Multicouche entièrement connecté sur une image en Niveaux de Gris engendre un très grand nombre de connexions et donc de coefficients à entraîner. C'est pourquoi dans cette méthode le MLP utilisé possède trois couches mais n'est pas entièrement connecté. Nous retrouvons 4 neurones cachés reliés à 4 sous-régions de 10×10 pixels, 16 reliés à 16 sous-régions de 5×5 pixels et enfin 6 neurones reliés à 6 sous-régions de 20×5 pixels (figure : 2.18).

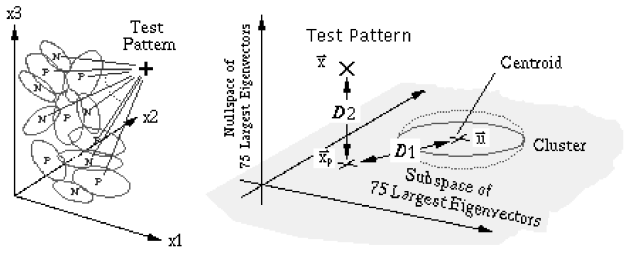


FIGURE 2.17 – Distances utilisées pour le détecteur de visages de Sung et Poggio, à gauche, la distance de Mahalanobis entre un vecteur $\mathbf{x}$ et le centroïde de chacun des 12 clusters, à droite, la distance entre le vecteur $\mathbf{x}$ et le sous-espace de dimension 75 d'un cluster.

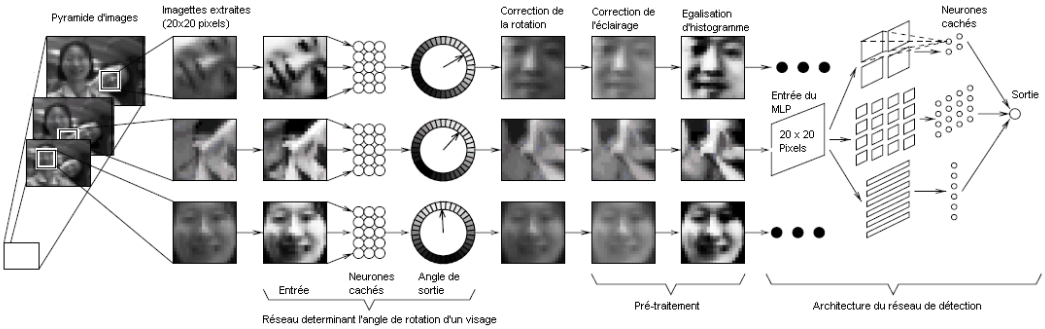


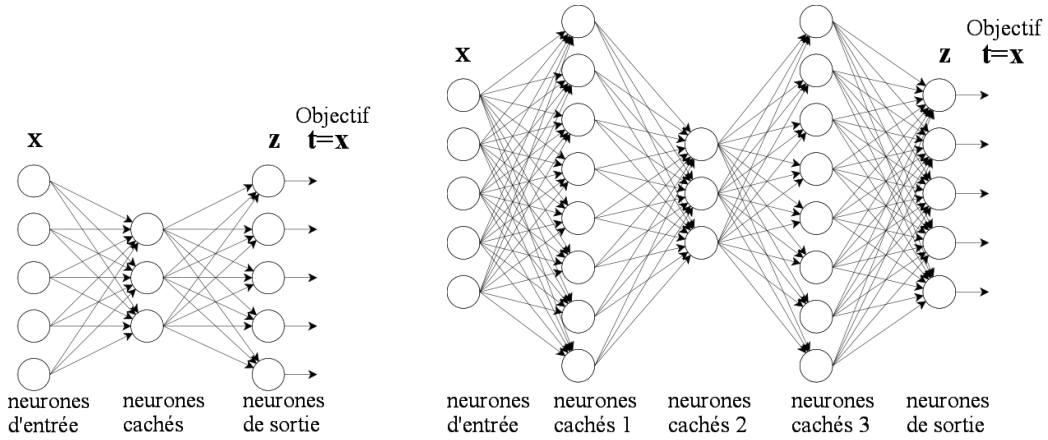
FIGURE 2.18 – Système de détection de visages invariant par rotation de Rowley *et al* . Un premier MLP est utilisé pour déterminer l'angle de rotation d'un visage présenté en entrée du système. La rotation est ensuite corrigée et après un prétraitement des images, un second MLP est utilisé pour déterminer si l'image d'entrée est un visage ou pas.

Dans [48] Rowley *et al* ajoutent au système de détection, un MLP capable de détecter l'angle de rotation d'un visage. Ainsi chaque imagette présentée en entrée du système voit son orientation corrigée de manière à pouvoir détecter des images de visages ayant subi une rotation (figure : 2.18). Inspiré par Baluja [6], ce MLP utilise une image en Niveaux de Gris en entrée et 36 neurones de sortie 'k' représentant chacun un angle de rotation de $k \times 10°$.

La grande variété d'architectures possibles pour un MLP en fait donc un outil extrêmement performant en traitement d'image. Un MLP peut aussi être utilisé afin d'extraire des descripteurs d'une image. De tels réseaux sont appelé réseaux auto-associatifs et effectuent une opération parfois appelé PCA non linéaire pour ses similitudes avec L'Analyse en Composantes Principales.

2.3.7.2 Perceptron Multicouche appliqué à l'extraction de descripteurs, réseaux auto-associatifs

Les réseaux de neurones auto-associatifs (AANN Auto-Associative Neural Networks) sont des Perceptrons Multicouches contenant autant de neurones d'entrée que de neurones de sortie (figure : 2.19). Ils sont entraînés de façon à ce que la sortie soit égale à l'entrée du réseau de neurones. La couche de neurones cachés centrale contient moins de neurones que l'entrée et la sortie. La sortie de la couche centrale **u** peut donc être utilisée comme descripteurs permettant de compresser l'information des vecteurs d'entrée. Le vecteur d'entrée **x** pouvant être reconstruit à partir de la partie droite du réseau et du vecteur **u**.



(a) AANN 3 couches (b) AANN 5 couches

FIGURE 2.19 – Réseaux de neurones auto-associatifs à trois couches et cinq couches. La sortie de la couche centrale aussi appelée couche de représentation permet de décrire les vecteurs d'entrée dans un espace de dimension plus réduite.

Dans le cas d'un AANN à trois couches (figure : 2.19a), si les fonctions d'activation sont linéaires, alors Balbi et Hornik [5] ont montré que la projection effectuée est équivalente à une PCA. Le vecteur **u** contient les plus grandes valeurs propres de la matrice de covariance et les poids des vecteurs de sortie correspondent aux vecteurs propres. Dans le cas où les fonctions d'activation sont non linéaires [55], l'AANN n'est plus équivalent à une PCA [13], les résultats restent cependant proches de ceux obtenus avec la PCA. L'utilisation de fonctions d'activation non linéaires n'améliorant notamment pas le taux de compression par rapport à une PCA [13, 53]. Ainsi, un tel système n'apportant pas de réels avantages par rapport à la PCA, on lui préfère généralement l'architecture à cinq couches (figure : 2.19b). Cette architecture permet avec l'utilisation de fonctions d'activation non linéaires de modéliser toute projection non linéaire des données d'entrée, ainsi que la fonction reconstruction correspondante. Cette méthode a été utilisée à de nombreuses reprises en traitement d'image, principalement pour des problèmes de compression et de reconnaissance [89, 87, 61, 127, 27, 93].

Une des raisons qui fait le succès des MLP, outre la variété des architectures possibles est qu'il existe un algorithme d'apprentissage relativement simple et efficace nommé algorithme de Rétropropagation que nous allons détailler dans la section suivante.

2.3.7.3 Entraînement d'un Perceptron Multicouche

Nous avons vu que toute fonction peut être modélisée à partir d'un Perceptron Multicouche. Dans cette section nous abordons le problème de déterminer les poids du MLP à partir d'un ensemble d'échantillons exemples $\mathbf{x_i}$ et de la sortie souhaitée $\mathbf{t_i}$ correspondante.

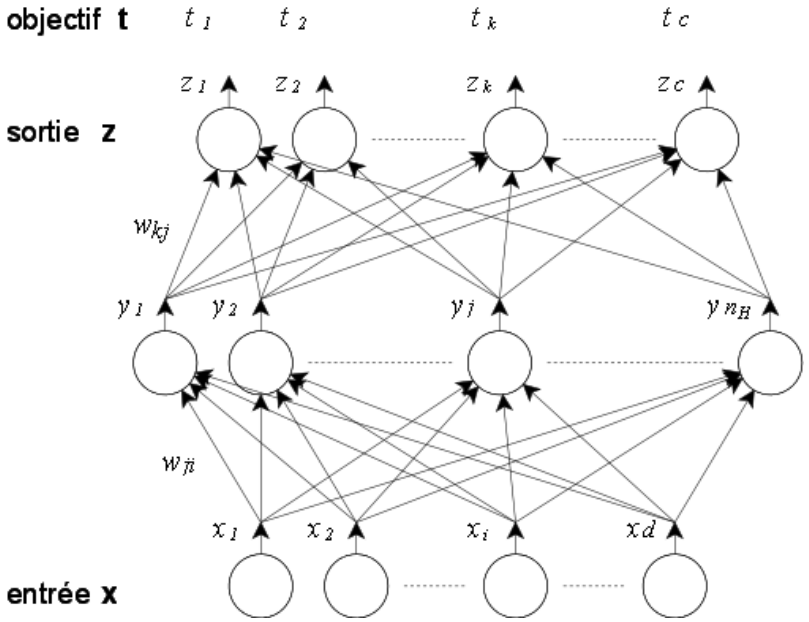


FIGURE 2.20 – MLP à trois couches entièrement connecté et notations utilisées. $\mathbf{x} = (x_1, \ldots, x_d)^T$ est un échantillon présenté au système. $\mathbf{z} = (z_1, \ldots, z_c)^T$ est la sortie correspondante et $\mathbf{t} = (t_1, \ldots, t_c)^T$ est la sortie souhaitée pour l'échantillon $\mathbf{x}$. Les poids w_{kj} correspondent aux synapses reliant le neurone de sortie k au neurone caché j. De même, les poids w_{ji} sont liés aux synapses reliant le neurone d'entrée i au neurone caché j. y_j est la sortie du j^{ieme} neurone caché. net_j est le produit scalaire entre les entrées du j^{ieme} neurone caché et les poids correspondants $y_j = \varphi(net_j)$. De même $z_k = \varphi(net_k)$.

Il existe de nombreuses méthodes permettant d'entraîner un réseau de neurones. L'utilisation d'un algorithme d'apprentissage plutôt qu'un autre dépend aussi bien de l'architecture du réseau et de la forme des exemples d'apprentissage que du problème à traiter. Cependant la méthode de Rétropropagation introduite par Rumhelhart *et al* [111] reste l'algorithme le plus utilisé et sans doute le plus universel pour entraîner un MLP. Afin de simplifier les notations, et les explications, nous détaillerons

ici la méthode pour un MLP à trois couches entièrement connecté (figure : 2.20), la méthode se généralisant aisément à toute architecture de MLP. L'algorithme de Rétropropagation est basé sur la minimisation par la méthode de descente par gradient de l'erreur quadratique E entre la sortie souhaité du MLP $\mathbf{t}$ et sortie réelle $\mathbf{z}$:

$$E(\mathbf{w}) = \frac{1}{2} \|\mathbf{z} - \mathbf{t}\|^2 \tag{2.84}$$

$\mathbf{w}$ étant l'ensemble des poids associés aux neurones du réseau. Le biais b des fonctions d'activation d'un neurone est souvent considéré comme un poids $w_0 = b$, le vecteur d'entrée du neurone ayant été augmenté par la valeur $x_0 = 1$. Afin de simplifier les notations nous noterons dorénavant w_{pq} les poids associés aux synapses reliant le neurone p d'une couche supérieure au neurone q de la couche inférieure. w_{p0} représentant le biais du neurone p de la couche supérieure. Les poids sont initialisés avec des valeurs aléatoires et sont ensuite itérativement modifiés dans la direction permettant de réduire l'erreur $E(\mathbf{w})$:

$$\Delta \mathbf{w} = -\eta \nabla_{\mathbf{w}} E \tag{2.85}$$

Ainsi chaque poids w_{pq} est itérativement modifié de la valeur :

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}} \tag{2.86}$$

Ou $\eta > 0$ est le taux d'apprentissage et détermine le changement relatif des poids entre chaque itération. Sa valeur doit être choisie suffisamment grande pour minimiser le nombre d'itérations nécessaires pour atteindre un minimum, et suffisamment faible pour assurer la convergence. L'algorithme de Rétropropagation permet de calculer simplement $\frac{\partial E}{\partial w_{pq}}$ malgré la compléxité et le nombre de paramètres de la fonction $E(\mathbf{w})$.

En effet, soit le produit scalaire entre les poids et les entrées d'un neurone $net_k = \mathbf{w_k} \cdot \mathbf{y}$, et $\mathbf{net} = (net_1, \ldots, net_k, \ldots, net_c)^T$, alors on peut écrire pour les poids reliés à la derniere couche de neurone du MLP :

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}} \tag{2.87}$$

Avec $\delta_k = -\frac{\partial E}{\partial net_k}$ la sensibilité du k^{ieme} neurone. Si $\varphi(.)$ la fonction d'activation du neurone est dérivable, alors :

$$\delta_k = -\frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial net_k} = (t_k - z_k)\, \varphi_k'\,(net_k) \tag{2.88}$$

Et $\frac{\partial net_k}{\partial w_{kj}} = y_j$ la j^{ieme} entrée du neurone (correspondant pour un MLP entièrement connecté à la sortie du j^{ieme} neurone de la couche précédente).

Ainsi, nous pouvons aisément calculer la mise à jour des poids associés aux synapses liés aux neurones de sortie :

$$\Delta w_{kj} = \eta \delta_k y_j = \eta \left(t_k - z_k\right) \varphi'_k \left(net_k\right) y_j \tag{2.89}$$

La règle d'apprentissage pour les poids w_{ji} entre la couche d'entrée et la couche cachée est plus subtile. L'intérêt de l'algorithme de Rétropropagation est que l'on peut déduire Δw_{ji} de la sensibilité δ_k et des poids w_{kj} des neurones de la couche supérieure :

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} \underbrace{\frac{\partial y_j}{\partial net_j}}_{\varphi'_j(net_j)} \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{x_i} \tag{2.90}$$

$$\frac{\partial E}{\partial y_j} = \nabla_{\mathbf{net}} E \cdot \frac{\partial \mathbf{net}}{\partial y_j} = -\sum_{k=1}^{c} \delta_k w_{kj} \tag{2.91}$$

$$\Delta w_{ji} = \eta \underbrace{\left[\sum_{k=1}^{c} \delta_k w_{kj}\right] \varphi'_j \left(net_j\right)}_{\delta_j} x_i \tag{2.92}$$

$$\Delta w_{ji} = \eta \delta_j x_i \tag{2.93}$$

Des équations 2.89 et 2.93, on peut déduire l'algorithme d'apprentissage pour un MLP à trois couches. On retrouve généralement deux variantes, la méthode de Rétropropagation stochastique (Stochastic Backpropagation : Algo 5) et la Rétropropagation par lots (Batch Backpropagation : Algo 6). L'algorithme stochastique met à jour les poids $\mathbf{w}$ du MLP à chaque présentation d'un échantillon exemple $\mathbf{x}$ au système. L'algorithme par lots ne met à jour les poids du réseau, qu'une fois l'ensemble des N échantillons exemples présentés au système. L'algorithme stochastique est généralement préféré car il converge plus vite que l'algorithme par lots, surtout pour des bases d'apprentissage contenant de nombreux échantillons. L'algorithme par lots permet quand à lui d'utiliser des méthodes d'apprentissage du second ordre difficilement utilisables avec la méthode stochastique, et se révélera ainsi plus performant pour le traitement de certain problèmes.

Par analogie avec l'équation 2.93, il est aisé de généraliser les algorithmes à n'importe quel MLP. En effet, la sensibilité δ_j d'un neurone se déduisant de la sensibilité δ_k des neurones de la couche supérieure (figure : 2.21), on peut calculer par récurrence les variations des poids du réseaux en partant de la dernière couche jusqu'a l'entrée du réseau pour un nombre quelconque de couches du MLP.

Différentes conditions d'arrêt peuvent être utilisées pour l'algorithme de Rétropropagation. Nous avons choisie ici, une condition d'arrêt liée à un nombre maximum d'itérations et une valeur seuil θ de l'erreur quadratique sur l'ensemble de la base référence $J(\mathbf{w})$. Un autre critère possible est par exemple $\|\nabla J(\mathbf{w})\| < \theta$. L'algorithme s'arrête alors quand l'erreur quadratique $J(\mathbf{w})$ approche un minimum local.

Algorithm 5 Stochastic Backpropagation

initialisation : η, $\mathbf{w}$, n_H, θ, max_{iter}, $m = 0$
while $m < max_{iter}$ and $J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{z_i} - \mathbf{t_i}\|^2 > \theta$ **do**
  $m = m + 1$
  $\mathbf{x} \leftarrow i^{ieme}$ échantillon exemple
  $\delta_k = (t_k - z_k)\,\varphi'_k\,(net_k)$
  $\delta_j = [\sum_{k=1}^{c} \delta_k w_{kj}]\,\varphi'_j\,(net_j)$
  $w_{kj} = w_{kj} + \eta\delta_k y_j$
  $w_{ji} = w_{ji} + \eta\delta_j x_i$
end while
return $\mathbf{w}$

Algorithm 6 Batch Backpropagation

initialisation : η, $\mathbf{w}$, n_H, θ, max_{iter}, $m = 0$
while $m < max_{iter}$ and $J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{z_i} - \mathbf{t_i}\|^2 > \theta$ **do**
  $m = m + 1$
  $\Delta w_{kj} = \Delta w_{ji} = 0$
  for i=1 to N **do**
    $\mathbf{x} \leftarrow$ échantillon exemple choisie aléatoirement
    $\delta_k = (t_k - z_k)\,\varphi'_k\,(net_k)$
    $\delta_j = [\sum_{k=1}^{c} \delta_k w_{kj}]\,\varphi'_j\,(net_j)$
    $\Delta w_{kj} = \Delta w_{kj} + \eta\delta_k y_j$
    $\Delta w_{ji} = \Delta w_{ji} + \eta\delta_j x_i$
  end for
  $w_{kj} = w_{kj} + \Delta w_{kj}$
  $w_{ji} = w_{ji} + \Delta w_{ji}$
end while
return $\mathbf{w}$

De nombreuses variations de l'algorithme de Rétropropagation ont été proposées dans la littérature. Le but principal étant d'accélérer la convergence, ou d'améliorer la généralisation du MLP (améliorer la classification sur des exemples ne figurant pas dans la base d'apprentissage). Parmi ces variantes on peut citer :

Validation croisée/(**Cross validation**), elle permet de minimiser les problèmes des sur-apprentissage et d'améliorer ainsi la généralisation du classifieur, cette méthode consiste à calculer $J(\mathbf{w})$ à partir d'une partie de la base d'exemples non utilisée pour l'apprentissage du MLP.

Weight Decay [133], est une autre méthode permettant d'améliorer la généralisation du MLP. Elle consiste à ajouter un terme de régularisation à l'erreur quadratique :

$$E = \frac{1}{2}\|\mathbf{z} - \mathbf{t}\|^2 + \frac{\alpha}{2}\|\mathbf{w}\|^2 \tag{2.94}$$

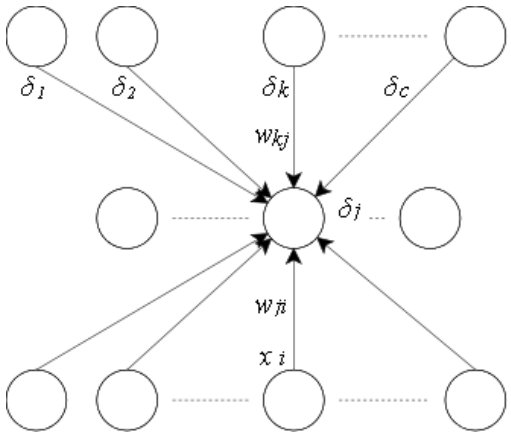


FIGURE 2.21 – La sensibilité δ_j d'un neurone se déduit directement de la sensibilité δ_k des neurones de la couche supérieure et des poids associés w_{kj} et w_{ji} liés au neurone : $\delta_j = [\sum_{k=1}^{c} \delta_k w_{kj}]\, \varphi'_j\,(net_j)$.

Avec $\alpha > 0$. Cette méthode permet ainsi d'éviter de trop grandes valeurs des poids w_{pq} avec pour conséquence une meilleure généralisation du MLP.

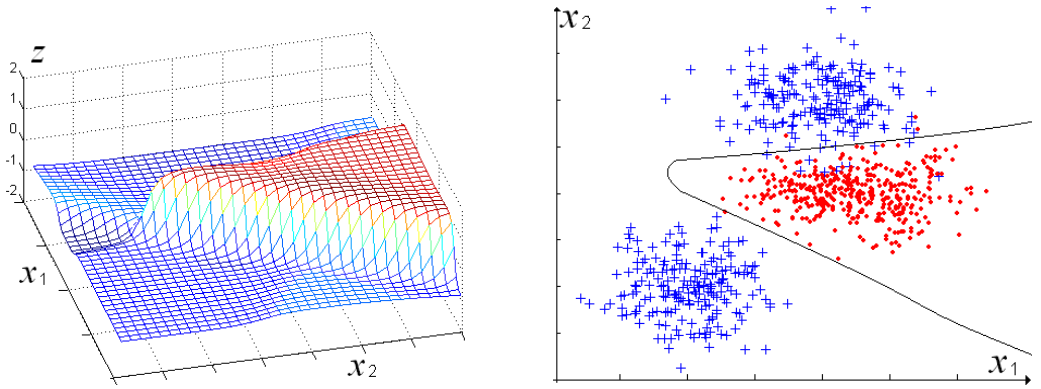
Momentum [96], est une variante qui permet d'accélérer la convergence de la Rétropropagation. L'idée est ici de calculer la variations $\Delta w_{pq}(n)$ des poids w_{pq} à la n^{ieme} itération en tenant compte des variations des poids à l'itération précédente.

$$\Delta w_{pq}(n) = -\eta \frac{\partial E}{\partial w_{pq}(n)} + \alpha \Delta w_{pq}(n-1) \quad 0 < \alpha < 1 \tag{2.95}$$

Ceci a un effet de lissage sur les variations des poids entre chaque itération et permet généralement d'améliorer la vitesse de convergence.

Enfin on peut trouver de nombreuses méthodes agissant sur la valeur du taux d'apprentissage η [114, 7, 8, 94, 76, 54]. Le principe étant de faire varier la valeur de η au cours des itérations successives de l'algorithme de Rétropropagation afin d'accélérer et d'améliorer la convergence.

Ces méthodes d'optimisation de l'algorithme de Rétropropagation ne sont pas toujours compatibles avec l'algorithme stochastique. De plus, leur intérêt est fortement dépendant de la forme du MLP, du nombre d'échantillons exemples ainsi que du type d'information à classer. Dans la majorité des problèmes, l'algorithme de Rétropropagation stochastique est préféré. La figure 2.22 montre la sortie obtenue pour un MLP entraîné par la méthode de Rétropropagation stochastique pour un problème à deux classes dans un espace bidimensionnel. Nous pouvons voir que les frontières définies par le MLP sont proches de ce que l'on pouvait espérer.



(a) courbe tridimensionnelle de la sortie d'un MLP

(b) frontière générée par le MLP

FIGURE 2.22 – Exemple de sortie d'un Perceptron Multicouche utilisé pour séparer deux classes dans un espace bidimensionnel. Le MLP possède 10 neurones cachés et un seul neurone de sortie. Les fonctions d'activation utilisées sont des tangentes hyperboliques. Le système est entraîné de manière à ce que la sortie z du MLP soit -1 ou 1 en fonction de la classe. La frontière correspond aux valeurs de l'entrée $\mathbf{x}$ telles que $g(\mathbf{x}) = z = 0$.

2.3.7.4 Perceptron Multicouche avec peu d'exemples d'apprentissage

Nous retrouvons dans la littérature un certain nombre de travaux en rapport avec la classification avec peu d'exemples à base de Perceptrons Multicouches. Les deux principaux problèmes liés au perceptron multicouche fonctionnant avec une base d'exemples réduite sont en premier lieu la généralisation, et en second lieu le risque de sur-apprentissage. Les solutions au premier problème sont communes à la grande majorité des classifieurs et consistent principalement à minimiser la dimension des vecteurs ou à éviter les biais d'apprentissage [103, 2]. Une autre solution au problème de généralisation est l'adaptation de la forme du perceptron multicouche au problème traité comme par exemple minimiser le nombre de connexions du réseau comme le font Rowley *et al* pour la détection de visages [47] ou encore l'extraction d'information avec un plus haut niveau d'abstraction en utilisant notamment les réseaux profonds [10]. Pour éviter le sur-apprentissage on peut en premier lieu minimiser la dimension du réseau de neurones et en particulier le nombre de neurones cachés [64, 79] ou encore stopper l'apprentissage avant que le MLP ne sur-apprenne [74]. Finalement, nous constatons qu'il existe de nombreux moyen de gérer les bases d'apprentissage réduites mais seules les expérimentations peuvent nous indiquer quelles méthodes sont les plus performantes pour une application particulière.

Le Perceptron Multicouche constitue aujourd'hui un des classifieurs les plus utilisé. La variété de ses architectures et l'existence de méthodes d'apprentissages simples et efficaces en font un outil performant et adaptable à de très nombreux problèmes, notamment la détection de visages. Une autre méthode de classification basée sur les

réseaux de neurones ayant donné de bons résultats en détection est le SNOW (Sparse Network Of Winnows).

2.3.8 SNOW

L'architecture de réseau de neurones appelée SNOW à été mise au point en 1998 par Roth [108] et appliquée à la détection de visages en 2000 [109]. Le SNOW est un réseau de neurones consistant en une seule couche de neurones avec une fonction d'activation linéaire. A la différence d'un MLP, le vecteur d'entrées ne contient que des valeurs binaires (0 ou 1). La sortie d'un neurone j est la somme des poids w_{ji} avec $x_i = 1$ (figure : 2.23).

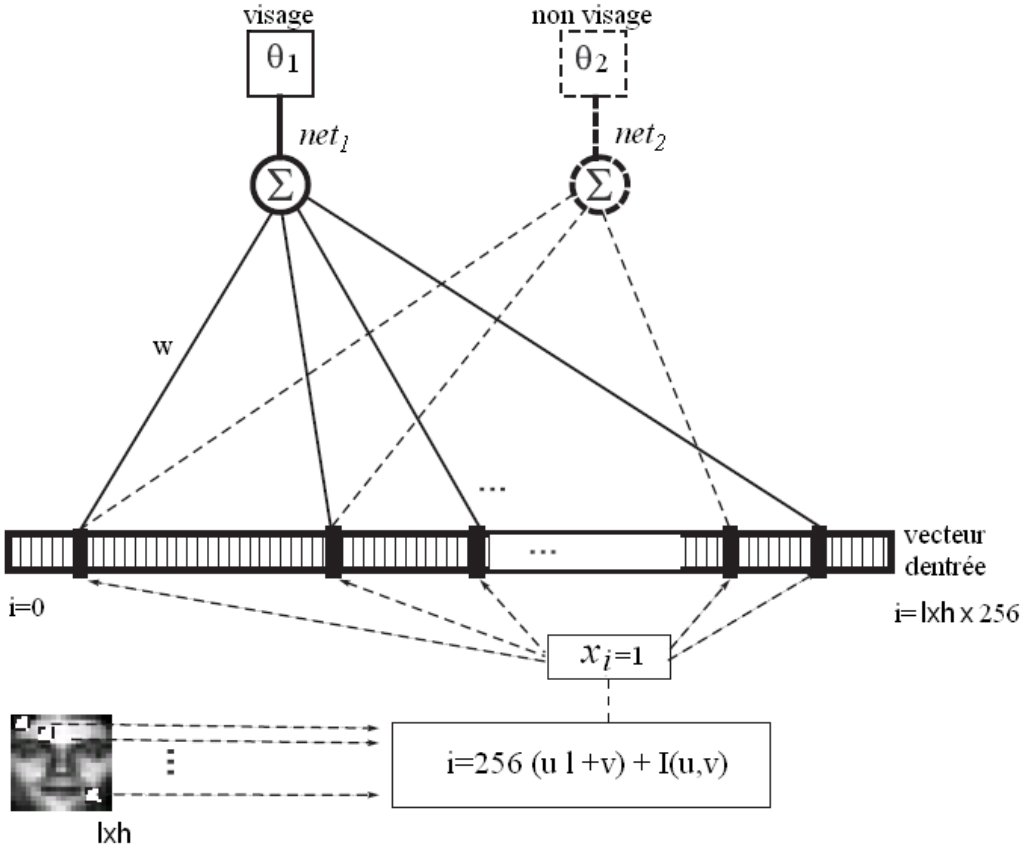


FIGURE 2.23 – SNOW utilisé dans un système de détection de visages. L'image en 256 Niveaux de Gris est représentée sous la forme d'un vecteur $\mathbf{x}$ binaire de dimension hauteur $\times$ largeur $\times$ 256.

Si nous prenons l'exemple du détecteur de visages, l'image d'un visage en Niveaux de Gris de dimension $h \times l$ et pour lequel l'intensité I de chaque pixel en (u, v) est codée sur 8 bits ($0 \leq I(u, v) < 256$). Le vecteur d'entrée $\mathbf{x}$ du SNOW est alors de dimension $h \times l \times 256$. la valeur du i^{ieme} élément de $\mathbf{x}$ est alors 1 pour les valeurs de $i = (u \times l + v) \times 256 + I(u, v)$ et 0 sinon. Sur une image de 20×20 pixels on

arrive à un vecteur d'entrée de dimension $d = 102400$. Cependant, seulement 400 éléments de $\mathbf{x}$ sont égaux à 1 (actifs), limitant le calcul de la sortie d'un neurone à une somme de 400 éléments. Ainsi, les temps de calcul du réseau de neurones ne sont pas impactés par la grande dimension du vecteur d'entrée. Le système de détection de visages de Yang *et al* [109] comporte seulement deux neurones, si la sortie net_1 du premier neurone est supérieure à la sortie net_2 du second, alors, un visage est détecté. Dans le cas contraire l'échantillon d'entrée est classé dans la catégorie 'non visage'. Avec seulement deux neurones, et une base d'apprentissage de 1681 visages, ce système a montré de très bon résultats en détection de visages avec des taux de détection équivalents à ceux obtenus avec un MLP [47] ou AdaBoost [131].

2.3.8.1 Apprentissage

Différentes méthodes d'apprentissage peuvent être utilisées pour entraîner un SNOW, y compris l'algorithme de Rétropropagation, ce réseau de neurone pouvant être considéré comme un MLP à deux couches avec des fonctions d'activation linéaires. Nous décrirons ici la méthode utilisée dans [109], elle même dérivée de la méthode de mise à jour des poids introduite par Littlestone dans [72]. La méthode utilisée est dite en ligne et basée décision, c'est à dire que les poids sont mis à jour à la présentation de chaque nouvel échantillon labellisé, uniquement dans le cas ou cet échantillon est mal classé (on dira de cet échantillon qu'il est *pertinent* pour l'apprentissage).

A chaque neurone est associé un seuil θ_j. la sortie z_j du neurone prend la valeur 1 si $net_j = \sum_i x_i w_{ji} > \theta_j$ et 0 sinon. On définie en plus deux paramètres de mises à jour $\alpha > 1$ et $0 < \beta < 1$. Enfin, à chaque échantillon exemple $\mathbf{x}$ présenté au système est associé un objectif t_j pour chaque neurone de sortie. La règle de mise à jour des poids est alors définie par l'algorithme 7.

Algorithm 7 Littlestone Winnow Update Rule

initialisation : α, β, θ_j
$\mathbf{x} \leftarrow$ échantillon exemple choisie aléatoirement
if $z_j \neq t_j$ **then**
  if $z_j = 1$ **then**
    $\forall i \ / x_i = 1, \quad w_{ji} = \beta x_i w_{ji}$
  else
    $\forall i \ / x_i = 1, \quad w_{ji} = \alpha x_i w_{ji}$
  end if
end if
return $\mathbf{w} = (\ldots, w_{ji}, \ldots)^T$

Ainsi, si un échantillon est mal classé par le neurone j car la somme des poids liés aux éléments actifs de l'entrée $\mathbf{x}$ est trop grande, alors la valeur de ces poids est diminuées d'un facteur β. Si un échantillon est mal classé car la somme de ces poids est trop faible, ces derniers voient leur valeur augmentée d'un facteur α.

Cet algorithme d'apprentissage s'avère très efficace malgré le très grand nombre de poids associés à un neurone. En effet, à chaque itération seule une très faible partie des poids est mise à jour (400 sur les 102400 que compte un neurone pour le détecteur de visages). De plus, le nombre d'échantillons pertinents pour l'apprentissage (mal classés) ne croit que de façon logarithmique par rapport au nombre total d'échantillons à classer. Enfin cette méthode est connue pour apprendre efficacement toute fonction de séparation linéaire et pour être robuste en présence de divers types de bruit [73, 59].

Une variante de ce détecteur de visages décrite dans [109] consiste à extraire les descripteurs booléens de l'image à tester, non pas en discrétisant directement la valeur du niveau de gris de chaque pixel, mais en utilisant des descripteurs représentant une information multi-échelle. Une image de 20×20 pixels est ainsi divisée en sous-régions de 1×1, 2×2, 4×4 et 10×10 pixels conduisant à un nombre de descripteurs actifs (élément de $\mathbf{x}$ égaux à 1) de $400 + 100 + 25 + 4 = 529$. Chaque sous-région est décrite par la valeur moyenne de ses pixels ainsi que leur variance. Les moyennes et les variances sont encodées dans 100 classes distinctes conduisant à un vecteur binaire d'entrée de dimension 529×100 éléments. Cette méthode à montré des résultats très légèrement supérieurs à celle utilisant directement la valeur des pixels en Niveaux de Gris.

2.3.9 Reseaux Convolutionnels

Une image a une structure bidimensionnelle ou la valeur de chaque pixel est fortement corrélée avec les pixels voisins. Les meilleurs systèmes de détection dans une image tiennent d'ailleurs généralement compte des particularités liées à une image. Viola et Jones [131] utilisent des fonctions de Haar effectuant ainsi des corrélations locales, Sung et Poggio [125] projettent les images dans des sous-espaces et Rowley *et al* [47] utilisent un MLP non complètement connecté découpant l'image en sous-régions (bandes horizontales et sous-régions rectangulaires). La structure du réseau convolutionnel (CNN pour Convolutional Neural Networks) inspiré du système visuel du chat [52] permet de l'utiliser directement sur une image en Niveaux de Gris sans prétraitement d'image ou extraction de descripteurs. Pour ce faire, l'architecture du réseau convolutionnel combine successivement des étapes de convolutions et de sous échantillonnages. L'étape de convolution est effectuée par la combinaison de deux techniques. Les champs réceptifs locaux (local receptive fields) permettant d'extraire des formes élémentaires tels que des contours orientés ou des coins [41, 42, 86, 65, 66] et le partage des poids (shared weights) [111, 42, 65] permettant de limiter le nombre de paramètres du réseau de neurones.

La première implémentation d'un réseau de neurones convolutionnel a été proposée par Fukushima [41, 42] et a été appliquée au problème de la reconnaissance de caractères écrits manuellement. Chaque neurone n'est connecté qu'à une sous-région (champ réceptif local) correspondant à un certain nombre de neurones voisins dans la couche de neurones précédente. Ceci constitue donc un extracteur de descripteur local. Cet extracteur de descripteur pouvant être utilisé dans différentes zones d'une image, les poids sont partagés pour toutes les positions possibles du champ réceptif

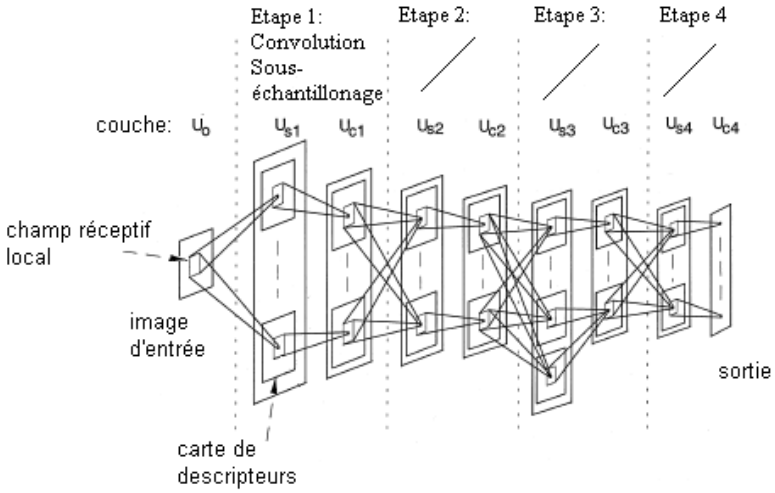


FIGURE 2.24 – Structure de base du Neocognitron : L'image de départ est filtrée, puis sous échantillonnée par des filtres appris (correspondant aux poids des neurones), l'opération est ensuite répétée afin de réduire progressivement la dimension des images (ou carte de descripteurs) et de faire en sorte que les filtres utilisés correspondent à une information de plus en plus spécifique aux images à classer.

local, conduisant donc à convoluer l'image de la couche précédente avec la matrice constituée des poids de chaque neurone. Chaque neurone formant ainsi à la couche suivante l'image filtrée (carte de descripteurs) d'une image de la couche précédente. L'image obtenue est ensuite sous-échantillonnée permettant ainsi de réduire la sensibilité du CNN aux variations d'échelle ou aux translations de l'image d'entrée. Le CNN consiste donc en une succession d'images filtrées puis sous-échantillonnées (le plus souvent par un facteur 2) (figure : 2.24).

De part la structure en couches successives du CNN, l'algorithme de Rétropropagation peut être utilisé pour entraîner ce dernier. dans [67], LeCun *et al* présentent le premier CNN entraîné grâce à l'algorithme de Rétropropagation et l'appliquent à la reconnaissance de caractères. L'architecture de ce CNN plus simple que le Neocognitron est présentée figure 2.25 et constitue souvent une base pour de nombreux systèmes utilisant des CNN.

2.3.9.1 Réseau de neurones convolutionnel pour la détection

Afin d'illustrer l'utilisation des CNN, nous présentons dans cette section le système de détection de visages de Garcia et Delakis [43] qui constitue aujourd'hui, un des systèmes de détection de visages les plus performant, autant en vitesse de calculs qu'au niveau du taux de détection. L'architecture du CFF (Convolutional Face Finder) (figure : 2.26) est inspirée de celle de LeCun *et al* [67]. Le réseau est entraîné pour renvoyer la valeur -1 ou 1 en fonction de la catégorie 'non visage' ou 'visage' des

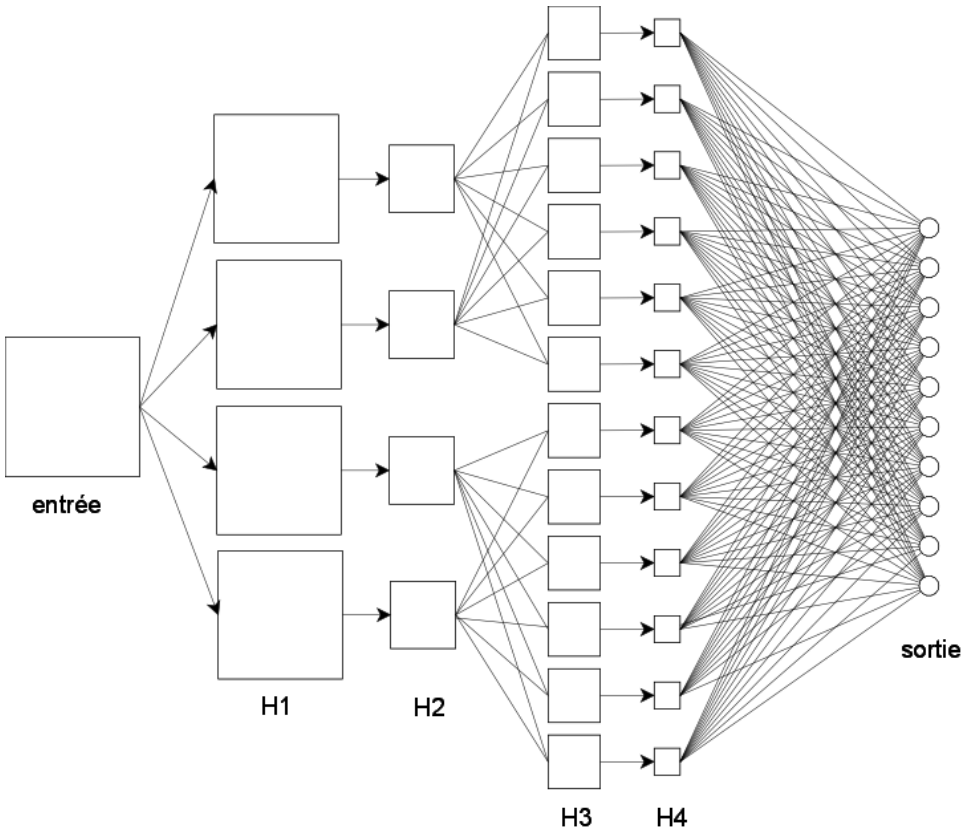


FIGURE 2.25 – Structure du CNN de LeCun pour la reconnaissance d'écriture manuelle de chiffres : L'image de départ est filtrée puis sous-échantillonnée par un facteur 2, respectivement par la couche H1 et H2. L'opération est ensuite réitérée sur les images correspondant à la couche H2 par les couches de neurones H3 et H4. On retrouve ensuite en sortie 10 neurones (un pour chaque chiffre à reconnaître).

images de 32×36 pixels présentées en entrée. La première couche du CFF est constituée de quatre cartes de dimension 28×32. Chaque carte correspondant à l'image d'entrée convoluée à l'aide d'un filtre appris de dimension 5×5 suivie de l'ajout d'un biais b. Le filtre et le biais correspondent aux poids et au biais d'un neurone localement connecté avec 5×5 poids. Ainsi la première couche du CFF est calculée à partir de seulement $(5 \times 5 + 1) \times 4 = 104$ paramètres. Le sous-échantillonnage est effectué à partir d'une lentille 2×2. Chaque neurone calcule la moyenne des quatre entrées multiplié par un coefficient a, ajoute un biais b et renvoie la tangente hyperbolique de ce résultat. Les champs réceptifs locaux (lentilles 2×2) des neurones ne se recouvrent pas et partagent les mêmes poids. Ainsi la carte de descripteurs obtenue est de dimension 14×16 et le nombre de paramètres liés à la couche de sous-échantillonnage est de $2 \times 4 = 8$. Une nouvelle convolution est ensuite effectuée à partir de deux lentilles 3×3 sur chacune des quatre cartes de descripteurs de la couche S1. De plus, une lentille de dimension $3 \times 3 \times 2$ est appliquée à chaque paire

de cartes possibles afin de fusionner les informations des différents descripteurs de la couche précédente. La couche C2 renvoie alors $8 + 6 = 14$ cartes de descripteurs et contient 194 paramètres $((3 \times 3 + 1) \times 8 + (3 \times 3 \times 2 + 1) \times 6)$. Enfin la dernière Couche S2 sous-échantillonne de la même manière que la couche S1, conduisant donc à $14 \times 2 = 28$ paramètres supplémentaires. Chacune des 14 cartes de dimension 7×5 est ensuite reliée à un des 14 neurones de la couche N1 eux même connectés au neurone de sortie.

Ainsi un tel réseau de neurones est capable de distinguer une image de 'visage' d'une image de 'non visage' avec seulement 951 paramètres entraînables, ce qui reste très faible au regard de la dimension des données d'entrée $(36 \times 32 = 1152)$.

De plus les étapes de convolutions et de sous échantillonnages peuvent être effectuées directement sur l'ensemble d'une image et non sur chaque rétine correspondant à l'image de 36×32 pixels traité par le CFF. Ainsi, d'un point de vue complexité de calcul, le réseau convolutionnel est très adapté aux problèmes de détection.

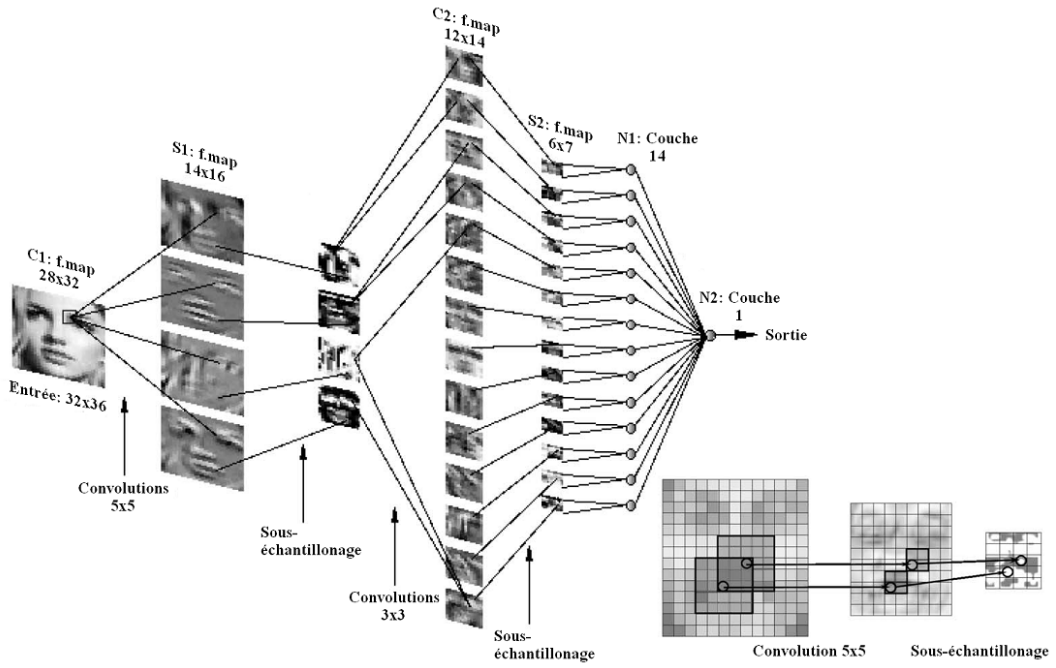


FIGURE 2.26 – Structure du CFF : L'image de départ est filtrée par quatre lentilles 5×5 puis sous-échantillonnée par un facteur 2. L'opération est ensuite réitérée avec deux lentilles 3×3 sur chaque image correspondante de la couche S1 ainsi qu'une lentille $3 \times 3 \times 2$ pour chaque paire d'images possibles de la couche S1, conduisant à 14 cartes de descripteurs dans la couche S2. Finalement le CFF se termine par un perceptron à trois couches constitué des couches S2, N1 et N2.

CHAPITRE 3

Détection par corrélation croisée

3.1 Introduction

La corrélation croisée est une mesure de similarité qui a de nombreux avantages. Cette méthode est facile à implémenter, aisément adaptable à une grande variété de formes et ne requiert pas d'extraction de descripteurs complexes ou une large base d'apprentissage. De plus, la corrélation croisée est l'opération linéaire optimale d'un point de vue rapport signal sur bruit pour détecter la position spatiale ou temporelle d'un signal connu dans un bruit blanc stationnaire [105]. Le maximum du signal résultant correspondant alors à la position la plus probable du signal recherché. L'énergie du bruit dans une image (en détection, le bruit correspond à tout ce qui n'est pas l'objet à détecter) n'est pas stationnaire, ce qui rend la simple corrélation croisée peu performante pour effectuer une détection d'objets. De plus, cette mesure de similarité n'est pas très bien adaptée à la détection d'objets complexes tels que des visages. En effet, la corrélation n'est que peu robuste aux variations d'illumination, d'échelle ou de rotation qui ne peuvent être considérées comme un simple bruit blanc. La corrélation croisée normée est une mesure de similarité basée sur la corrélation mais qui permet grâce à une normalisation des signaux, de rendre plus robuste la détection aux variations d'énergie et de luminosité de l'image. Excepté pour des formes simples [75], la corrélation croisée normée n'a été que peu employée pour la détection d'objets car elle ne permet pas de tenir compte des variations de forme, de couleur, de prise de vue, ou d'échelle. Dans ce chapitre, nous proposons d'associer la corrélation normée croisée à la méthode des plus proches voisins afin de pouvoir représenter les différentes formes que peut prendre l'objet à détecter. Ainsi, un objet sera détecté à une position et échelle donnée si la mesure de similarité maximum entre l'image test et l'ensemble des images exemples est supérieure à un seuil donné. Plus la base d'exemples est grande, mieux cet objet sera modélisé. Cependant, les temps de calculs sont directement proportionnels au nombre d'images exemples disponibles, ce qui limite la taille de la base d'exemples. Nous effectuerons nos expérimentations sur la détection de visages qui bénéficie d'une littérature abondante, les visages étant considérés comme

l'objet complexe par excellence (un visage peut revêtir différentes formes, différentes couleurs, de nombreuses expressions différentes, ainsi que des éclairages très divers). Nous commencerons par décrire les principes de la corrélation croisée ainsi que la corrélation croisée normée et la corrélation croisée normée centrée. Nous étudierons ensuite la corrélation directement appliquée aux images en Niveaux de Gris afin de déterminer l'influence des différentes variations de forme, de position et d'échelle. Ensuite, nous étudierons la corrélation sur des images préalablement traitées par la méthode de Sobel afin d'extraire les contours des images et de diminuer la sensibilité de la mesure de similarité aux variations de luminosité. Finalement, nous introduirons une méthode dérivée de la PCA permettant d'extraire les formes revenant le plus souvent dans une base d'images exemples et de calculer ainsi des filtres adaptés à l'objet que nous souhaitons détecter.

3.2 Principes de la corrélation

La corrélation croisée est aussi connue en statistique pour désigner la covariance de vecteurs aléatoires $\mathbf{x}$ et $\mathbf{y}$. En traitement du signal, la corrélation permet de mesurer la similarité entre deux signaux $x(\mathbf{t})$ et $y(\mathbf{t})$. En traitement d'image, $\mathbf{t}$ est un vecteur à deux dimensions représentant les coordonnées (i, j) des pixels des images. La fonction $s(\boldsymbol{\tau})$ résultante s'écrit :

$$s(\boldsymbol{\tau}) = x(\mathbf{t}) * y(-\mathbf{t}) = \int x(\mathbf{t})\, y(\mathbf{t}+\boldsymbol{\tau})\, d\mathbf{t} \tag{3.1}$$

En pratique, si $x(i,j)$ représente l'image exemple de dimension $h_x \times l_x$ et $y(i,j)$ représente l'image test de dimension $h_y \times l_y$ dans laquelle nous souhaitons effectuer la détection. Alors l'image résultante $s(u,v)$ de dimension $(h_s = h_y - h_x) \times (l_s = l_y - l_x)$ s'ecrie :

$$s(u,v) = \sum_{i=j=0}^{i<h_x, j<l_x} y(u+i, v+j)\, x(i,j) \tag{3.2}$$

La corrélation croisée normée est basée sur la norme $L2$ des images et se calcule ainsi :

$$\sigma_x = \left(\sum_{i=j=0}^{i<h_x, j<l_x} x(i,j)^2 \right)^{\frac{1}{2}} \qquad \sigma_y(u,v) = \left(\sum_{i=j=0}^{i<h_x, j<l_x} y(u+i, v+j)^2 \right)^{\frac{1}{2}} \tag{3.3}$$

$$s_n(u,v) = \frac{1}{\sigma_x \sigma_y(u,v)} \left(\sum_{i=j=0}^{i<h_x, j<l_x} y(u+i, v+j)\, x(i,j) \right) \tag{3.4}$$

Enfin, la corrélation croisée normée centrée correspond à la corrélation croisée normée des images auxquelles on soustrait leur valeur moyenne :

$$m_x = \frac{1}{h_x l_x} \sum_{i=j=0}^{i<h_x, j<l_x} x(i,j) \qquad m_y(u,v) = \frac{1}{h_x l_x} \sum_{i=j=0}^{i<h_x, j<l_x} y(u+i, v+j) \tag{3.5}$$

$$\sigma_x = \left(\sum_{i=j=0}^{i<h_x, j<l_x} [x(i,j) - m_x]^2 \right)^{\frac{1}{2}} \qquad \sigma_y(u,v) = \left(\sum_{i=j=0}^{i<h_x, j<l_x} [y(u+i, v+j) - m_y(u,v)]^2 \right)^{\frac{1}{2}}$$

$$s_{nc}(u,v) = \frac{1}{\sigma_x \sigma_y(u,v)} \left(\sum_{i=j=0}^{i<h_x, j<l_x} [y(u+i, v+j) - m_y(u,v)] [x(i,j) - m_x] \right) \tag{3.6}$$

3.3 Détection par Niveaux de Gris

Dans cette section, nous commençons par mettre en œuvre un système de détection basé sur la corrélation des images en Niveaux de Gris, associé à la méthode des plus proches voisins. Nous utiliserons pour tester ce système la base de données 'Face 1999 (Front)' (Annexe : A.1). Cette base comporte 450 visages de 27 personnes distinctes. Elle est normalement destinée à la reconnaissance de visages plutôt qu'à la détection ; elle est cependant bien adaptée pour comparer les résultats d'un système de détection aussi basique qu'une simple corrélation. Nous utiliserons une base d'exemples composée de 80 visages (figure : 3.1).

Chaque visage exemple est redimensionné à la dimension minimum des visages que nous souhaitons détecter. Les images de test sont successivement sous-échantillonnées avec un facteur 1.2. Nous formons ainsi une pyramide d'images permettant d'effectuer une détection multi-échelle. Si un visage est détecté sur la première image de la pyramide (l'image non sous-échantillonnée), alors la dimension du visage est la même que celle de l'image de référence. Dans le cas général, si un visage est détecté dans une image de la pyramide sous échantillonnée d'un facteur α, alors la dimension du visage dans l'image de test sera celle de l'image de référence multipliée par α. Chaque visage de la base de référence est corrélé avec l'ensemble des images de la pyramide résultant ainsi en un ensemble de cartes de scores de corrélation. Un visage est détecté s'il correspond à un maximum local sur la carte de score et s'il n'est pas superposé à un autre visage avec un score de détection supérieur. Nous considérerons que deux visages sont superposés si l'aire $A1$ formée par l'intersection des deux régions correspondantes divisée par l'aire $A2$ de la superposition des deux régions est supérieure à 0.1. De même, nous considérerons que nous avons effectué une bonne détection si pour les deux régions correspondant respectivement au visage à détecter et à la détection effectuée par le système, $A1/A2$ est supérieur 0.5 (figure : 3.2).

Les résultats de la détection sont influencés par différents paramètres. Le premier d'entre eux est l'utilisation comme mesure de similarité, de la corrélation simple,

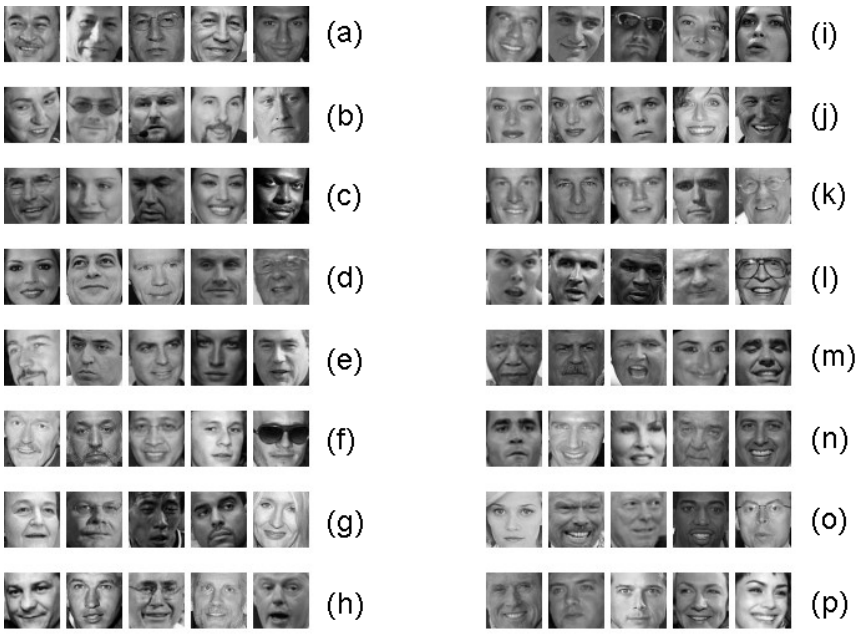


FIGURE 3.1 – Base d'images de référence pour le système de détection par corrélation. Elle est divisée en 16 bases de données de 5 visages respectivement nommées (a) à (p).

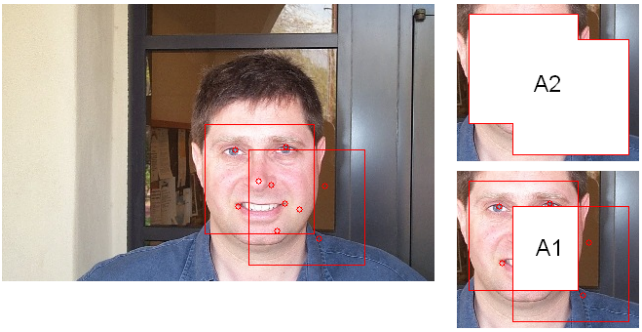


FIGURE 3.2 – Exemple de deux régions superposées. L'aire $A1$ correspond à l'intersection des deux régions, l'aire $A2$ correspond à l'union des deux régions. Si $\frac{A1}{A2} > 0.1$ alors on considère que les deux régions sont superposées. Si $\frac{A1}{A2} > 0.5$ et que une des deux régions correspond à l'objet à détecter manuellement annoté, alors nous effectuons une bonne détection.

la corrélation normée ou bien la corrélation normée centrée. Mais aussi le nombre d'exemples de la base de référence ou la dimension des images exemples. Nous commencerons par nous attacher à déterminer l'influence de la mesure de similarité utilisée.

3.3.1 Influence de la normalisation et du centrage sur la corrélation

La corrélation simple est peu utilisée en détection dans des images, on lui préfère souvent la corrélation normée, ou normée centrée [75]. Nous nous proposons dans cette section d'étudier l'influence de la normalisation et du centrage sur un problème de détection de visages. Comme nous travaillons avec des bases d'exemples de taille réduite, nous utiliserons, afin de tenir compte de l'influence de la base utilisée, quatre bases de données de référence de cinq visages (figure : 3.1) respectivement nommées base (a), (b),(c) et (d). Les images de références utilisées pour l'expérience sont de dimension hauteur fois largeur égal $44 \times 44 = 1936$ pixels. Nous présentons les résultats sous forme de courbes Rappel Précision. Ces courbes rapportent la Précision de la détection (nombre de bonnes détections divisé par le nombre total de détections) par rapport au Rappel (nombre de bonnes détections divisé par le nombre total d'objets à détecter). Ainsi, plus une courbe se rapproche du coin supérieur droit, meilleur est le système de détection.

Nous pouvons constater que la normalisation est indispensable au système de détection. En effet, la courbe correspondant à la corrélation croisée n'est visible sur aucune des quatre figures car le système n'a pu effectuer aucune bonne détection. Ensuite, on remarque que ce système est très dépendant de la base de référence utilisée. Les résultats montrés par les figures 3.3a et 3.3d correspondant aux bases d'exemples (a) et (d) sont très nettement supérieurs aux résultats obtenus avec les bases de référence (b) et (c) (figure : 3.3b,3.3c). Il est par contre difficile ici de départager la corrélation croisée normée de la corrélation croisée normée centrée, c'est pourquoi nous continuerons par la suite à utiliser ces deux mesures.

Enfin, même si un tel système est capable d'effectuer quelques bonnes détections en utilisant la corrélation croisée normée et une base de référence adaptée, les taux de détection restent insuffisants pour une application pratique.

3.3.2 Influence du nombre d'échantillons exemples

Nous avons vu dans la section précédente que les résultats de la détection sont très dépendants de la base d'exemples utilisée. Les mauvais résultats des bases d'exemples (b) et (c) peuvent s'expliquer de deux manières. Soit les exemples de référence sont trop différents des visages de la base de test, soit certains visages de la base exemple peuvent avoir une très forte corrélation avec des images n'étant pas des visages. L'augmentation du nombre d'exemples de la base de référence maximise les chances d'avoir une forte corrélation entre un visage exemple et un visage de la base de test mais augmente aussi le score de corrélation des fausses détections. Afin de déterminer la sensibilité de notre système de détection au nombre d'exemples de la base de référence, nous avons appliqué notre système pour une base de données de cinq, dix, vingt, quarante puis quatre-vingt exemples avec la corrélation normée et la corrélation normée centrée (figure : 3.4).

La base de cinq visages que nous avons utilisée est celle qui a donné les meilleurs résultats de détection, à la fois avec la corrélation normée et avec la corrélation

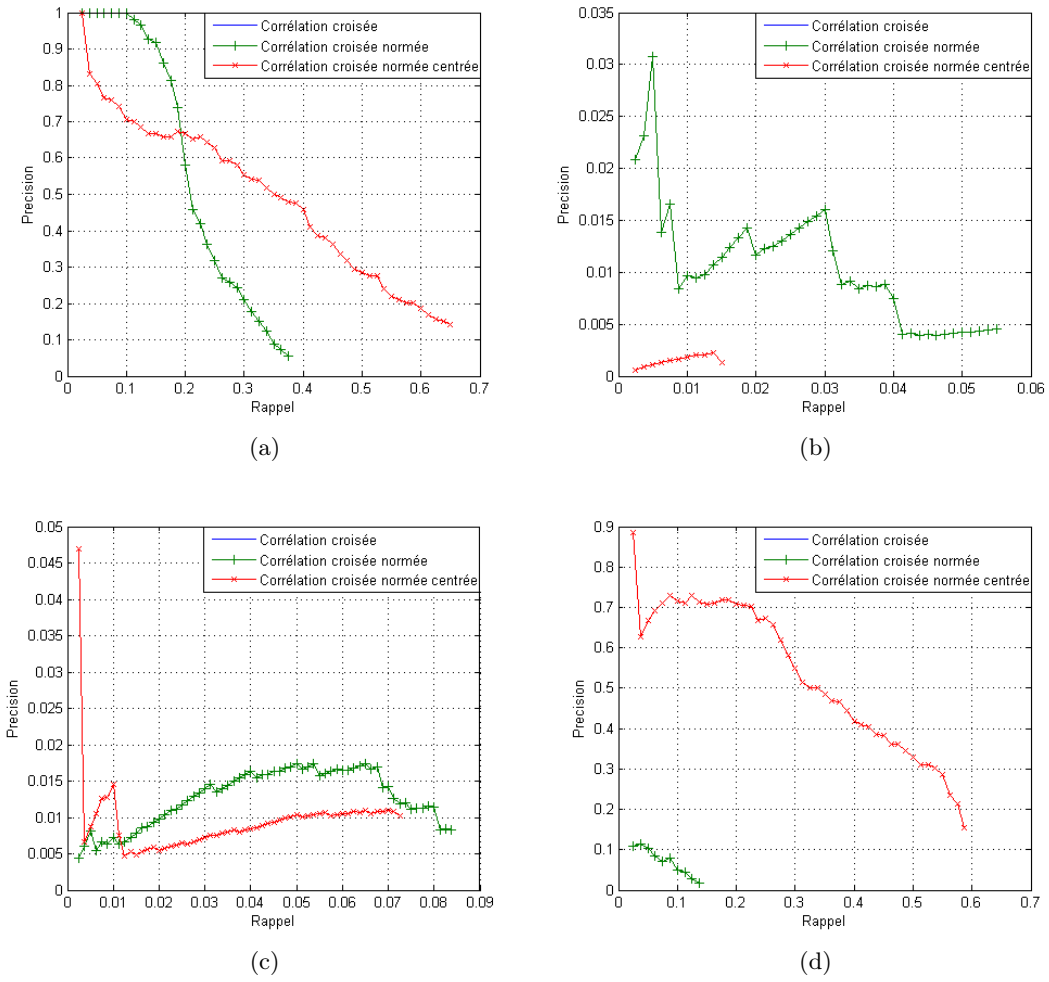
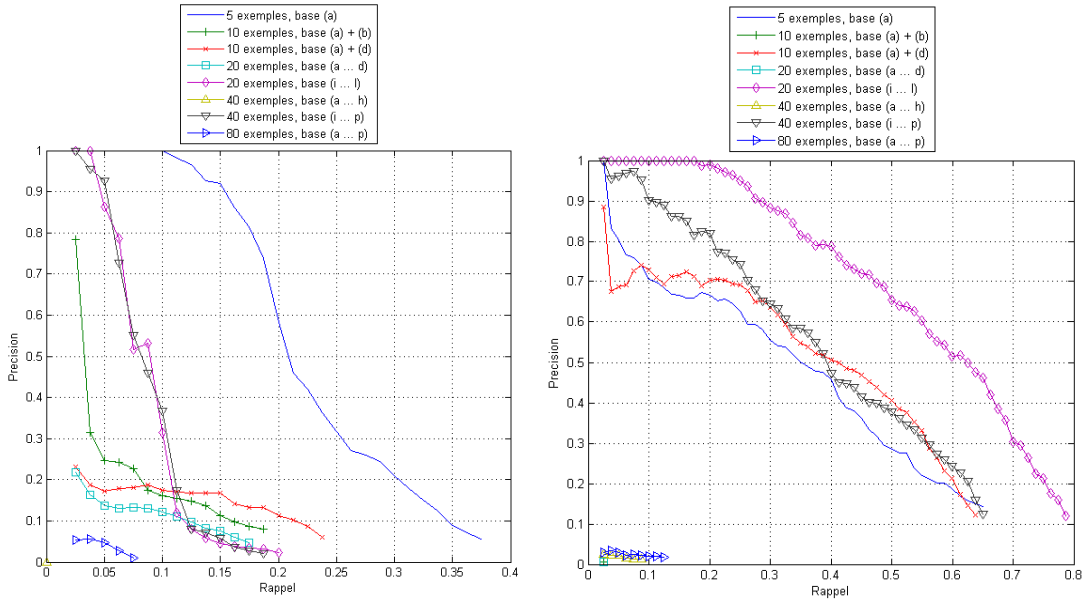


FIGURE 3.3 – Courbes Rappel Précision du système de détection par corrélation croisée des images en Niveaux de Gris. Les figures (a) (b) (c) et (d) ont été obtenues avec quatre bases de référence distinctes de cinq visages. On remarque donc que les résultats sont très dépendants de la base utilisée. La courbe correspondant à la corrélation croisée simple n'est pas visible car un tel système n'a pu effectuer aucune bonne détection.

normée centrée, *i.e*, la base (a). Nous avons ensuite utilisé une base de dix visages constituée des bases d'exemples (a) et (b), puis une seconde base de dix visages constituée des bases (a) et (d). On remarque ainsi que la combinaison de deux bases d'exemples donnant des résultats équivalents (a) et (d) entraîne une amélioration du taux de détection. Cependant, combiner deux bases d'exemples (a) et (b), l'une donnant des résultats nettement supérieur à l'autre, entraîne des taux de détection proches de ceux obtenus avec la base d'exemples la moins adaptée. Ainsi, bien que l'augmentation du nombre d'exemples a tendance à entraîner une amélioration des



(a) Corrélation croisée normée (b) Corrélation croisée normée centrée

FIGURE 3.4 – Courbes Rappel Précision du système de détection par corrélation croisée des images en Niveaux de Gris pour différents nombres d'images de référence.

résultats (les meilleurs résultats étant obtenus pour des bases de 20 et 40 images exemples), les taux de détection sont toujours très dépendants de la base de référence. Ainsi, les taux de détection obtenus avec la base de référence de 80 images sont parmi les plus faibles. Enfin, on peut remarquer que la corrélation croisée normée centrée obtient de meilleurs résultats que la corrélation croisée normée (figure : 3.5).

3.3.3 Influence de la dimension des images exemples

Nous avons vu que la corrélation croisée normée des images en Niveaux de Gris permet, dans une certaine mesure, d'obtenir un détecteur de visages capable de fonctionner avec une base d'exemples très réduite. Cependant, les résultats sont très liés à la base d'exemples utilisée et l'augmentation du nombre d'images exemples n'entraîne pas systématiquement une amélioration du taux de détection et peut même entraîner une nette baisse. Jusqu'à présent, nous avons toujours utilisé des images exemples d'une aire de 1936 Pixels. Nous proposons dans cette section de déterminer l'influence de la dimension des images exemples et en particulier, de vérifier s'il est possible de rendre le système moins sensible à la base de référence utilisée en augmentant la dimension des images de référence. Il faut noter que la dimension des images exemples correspond à la dimension minimale des visages détectables. Dans notre base de données d'images test, les visages sont toujours de dimension supérieure à la dimension des visages de références. La figure 3.6b compare les résultats obtenus pour la base (a) de cinq visages avec des images de référence d'une aire de 484, 961, 1936, 2916 et

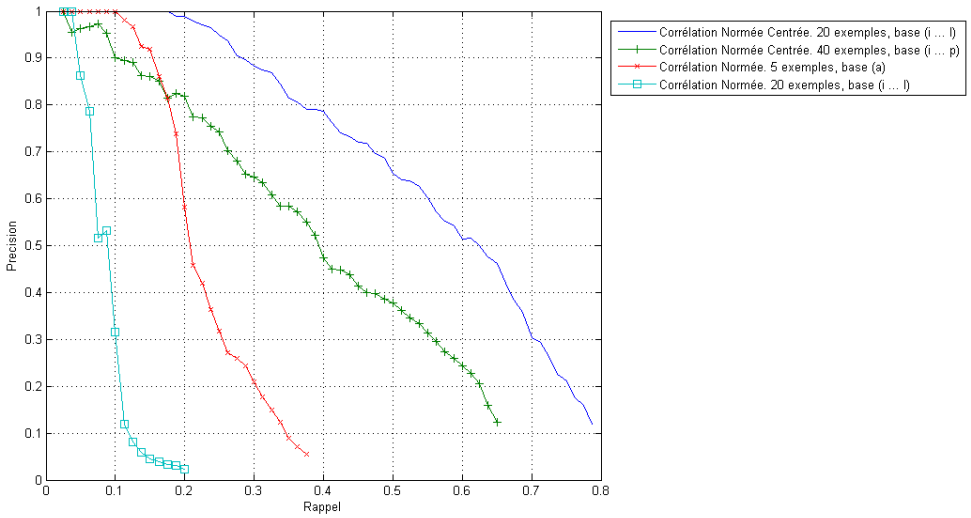
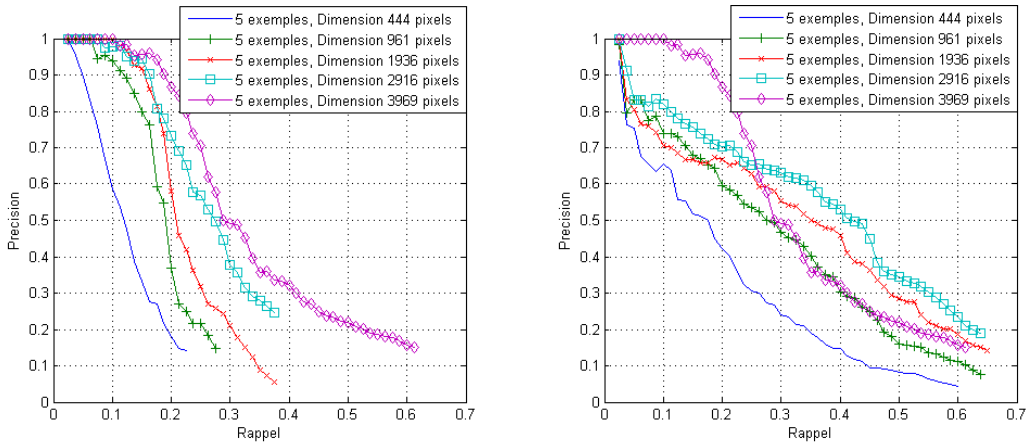


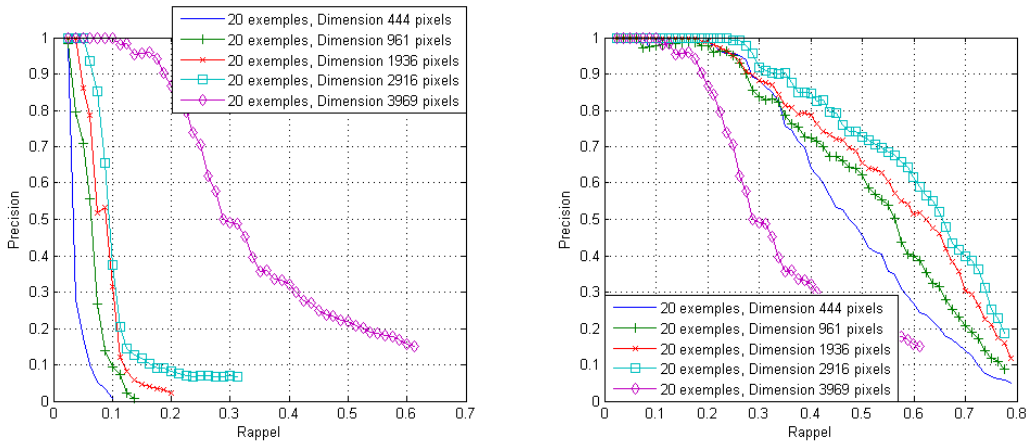
FIGURE 3.5 – Comparaisons des meilleurs résultats pour le système de détection par corrélation croisée normée et croisée normée centrée des images en Niveaux de Gris. La corrélation croisée normée centrée obtient des taux de détection nettement supérieurs à la corrélation normée.

3969 pixels. On remarque une amélioration sensible des résultats avec l'augmentation de la dimension des images de référence, aussi bien pour la corrélation croisée normée centrée que pour la corrélation croisée normée. Le même constat peut être fait pour la base de 20 visages (i ... l) (figure : 3.7b). Nous avons ensuite effectué (figure : 3.8b), les mêmes expérimentations pour le base de 10 visages formée de l'association des bases (a) et (b) qui donnaient des taux de détection très faibles. L'augmentation de la dimension des images exemples entraîne une amélioration des résultats, sans pour autant permettre que cette base d'exemples obtienne des résultats comparables aux deux précédentes.



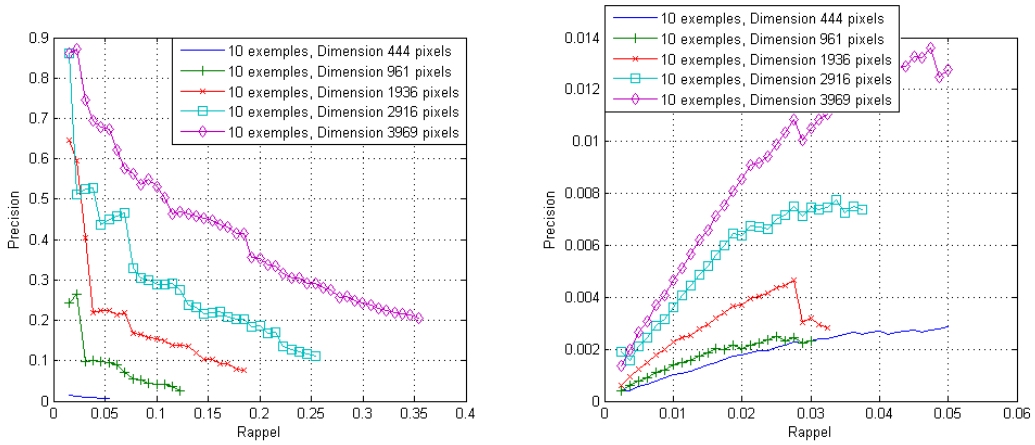
(a) Corrélation croisée normée (b) Corrélation croisée normée centrée

FIGURE 3.6 – Courbes Rappel Précision du système de détection par corrélation croisée des images en Niveaux de Gris avec 5 images exemples. Influence de la dimension des exemples.



(a) Corrélation croisée normée (b) Corrélation croisée normée centrée

FIGURE 3.7 – Courbes Rappel Précision du système de détection par corrélation croisée des images en Niveaux de Gris avec 20 images exemples. Influence de la dimension des exemples.



(a) Corrélation croisée normée (b) Corrélation croisée normée centrée

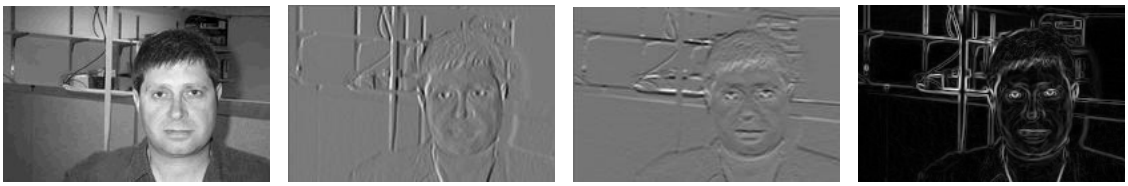
FIGURE 3.8 – Courbes Rappel Précision du système de détection par corrélation croisée des images en Niveaux de Gris avec 10 images exemples. Influence de la dimension des exemples.

3.4 Utilisation de filtres détecteurs de contours

Nous avons vu dans la section précédente que la mesure de similarité basée sur la corrélation des images en Niveaux de Gris permet d'obtenir, sous certaines conditions, des résultats intéressants compte tenue du très faible nombre d'images exemples sur des objets aussi complexes que des visages. Ces résultats sont cependant très dépendants de la base de référence utilisée et ceci indépendamment du nombre ou de la dimension des images exemples. Ainsi, la corrélation croisée des images en Niveaux de Gris semble difficilement utilisable en pratique. Dans cette section, nous utilisons la corrélation croisée sur les contours des images plutôt que sur les images en Niveaux de Gris. Le but est de diminuer la sensibilité de la mesure de similarité aux variations de luminosité et ainsi améliorer les résultats de notre système de détection tout en minimisant l'influence des bases d'exemples utilisées. Dans un premiers temps, nous utiliserons l'algorithme de Sobel pour détecter les contours.

3.4.1 Algorithme de Sobel

L'algorithme de Sobel [122] est connu pour être l'une des méthodes les plus simples et efficaces pour effectuer la détection de contours. Cette méthode consiste à extraire en chaque point d'une image donnée une approximation de la norme du gradient de l'image I afin de faire ressortir les fortes variations de Niveaux de Gris de l'image. Cette méthode consiste à combiner deux filtres (F_x) et (F_y) généralement de dimension 3×3 permettant d'approximer la dérivée horizontale (G_x) et verticale (G_y) de l'image. L'image finale (G) est la norme $(\sqrt{G_x^2 + G_y^2})$ du gradient de l'image traitée. (figure : 3.9).



(a) Image Originale (I) (b) G_x (c) G_y (d) G

FIGURE 3.9 – Exemple d'utilisation de l'algorithme détecteur de contours de Sobel. L'image (a) est l'image originale en Niveaux de Gris, l'image (b) est une approximation du gradient horizontal de l'image, l'image (c) est une approximation du gradient vertical de l'image et enfin l'image (d) représente une approximation de la norme du gradient de l'image et permet d'en faire ressortir les contours.

Les filtres F_x et F_y combinent un lissage gaussien et un opérateur différentiel afin de minimiser la sensibilité au bruit des dérivées horizontales et verticales. Les filtres étant constitués de nombres entiers, le lissage n'est qu'une approximation d'un lissage Gaussien. Les filtres F_x et F_y (3×3) se calculent ainsi :

$$F_x = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \times \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{3.7}$$

$$F_y = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{3.8}$$

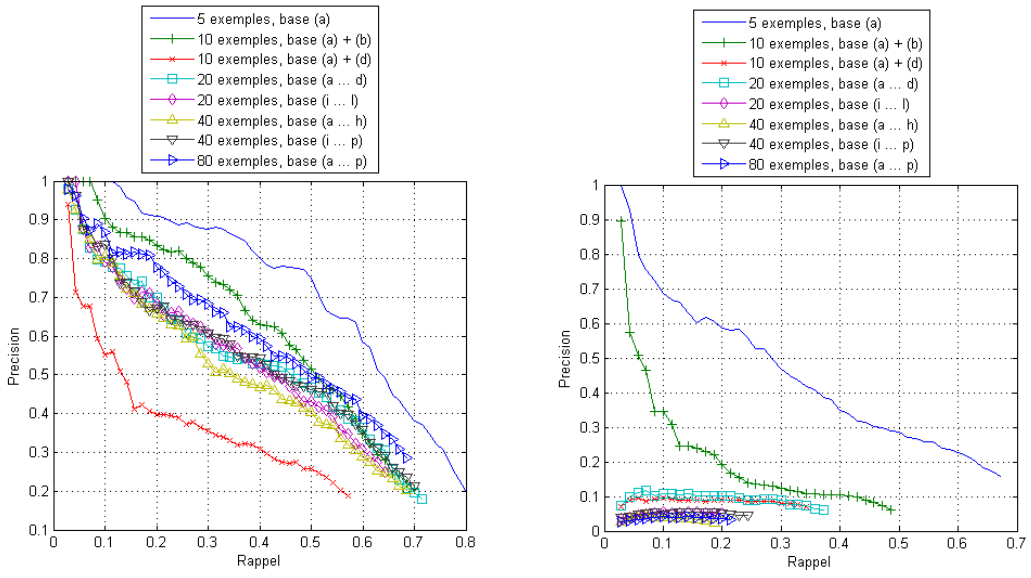
Les images filtrées correspondantes G_x et G_y sont alors calculées par convolution :

$$G_x = F_x * I \tag{3.9}$$

$$G_y = F_y * I \tag{3.10}$$

3.4.2 Corrélation avec utilisation d'un algorithme détecteur de contours

Dans cette section, nous proposons de remplacer l'image en Niveaux de Gris par une image représentant les contours. Pour se faire nous utilisons l'algorithme de Sobel. Le but est d'améliorer les taux de détection et de minimiser la sensibilité du système à la base d'exemples utilisée.



(a) Corrélation croisée normée

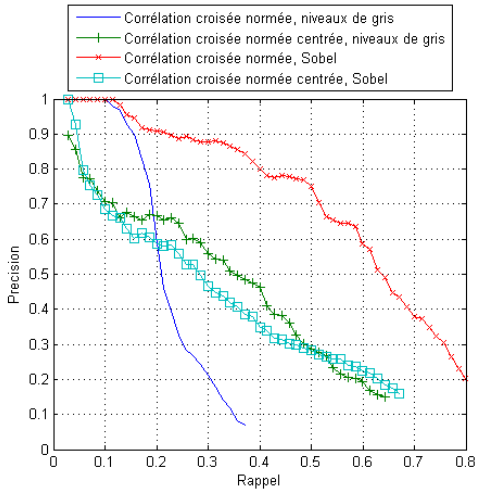
(b) Corrélation croisée normée centrée

FIGURE 3.10 – Courbes Rappel Précision du système de détection par corrélation croisée des contours des images pour différents nombres d'images de référence.

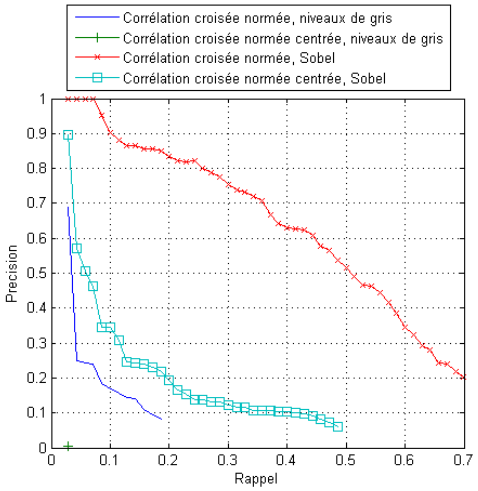
L'utilisation de la corrélation croisée normée comme mesure de similarité devient alors une mesure de la superposition des contours des images. La figure 3.10 montre

que les résultats restent très sensibles à la base de référence utilisée. On remarque notamment que les meilleurs résultats sont obtenus avec 5 et 10 exemples. Cependant, la corrélation croisée normée semble moins sensible à la base utilisée que la corrélation croisée normée centrée. De plus, l'utilisation de l'image représentant les contours en lieu et place de l'image en Niveaux de Gris pour le système de détection par corrélation croisée normée améliore nettement les résultats. En effet, nous obtenons des taux de détection toujours au moins équivalents à l'utilisation de la corrélation croisée normée centrée avec les images en Niveaux de Gris (figure : 3.11). De plus, on remarque que bien que les écarts de résultats en fonction de la base d'exemples utilisée restent importants, ils sont bien inférieurs aux écarts obtenus en utilisant les images en Niveaux de Gris.

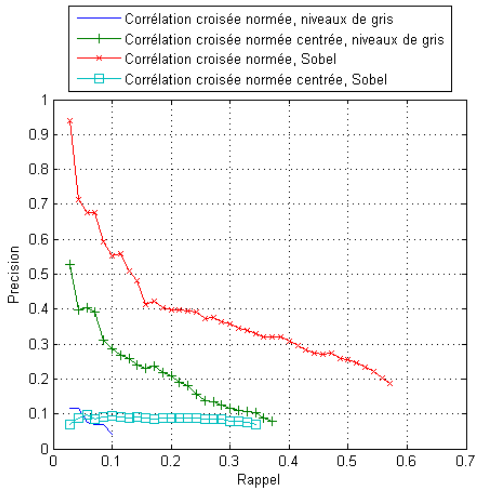
En ce qui concerne la sensibilité de la corrélation croisée normée des contours à la dimension des images exemples (figure : 3.12), on remarque que l'on améliore nettement les taux de détection en augmentant les dimensions des images de référence jusqu'à une aire d'environ 2000 pixels. Les gains pour des dimensions supérieures sont ensuite minimes.



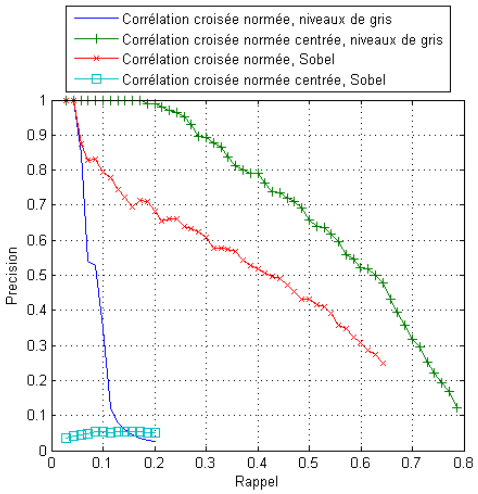
(a) 5 exemples, base (a)



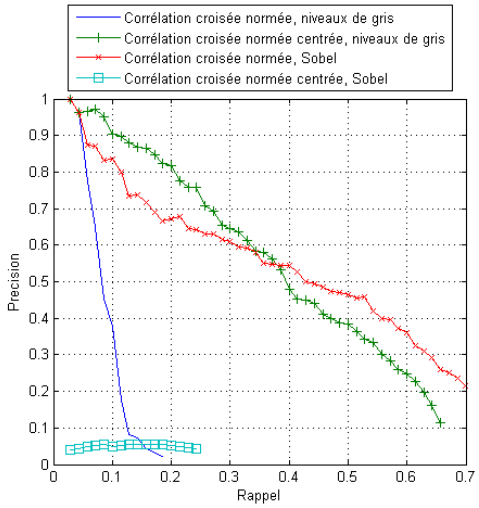
(b) 10 exemples, base (a)+(b)



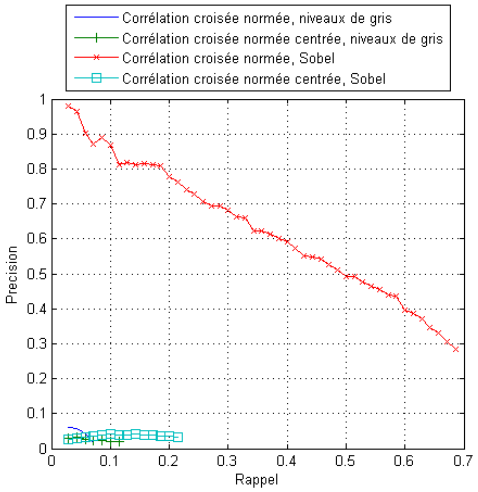
(c) 10 exemples, base (a)+(d)



(d) 20 exemples, base (i ... l)

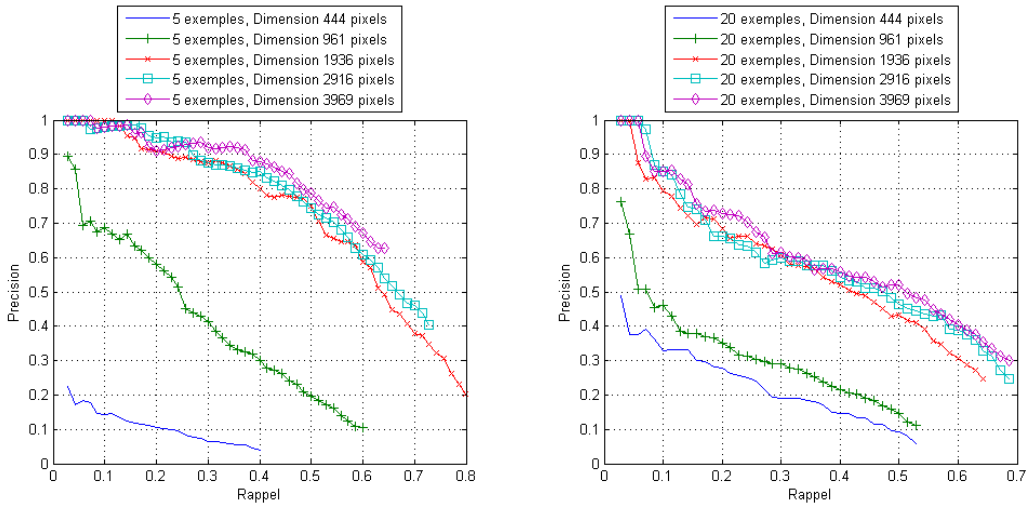


(e) 40 exemples, base (i ... p)



(f) 80 exemples, base (a ... p)

FIGURE 3.11 – Courbes Rappel Précision du système de détection par corrélation croisée. Comparaison entre l'utilisation des images en Niveaux de Gris et des images de contours. La corrélation croisée normée des contours des images semble donner les meilleurs résultats.



(a) (b)

FIGURE 3.12 – Courbes Rappel Précision du système de détection par corrélation croisée normée des contours des images avec 5 puis 20 images exemples. Influence de la dimension des exemples. On remarque une amélioration des taux de détection avec l'augmentation des dimensions des images exemples puis une stagnation lorsque l'aire des images de référence atteint 2000 pixels.

3.5 Association de la corrélation des contours et des Niveaux de Gris

L'image en Niveaux de Gris et les contours de cette dernière apportent des informations différentes. Afin d'améliorer les taux de détection du système basé sur la corrélation, nous avons associé les résultats obtenus pour les images en Niveaux de Gris et les images de contours.

3.5.1 Principe de l'association

Afin d'associer les résultats de la corrélation croisée des contours et de l'image en Niveaux de Gris, nous avons mis au point un système fonctionnant en deux phases (figure : 3.13). Une première étape que nous appellerons phase de prédétection et une seconde que nous nommerons phase de décision. Pour chaque image de référence, nous effectuons une prédétection en utilisant la corrélation croisée normée des contours des images. Un objet est prédétecté s'il correspond à un maximum local sur la carte de score et si le score correspondant est supérieur à un seuil s_1. Ainsi, le système de prédétection est équivalent au système décrit dans la section précédente sans l'utilisation de la fonction permettant d'éliminer les détections superposées. Si un objet est prédétecté, le système effectue une corrélation normée centrée entre l'image de référence et l'image en Niveaux de Gris de l'objet prédétecté. Le résultat de cette corrélation correspondra alors au score s de détection. La même opération est effectuée pour l'ensemble des images de référence. Enfin, seul sont gardées les détections non superposées avec une autre détection ayant obtenu un score s supérieur.

Un tel système présente l'avantage de ne consommer que peu de ressources supplémentaires par rapport au système présenté dans les sections précédentes. En effet, la première phase de prédétection effectue, comme pour le système précédent une corrélation normée sur l'ensemble des positions possibles de l'objet à détecter dans l'image. Cependant, la seconde phase n'effectue une corrélation que pour les objets prédétectés, ce qui ne constitue qu'une infime partie de l'ensemble des positions possibles. Les temps de calculs de la phase de décisions sont ainsi négligeables comparés à l'étape de prédétection.

Afin de tester ce système, nous avons utilisé les mêmes bases de données que précédemment. Les images de référence utilisées ont une aire de 1936 pixels, aussi bien pour la phase de prédétection que pour la phase de décision. La première question que l'on peut se poser est la valeur du seuil de prédétection s_1 que nous devons utiliser. Une valeur faible entraîne un grand nombre de prédétections et risque d'augmenter le nombre de fausses détections mais permettra de minimiser les chances de ne pas détecter un objet. Une valeur élevé de s_1 minimisera le nombre de prédétections diminuant ainsi les chances de fausses détections mais augmentera la proportions d'objets non détectés. Comme on peut le voir sur la figure 3.14, le Rappel du système de prédétection est très sensible à la valeur du seuil s_1. Une valeur de 0.70 permet de prédétecter plus de 95% des objets, mais pour une probabilité de bonne détection de seulement 1%. Une valeur de 0.77 permet d'obtenir une Précision de 1, mais ne peut détecter qu'un objet sur dix. Dans la mesure où la première phase n'est qu'une simple

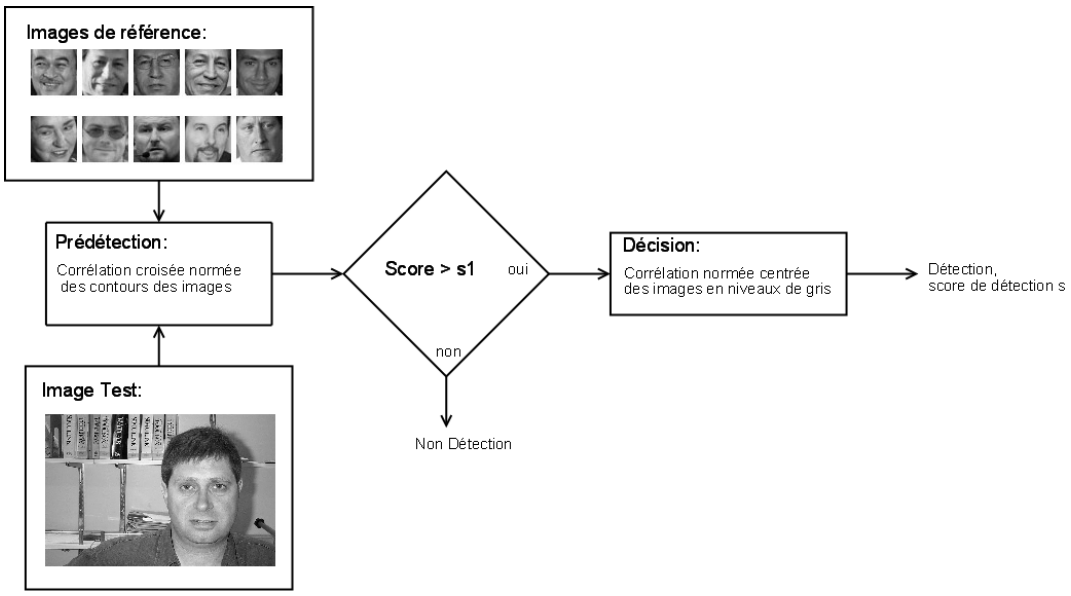


FIGURE 3.13 – Système de détection par association de la corrélation des contours et des images en Niveaux de Gris. Le système de détection par corrélation croisée normée des contours est utilisé comme système de prédétection. Les prédétections associées à un score supérieur à un seuil s_1 sont gardées et vérifiées grâce au système de décision basé sur la corrélation croisée normée centrée des images en Niveaux de Gris.

phase de prédétection, nous devons favoriser le Rappel par rapport à la Précision.

De plus, un seuil de 0.70, qui entraîne une Précision très faible pour le système de prédétection permet de ne tester qu'une centaine de positions possibles de l'objet dans la phase suivante au lieu des 327150 positions possibles au départ, minimisant ainsi grandement les possibilités de fausses détections.

Nous avons testé notre système pour quatre seuils différents (figure : 3.15). Une première fois avec une base d'exemples qui donnait de bons résultats $i.e$, la base (a). Une seconde fois avec une base d'exemples bien moins adaptée, entraînant des taux de détection très faible, $i.e$, la base (b). Une valeur du seuil s_1 de 0.72 semble donner le meilleurs compromis entre la Précision et le Rappel.

On remarque que notre système combinant deux corrélations améliore nettement les performances du système de détection (figure : 3.16), particulièrement lorsque le Rappel se rapproche de 1. Cette méthode permet, avec très peu d'exemples, d'atteindre des taux de détection permettant une utilisation pratique d'un tel système. On remarque en particulier qu'une base de seulement cinq exemples permet de détecter plus de 80% des 450 visages pour un total de seulement 32 fausses détections, c'est a dire, moins d'une fausse détection toutes les dix images. Ce système reste malgré tout très sensible à la base d'exemples utilisée. En particulier, on remarque que la base de 80 exemples obtient des résultats inférieurs aux bases de 5, 10 ou 20 exemples. Cela est dû à l'utilisation des images en Niveaux de Gris pour la phase de

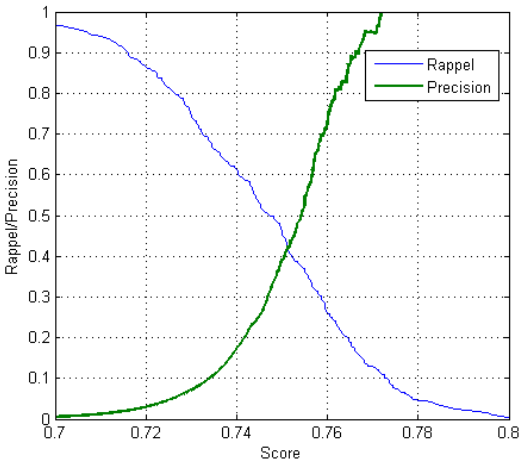
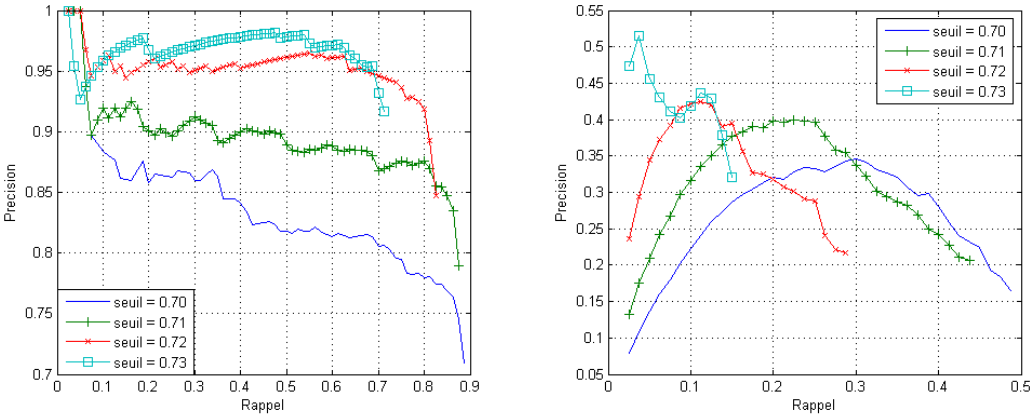


FIGURE 3.14 – Influence du seuil de prédétection sur la Précision et le Rappel du système de prédétection avec une base de 5 exemples (base (a)).

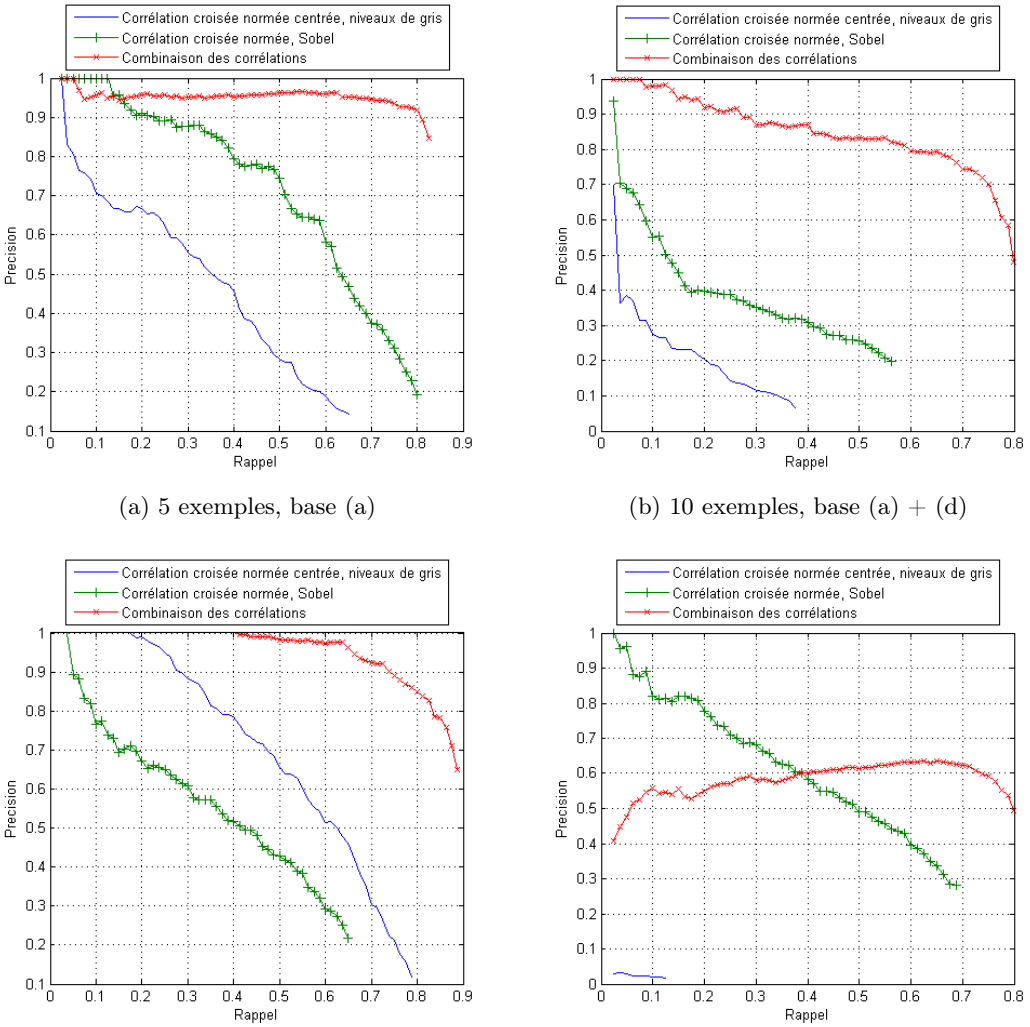


(a) 5 exemples, base (a)

(b) 5 exemples, base (b)

FIGURE 3.15 – Courbes Rappel Précision du système de détection par corrélation croisée normée des contours puis par corrélation normée centrée des images en Niveaux de Gris. Influence du seuil de prédétection.

décision qui pour cette base de 80 exemples donne des résultats très inférieurs à ceux obtenus avec les contours des images. Ainsi, de par l'utilisation des images en Niveaux de Gris, ce système reste très sensible aux variations de luminosité des images. Afin de minimiser ce problème, nous proposons dans la section suivante d'appliquer des traitements d'images permettant de minimiser les effets des variations d'éclairage sur les images en Niveaux de Gris



(a) 5 exemples, base (a)

(b) 10 exemples, base (a) + (d)

(c) 20 exemples, base (i ... l)

(d) 80 exemples, base (a ... p)

FIGURE 3.16 – Courbes Rappel Précision du système de détection par corrélation croisée normée des contours, par corrélation normée centrée des images en Niveaux de Gris, puis en combinant ces deux méthodes. Nous constatons une nette amélioration des résultats, particulièrement pour les valeurs du Rappel proches de 1.

3.5.2 Correction des variations d'illumination

La phase de décision n'est utilisée dans ce système que pour un nombre très limité de prédétections. il devient alors possible, sans allonger significativement les temps de calculs, d'utiliser des traitements d'images complexes avant d'effectuer la corrélation des Niveaux de Gris. Nous retrouvons dans la littérature deux traitements d'images très souvent utilisés pour minimiser les variations d'illumination [85, 47, 131, 121]. Le premier est la correction du gradient d'illumination par la soustraction de la fonction linéaire la mieux adaptée à l'image en Niveaux de Gris. Le second est l'égalisation d'histogramme afin de corriger les variations d'intensité de l'éclairage.

3.5.2.1 Correction du gradient d'illumination

La correction du gradient d'illumination s'effectue en soustrayant la fonction linéaire la mieux adaptée à l'image en Niveaux de Gris. Cette fonction est calculée de façon à minimiser la distance $L2$ entre l'image à traiter et la fonction linéaire bidimensionnelle. Une telle fonction s'écrit sous la forme :

$$f(x,y) = a\,x + b\,y + c \tag{3.11}$$

$$= \mathbf{p}\cdot\mathbf{v} \tag{3.12}$$

$$\mathbf{p} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.13}$$

Déterminer la fonction linéaire la mieux adaptée correspond donc pour une image $I(x,y)$ de dimension $h \times l$ à calculer les paramètres $\mathbf{p}$ tels que l'erreur quadratique E soit minimum, *i.e*, trouver $\mathbf{p}$ tel que $\nabla_{\mathbf{p}} E = 0$.

$$E = \sum_{x,y} (f - I)^2 \tag{3.14}$$

$$\nabla_{\mathbf{p}} E = \underbrace{\left[\sum_{x,y} \mathbf{v}\mathbf{v}^T\right]}_{A} \mathbf{p} - \underbrace{\sum_{x,y} (I\mathbf{v})}_{\mathbf{s}} = 0 \tag{3.15}$$

$$\mathbf{p} = A^{-1}\mathbf{s} \tag{3.16}$$

Les images de références ayant toutes la même dimension $(h \times l)$ et la matrice A ne dépendant que des dimensions des images à traiter. A^{-1} peut être calculé une seule fois pour l'ensemble des images. Afin d'illustrer les résultats de ce traitement d'image, nous avons appliqué cette méthode à des images de visages (figure : 3.17).

3.5.2.2 Egalisation d'histogramme

Cette méthode permet d'augmenter le contraste des images en modifiant la distribution des Niveaux de Gris de manière à ce que l'histogramme des Niveaux de

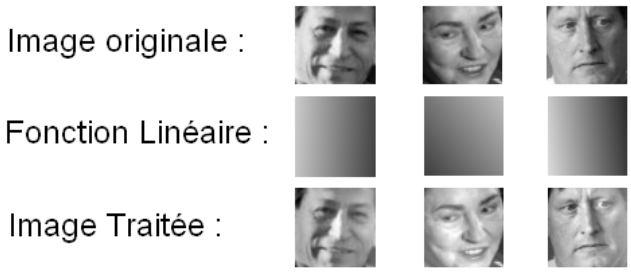


FIGURE 3.17 – Exemple d'application de la correction du gradient d'illumination sur trois images de visages. Nous pouvons constater que cette méthode permet de minimiser les effets dues aux éclairages latéraux.

Gris $H_I(g)$ corresponde à une distribution uniforme. Pour ce faire, on définit l'histogramme cumulatif de l'image I de dimension $n = h \times l$ pixels comme la fonction C_I définie sur l'intervalle de Niveaux de Gris $\{0, ..., M\}$, associant à tout niveau de gris g, le nombre de points ayant un niveau de gris inférieur ou égal à g dans l'image I. Obtenir une répartition uniforme des Niveaux de Gris dans l'image revient à associer à chaque niveau de gris g, le niveau de gris défini par la fonction de rehaussement $f(g)$ tel que $C_I(f(g))$ soit une fonction linéaire. Ceci peut être réalisé en définissant la fonction $f(g)$ comme suit :

$$f(g) = \frac{M}{2n} \left[C_I(g) + C_I(g-1) \right] \tag{3.17}$$

L'effet de cette fonction de rehaussement est d'espacer chaque Niveaux de Gris d'un écart proportionnel au nombre de points ayant ce niveau de gris dans l'image I. Ainsi, tout niveau de gris fortement représenté dans l'image se verra écarté des Niveaux de Gris adjacents. En pratique, ce traitement aura tendance à équilibrer la proportion de zones claires, moyennes et sombres dans l'image. La figure 3.18 montre l'application de cette méthode à un visage et un paysage. On constate sur ces images que l'égalisation d'histogramme permet une meilleure visualisation des détails.

3.5.2.3 Résultats expérimentaux

Afin de vérifier l'intérêt de traiter les images en Niveaux de Gris lors de la phase de décision, nous avons comparé le système de détection sans traitement d'image, avec l'utilisation de la correction du gradient d'illumination, puis l'égalisation d'histogramme et enfin la combinaison des deux traitements d'image (figure : 3.19). On remarque que les deux traitements d'image ont généralement tendance à améliorer les taux de détection. Cependant, cette amélioration est très variable en fonction de la base d'exemples utilisée. Il peut notamment arriver qu'un des deux traitements d'image entraîne une diminution des taux de détection (figure : 3.19b et 3.19d). Cependant, la combinaison des deux traitements d'image entraîne systématiquement une nette amélioration des résultats et permet, avec une base de seulement cinq

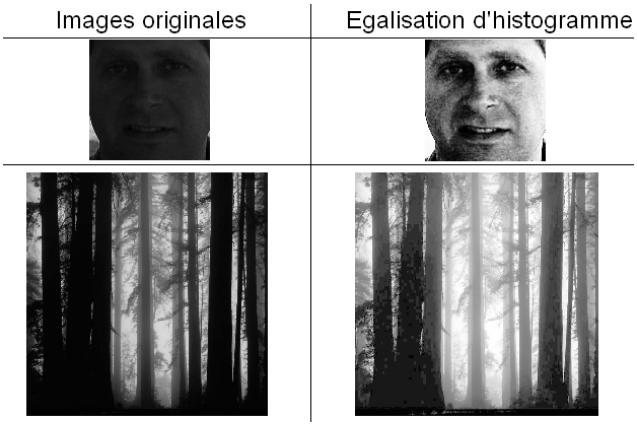
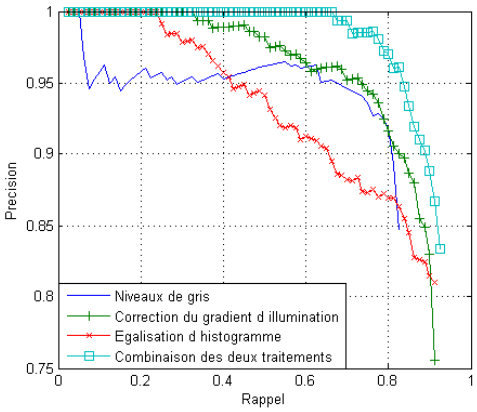
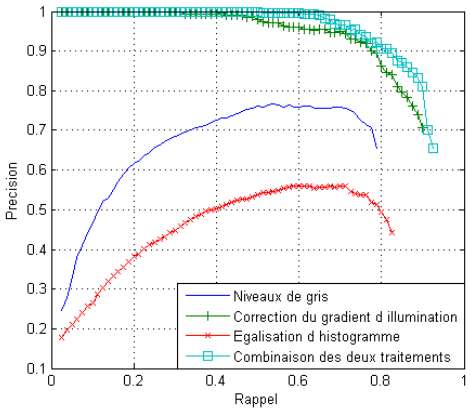


FIGURE 3.18 – Exemple d'application de l'égalisation d'histogramme sur un visage et une image de paysage. Nous pouvons constater que ce traitement permet une amélioration du contraste ainsi qu'une nette amélioration de la visualisation des détails des images.

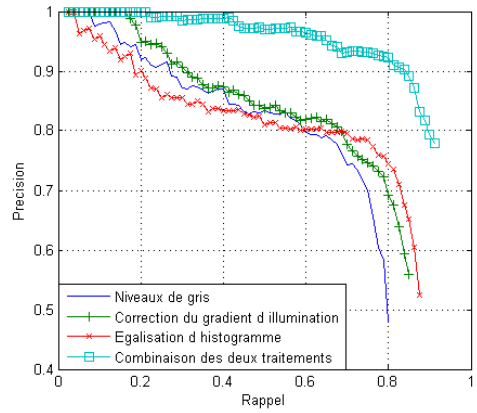
exemples, de détecter près de 90% des 450 visages pour seulement 45 fausses détections (figure : 3.19a). Nous pouvons aussi noter que la correction du gradient d'illumination se montre d'autant plus efficace que le système sans traitement d'image donne de mauvais résultats (figure : 3.19b, 3.19f). Ainsi, cette méthode de correction d'illumination permet de rendre le système de détection par corrélation beaucoup moins sensible à la base d'exemples.



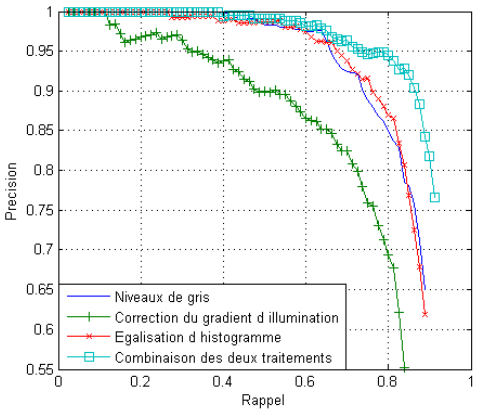
(a) 5 exemples, base (a)



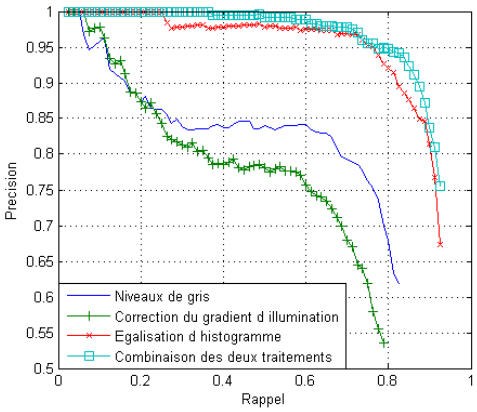
(b) 10 exemples, base (a)+(b)



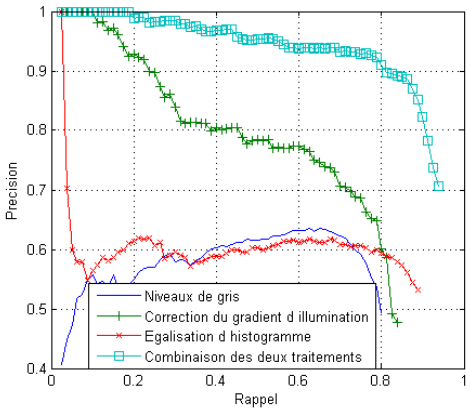
(c) 10 exemples, base (a)+(d)



(d) 20 exemples, base (i ... l)



(e) 40 exemples, base (i ... p)



(f) 80 exemples, base (a ... p)

FIGURE 3.19 – Courbes Rappel Précision du système de détection par corrélation croisée. Comparaison des résultats pour différents traitements de l'image en Niveaux de Gris. La combinaison de la correction du gradient d'illumination et de l'égalisation d'histogramme donne systématiquement les meilleurs résultats.

3.5.3 Correction des variations de forme par la méthode de déformation affine

Nous avons constaté dans la section précédente l'utilité des méthodes de correction des variations d'illumination appliquées au système de décision. Le score donné par le système de décision étant basé sur la corrélation, il est aussi sensible aux variations de forme que peut subir un objet complexe tel qu'un visage et notamment, à la rotation ou aux variations d'échelle. Afin de minimiser les effets des variations de forme sur le score du système de décision et donc sur les taux de détection, nous avons introduit une méthode permettant de déformer les images des 'objet' prédétectés de façons à maximiser la correspondance entre ces derniers et l'image de référence ayant conduit à la prédétection. Ainsi, le système de décision applique une déformation affine à l'image de l'objet prédétecté afin de corriger les variations de forme entre les deux images et de maximiser le score de la corrélation normée centrée (figure : 3.20). Par la suite, nous parlerons généralement de déformation de l'image de test par rapport à l'image de référence car, dans notre algorithme, la déformation est appliquée en modifiant la fonction bidimensionnelle représentative de l'image test. Cependant, il n'y a mathématiquement aucune différence entre déformer l'image de référence par rapport à l'image test ou l'inverse, ceci ne dépendant que du référentiel dans lequel nous nous plaçons.

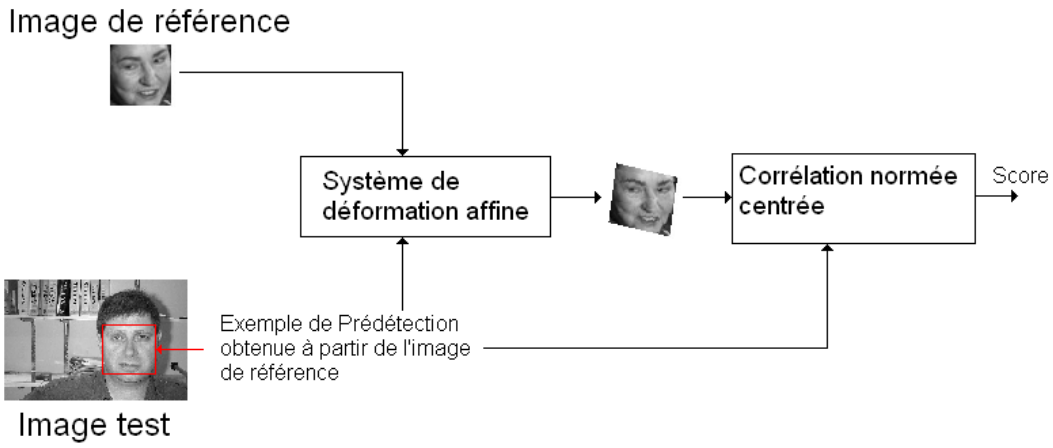


FIGURE 3.20 – Utilisation de la méthode de déformation affine dans la phase de décision du système de détection. Lorsqu'une image de référence conduit à une prédétection, on applique la méthode de déformation affine afin d'appareiller au mieux l'image de référence et l'image de test.

La méthode de compensation de mouvement par déformation affine consiste à déformer une image de test selon, le plus généralement, six paramètres afin de minimiser la distance entre cette dernière et une image de référence. Elle est le plus souvent utilisée dans l'estimation de mouvements pour des séquences d'images [32]. Le but de la compensation de mouvement est d'identifier les mouvements d'objets dans une scène ou plus précisément, la projection de ce mouvement dans le plan de l'image.

On retrouve de très nombreuses applications comme pour les systèmes de caméras stéréoscopiques [37], la compression vidéo [88], ou encore l'interpolation temporelle d'images [44, 120]. La déformation affine constitue une des méthodes d'estimation de déformation les plus simples puisque seul six paramètres sont nécessaires pour caractériser la déformation. Cependant, nous allons voir que cette méthode permet de compenser des déformations importantes appliquées à des visages.

3.5.3.1 Principe de la déformation affine

La déformation affine est l'approximation au premier ordre de la déformation subie par l'image d'un objet plan rigide soumis à un déplacement et une rotation. Ainsi, la déformation affine consiste à translater, incliner et modifier l'échelle d'une image test G afin de maximiser la correspondance avec l'image de référence F.

Si nous notons $G^* = \{g^*(\mathbf{r})\}$ le résultat de la déformation affine d'une image $G = \{g(\mathbf{r})\}$, nous pouvons écrire :

$$g^*(\mathbf{r}) = g(\mathbf{r} + \mathbf{d_r}) \tag{3.18}$$

$$\mathbf{r} = \begin{pmatrix} u \\ v \end{pmatrix} \tag{3.19}$$

$$\mathbf{d_r} = \begin{pmatrix} d_u \\ d_v \end{pmatrix} = \begin{pmatrix} a_0 u + a_1 v + a_2 \\ a_3 u + a_4 v + a_5 \end{pmatrix} \tag{3.20}$$

Les six paramètres $(a_0, ..., a_5)$ définissent la déformation affine. a_2 et a_5 correspondent à la translation, a_0, a_1, a_2 et a_3 déterminent l'inclinaison et l'échelle horizontale et verticale de l'image. Ainsi, déterminer la déformation affine qui maximise la correspondance entre les image G^* et F revient à trouver les paramètres $(a_0, ..., a_5)$ qui maximisent le critère de similarité Ψ.

3.5.3.2 Maximisation de la corrélation normée centrée

Le critère le plus couramment utilisé pour déterminer la meilleure déformation affine est la minimisation de la distance L2 entre les images que nous souhaitons appareiller. Afin de maximiser la robustesse du système de déformation aux variations d'illumination et d'être plus cohérent avec la mesure de similarité que nous utilisons déjà, nous avons utilisé la corrélation normée centrée comme critère de similarité. Ainsi, nous cherchons les paramètres $(a_0, ..., a_5)$ qui maximisent la fonction Ψ :

$$\Psi = \sum_{\mathbf{r} \in F} \overbrace{\left(\frac{f(\mathbf{r}) - m_f}{\sigma_f} \right)}^{f_n} \underbrace{\left(\frac{g(\mathbf{p} + \mathbf{r} + \mathbf{d_r}) - m_g}{\sigma_g} \right)}_{g_n} \tag{3.21}$$

$F = \{f(\mathbf{r})\}$ et $G = \{g(\mathbf{r})\}$ sont respectivement l'image de référence de dimension $n = l \times h$ et l'image de test, $\mathbf{p}$ les coordonnées initiales de G ou l'objet a été prédétecté.

m_f et m_g sont les moyennes des fonctions $f(\mathbf{r})$ et $g^*(\mathbf{p} + \mathbf{r})$, $\mathbf{r} \in F$:

$$m_f = \frac{1}{n} \sum_{\mathbf{r} \in F} f(\mathbf{r}) \tag{3.22}$$

$$m_g = \frac{1}{n} \sum_{\mathbf{r} \in F} g(\mathbf{p} + \mathbf{r} + \mathbf{d_r}) \tag{3.23}$$

σ_f et σ_g représentent l'écart type des fonctions $f(\mathbf{r})$ et $g^*(\mathbf{p} + \mathbf{r})$, $\mathbf{r} \in F$:

$$\sigma_f = \sqrt{\sum_{\mathbf{r} \in F} \left(f(\mathbf{r}) - m_f\right)^2} \tag{3.24}$$

$$\sigma_g = \sqrt{\sum_{\mathbf{r} \in F} \left(g(\mathbf{p} + \mathbf{r} + \mathbf{d_r}) - m_g\right)^2} \tag{3.25}$$

Maximiser la fonction Ψ conduit donc à résoudre le système de six équations suivant :

$$\frac{\partial \Psi}{\partial a_i} = 0 \qquad i \in [0, 5] \tag{3.26}$$

Ce système d'équation ne peut pas être résolu analytiquement, cependant il existe différentes méthodes permettant d'optimiser la fonction Ψ. Ce type de problèmes est le plus souvent résolu par des méthodes de descente par gradient. Cependant, le problème d'optimisation n'étant que de dimension six, il semble approprié d'utiliser une méthode d'optimisation non linéaire. Dans [32] Dugelay et Sanson montrent que la méthode d'optimisation itérative de Gauss Newton permet une convergence rapide et robuste vers une solution au problème de la déformation affine.

Cette méthode est basée sur deux approximations pour effectuer l'optimisation :

- La fonction Ψ à optimiser est localement une fonction polynomiale du second ordre.
- La dérivée seconde de la fonction g est nulle (La matrice Hessien de $g(\mathbf{r})$, $H_g = \mathbf{0}$). Autrement dit, la variation de la luminance de l'image G est localement linéaire.

On note $A_k = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix}^T$ la valeur des paramètres de la déformation affine à la k^{ieme} itération.

De par l'approximation de la forme localement polynomiale à l'ordre deux de la fonction Ψ, les paramètres de la déformation affine à la $(k+1)^{ieme}$ itération se déterminent par l'équation suivante.

$$A_{k+1} = A_k - H_A^{-1} G_A \tag{3.27}$$

Ou H_A est la matrice Hessien de la fonction Ψ et G_A en est le gradient.

$$G_A = \begin{pmatrix} \frac{\partial \Psi}{\partial a_i} \\ \vdots \end{pmatrix} \quad H_A = \begin{pmatrix} \frac{\partial^2 \Psi}{\partial a_i \partial a_j} & \cdots \\ \vdots & \ddots \end{pmatrix} \tag{3.28}$$

Afin de rendre plus claires les équations, nous utiliserons les notations suivantes :

$$\begin{array}{ll} g_p \Leftrightarrow g(\mathbf{p}+\mathbf{r}+\mathbf{d_r}) & \\ \mathbf{g_r} \Leftrightarrow \nabla_r(g_p) & \mathbf{d_r^i} \Leftrightarrow \frac{\partial \mathbf{d_r}}{\partial a_i} \\ m_g^i \Leftrightarrow \frac{\partial m_g}{\partial a_i} & \sigma_g^i \Leftrightarrow \frac{\partial \sigma_g}{\partial a_i} \end{array}$$

- g_p la valeur de g au point $(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$.
- $\mathbf{g_r}$ la valeur du gradient de g au point $(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$.
- $\mathbf{d_r^i}$ la fonction dérivée de $\mathbf{d_r}$ par rapport au paramètre a_i.
- m_g^i la fonction dérivée par rapport au i^{ieme} paramètre de la moyenne de la fonction $g(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$, $\mathbf{r} \in F$.
- σ_g^i la fonction dérivée de l'écart type de $g(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$, $\mathbf{r} \in F$ par rapport à a_i.

Afin de déterminer A_{k+1}, nous devons calculer à chaque itération les matrices G_A et H_A. L'approximation de linéarité locale de la fonction $g(\mathbf{r})$ permet de déterminer G_A et H_A simplement à partir des fonctions $f(\mathbf{r})$ et $g(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$ ainsi que le gradient de $g(\mathbf{r})$ numériquement calculable par des méthodes d'approximation bilinéaires. G_A se détermine ainsi :

$$\begin{aligned} \frac{\partial \Psi}{\partial a_i} &= \sum_{\mathbf{r} \in F} \left(\frac{f(\mathbf{r}) - m_f}{\sigma_f} \right) \frac{\partial}{\partial a_i} \left(\frac{g_p - m_g}{\sigma_g} \right) \\ &= \sum_{\mathbf{r} \in F} f_n \frac{\partial g_n}{\partial a_i} \\ &= \sum_{\mathbf{r} \in F} f_n \frac{\left({\mathbf{d_r^i}}^T \mathbf{g_r} - m_g^i \right) \sigma_g - (g_p - m_g)\, \sigma_g^i}{\sigma_g^2} \end{aligned} \tag{3.29}$$

Avec :

$$m_g^i = \frac{1}{n} \sum_{\mathbf{r} \in F} {\mathbf{d_r^i}}^T \mathbf{g_r} \tag{3.30}$$

Si on note $V_g^i = \frac{\partial V_g}{\partial a_i}$ la dérivée de la variance V_g de $g(\mathbf{p}+\mathbf{r}+\mathbf{d_r})$:

$$\begin{aligned} \sigma_g^2 = \quad V_g &= \sum_{\mathbf{r} \in F} (g_p - m_g)^2 \\ V_g^i &= \sum_{\mathbf{r} \in F} 2 \left({\mathbf{d_r^i}}^T \mathbf{g_r} - m_g^i \right) (g_p - m_g) \\ \sigma_g^i &= \frac{V_g^i}{2 V_g^{\frac{1}{2}}} \end{aligned} \tag{3.31}$$

De même, si on remarque que $\forall(i,j)$, $\frac{\partial \mathbf{d_r^i}}{\partial a_j} = \mathbf{0}$, La matrice Hessien H_A est déterminée comme suit :

$$\frac{\partial^2 \Psi}{\partial a_i \partial a_j} = \sum_{\mathbf{r}\in F} f_n \frac{\partial^2 g_n}{\partial a_i \partial a_j} \tag{3.32}$$

$$\begin{aligned}\sigma_g^4 \frac{\partial^2 g_n}{\partial a_i \partial a_j} = & \\ & 2\left(g_p - m_g\right)\sigma_g \sigma_g^i \sigma_g^j - \left(\mathbf{d_r^j}^T \mathbf{g_r} - m_g^j\right)\sigma_g^i \sigma_g^2 - \\ & \left(g_p - m_g\right)\sigma_g^{ij}\sigma_g^2 - \left(\mathbf{d_r^i}^T \mathbf{g_r} - m_g^i\right)\sigma_g^j \sigma_g^2\end{aligned} \tag{3.33}$$

$\sigma_g^{ij} = \frac{\partial^2 \sigma_g}{\partial a_i \partial a_j}$ étant la dérivée seconde de σ_g par rapport aux paramètres a_i et a_j.

$$\begin{aligned}\sigma_g^{ij} &= \frac{\partial}{\partial a_j}\frac{1}{2}\left(V_g^i V_g^{\frac{-1}{2}}\right) \\ &= \frac{1}{2V_g}\left(V_g^{ij}\sigma_g - \frac{V_g^i V_g^j}{2\sigma_g}\right)\end{aligned} \tag{3.34}$$

Avec $V_g^{ij} = \frac{\partial V_g^i}{\partial a_j}$ La dérivée seconde de la variance V_g :

$$V_g^{ij} = \sum_{\mathbf{r}\in F} 2\left(\mathbf{d_r^i}^T \mathbf{g_r} - m_g^i\right)\left(\mathbf{d_r^j}^T \mathbf{g_r} - m_g^j\right) \tag{3.35}$$

Une fois les matrices G_A et H_A déterminées pour la k^{ieme} itération, il suffit d'inverser la matrice H_A et d'utiliser l'équation 3.27 afin de déterminer une nouvelle valeur des paramètres de la déformation affine. Nous considérons qu'il y a convergence si la distance $L2$ entre les paramètres calculés pour deux itérations successives est inférieure à un seuil $d = 10^{-5}$, le nombre maximum d'itérations du système étant fixé à 40 afin d'éviter des temps de calculs prohibitifs et des solutions trop éloignées des conditions initiales. Autrement dit, la solution à la détermination des paramètres optimums de la déformation affine est donnée par A^k, la valeur des six paramètres à la k^{ieme} itération, avec $k \leq 40$ et $\left\|A^k - A^{k-1}\right\| < d$.

3.5.3.3 Résultats expérimentaux sur la corrélation d'images de visages

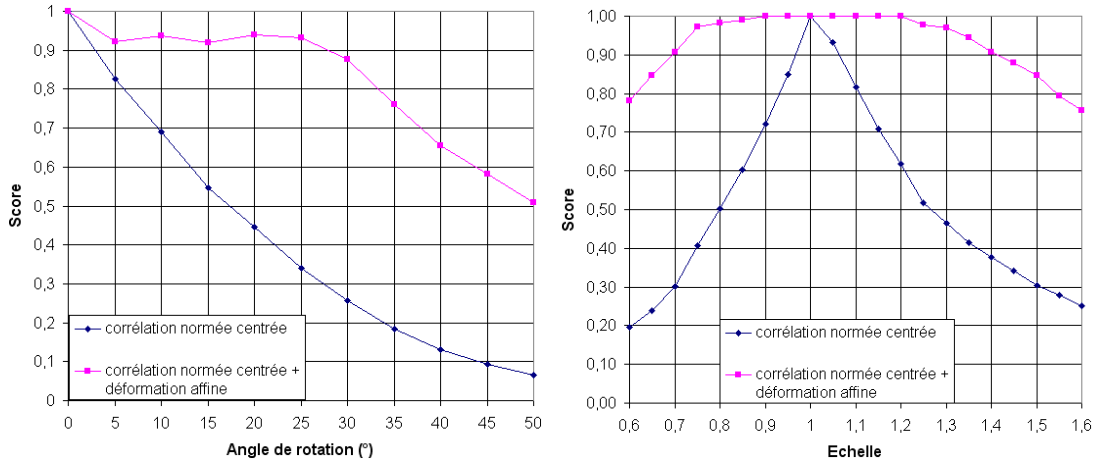
Dans cette section, nous présentons deux expérimentations distinctes. La première partie consiste à mesurer la capacité de la méthode de déformation affine à compenser les déformations et à améliorer la robustesse de notre mesure de similarité basée sur la corrélation normée centrée. La seconde partie aura pour but de mesurer l'efficacité d'une mesure de similarité basée sur la déformation affine et la corrélation normée centrée pour notre système de détection.

Déformation affine appliquée à la compensation de déformations : Afin de mesurer l'efficacité de la déformation affine pour améliorer la robustesse de la corrélation normée centrée aux rotations et aux changements d'échelles, nous avons commencé par comparer le score de corrélation normée centrée avec et sans l'utilisation de la méthode de déformation affine. Pour ce faire, nous avons utilisé 80 images contenant chacune un visage de dimension 44×44 soit environ 2000 pixels. Pour chaque image, nous en avons extrait le visage et appliqué une rotation ou un changement d'échelle afin d'obtenir une image de référence F que nous comparons à l'image originale G. Ainsi, plus la méthode de déformation affine sera capable de compenser les déformations dues à la rotation ou au changement d'échelle, plus le score de la mesure de similarité associée se rapprochera de 1. Nous avons reporté sur la figure 3.21 le score moyen de la corrélation normée centrée en fonction des déformations (rotation ou changement d'échelle) avec et sans l'utilisation de la méthode de déformation affine. La première constatation que nous pouvons faire est que la corrélation normée centrée est assez sensible aux déformations puisqu'une rotation de 10° ou une variation de 10% de l'échelle entraîne une chute du score moyen de 30%. La seconde constatation que nous faisons est que la méthode de déformation affine permet de compenser efficacement les déformations que nous avons appliquées. Ainsi, le système de déformation affine est capable d'appareiller des images de visages ayant subi des rotations de plus de 40° ou de fortes variations d'échelle.

Si les déformations dues aux variations d'échelle relativement faibles sont parfaitement compensées par la méthode de déformation affine, ce n'est pas le cas de la rotation qui n'obtient pas un score moyen de corrélation de 1 après compensation même pour des rotations faibles. Ceci s'explique par les interpolations effectuées lors du calcul des rotations des images. Cependant, on constate que la déformation affine permet tout de même de bien corriger cette déformation puisque nous obtenons un score moyen de 0.9 pour une rotation des visages de 30° (une correction parfaite donnant un score de corrélation de 1).

Enfin, afin de mesurer l'efficacité de l'algorithme d'optimisation de Newton Gauss appliqué à l'optimisation de la corrélation normée centrée, nous avons mesuré le nombre moyen d'itérations nécessaire à notre méthode pour converger vers une solution (figure : 3.22). Si pour de fortes déformations le nombre moyen d'itérations de l'algorithme tend vers les 40 itérations qui correspondent au nombre maximum autorisées, nous pouvons constater que pour des déformations modérées, l'algorithme converge en seulement une dizaine d'itérations, démontrant ainsi l'efficacité de cette méthode. Une approche multirésolution permettrait certainement de corriger des variations d'échelle et des rotations plus importantes, cependant nous n'en avons pas l'utilité pour notre système de détection.

Application au système de détection : Nous avons pu constater que la méthode de déformation affine permet de compenser efficacement les déformations simples appliquées à des visages. Cependant, cela ne permet pas d'assurer l'efficacité d'une mesure de similarité basée sur la déformation affine et la corrélation normée centrée dans le cadre d'un système de détection. En effet, si la compensation de mouvement

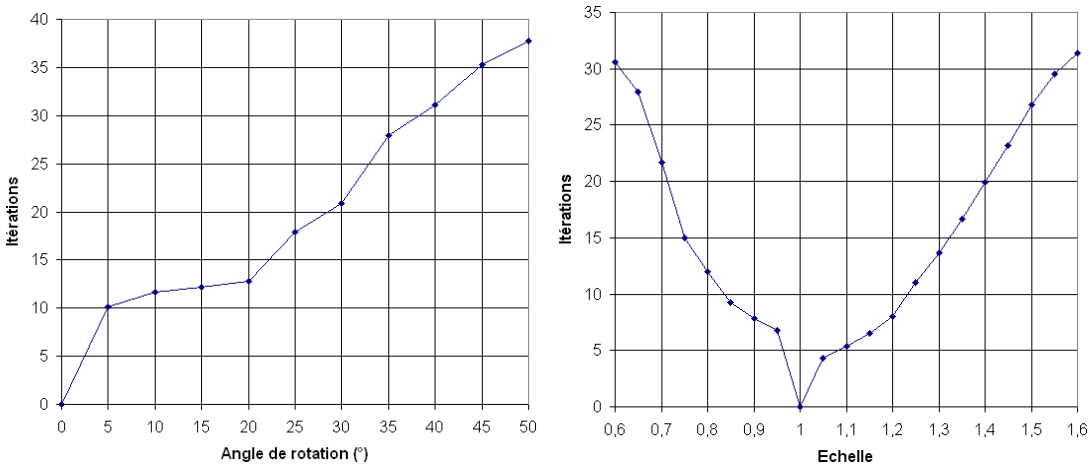


(a) (b)

FIGURE 3.21 – Score moyen sur 80 visages de la corrélation normée centrée avec et sans l'application de la méthode de déformation affine pour des visages ayant subi une rotation ou un changement d'échelle. On constate que la corrélation normée centrée est très sensible à la rotation ou aux variations d'échelle, cependant le système de déformation affine permet de compenser ces déformations et d'obtenir une mesure de similarité presque insensible aux variations modérées d'échelle ou d'angle d'inclinaison des visages.

par déformation affine permet de maximiser le score de corrélation entre deux visages en compensant les variations de forme entre ces derniers, elle permet aussi de maximiser le score de corrélation entre un visage de référence et un objet qui ne serait pas un visage. Autrement dit, la méthode de déformation peut entraîner un score de détection plus important pour les fausses alarmes et provoquer une baisse des performances du système de détection. Afin de tester l'utilité d'un tel système, nous avons appliqué le système de détection à différentes bases de référence. Nous avons comparé les résultats obtenus avec et sans l'utilisation de la déformation affine (figure : 3.23). Nous constatons que l'efficacité du système de déformation affine est relativement dépendante de la base de détection utilisée, si dans certains cas nous constatons une amélioration sensible des résultats (figure : 3.25b,3.23f), le système de déformation peut aussi conduire à une baisse des taux de détection (figure : 3.25b, 3.23d). De façon générale, le système de déformation affine tend à augmenter la Précision au détriment du Rappel. En effet, la déformation affine permet d'augmenter la proportion de détection avec un score de similarité élevé qui garantit une très faible probabilité d'effectuer une fausse détection, ce qui augmente la Précision du système lorsque cette dernière est déjà proche de 1. Cependant, la déformation affine augmente aussi le score de similarité de nombreuses fausses détections, entraînant une baisse de la Précision lorsque cette dernière diminue.

Si nous combinons la méthode de déformation affine avec la correction d'illumination, c'est à dire, que nous appliquons la correction du gradient d'illumination

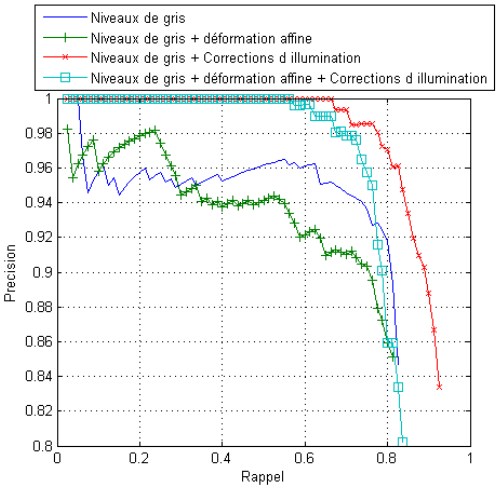


(a) (b)

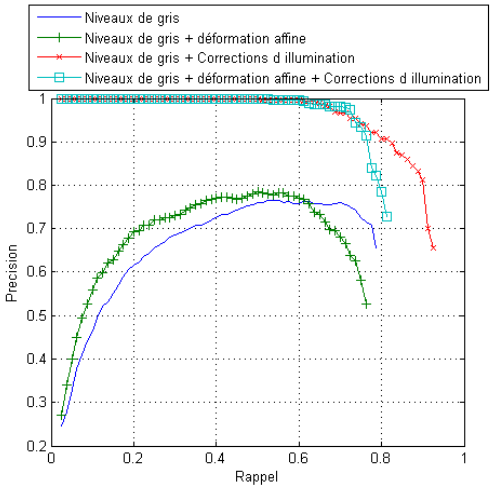
FIGURE 3.22 – Nombre moyen d'itérations nécessaire au système de déformation affine pour converger vers une solution pour 80 visages ayant subi une rotation ou un changement d'échelle. Plus la déformation à compenser est importante plus le nombre d'itérations nécessaire pour obtenir une solution augmente. Nous pouvons cependant remarquer que pour des rotations et des variations d'échelle modérées, le système de déformation affine converge en une dizaine d'itérations.

et l'égalisation d'histogramme aux images déformées avant d'effectuer la corrélation normée centrée donnant le score de décision, nous ne constatons pas une amélioration générale des résultats par rapport au même système sans l'utilisation de la déformation affine. Cependant, on remarque que le système de détection est moins sensible à la base de données utilisée, en particuliers, les meilleurs taux de détection sont atteints pour le système utilisant le maximum d'exemples ce qui n'était pas le cas sans l'utilisation de la méthode déformation affine.

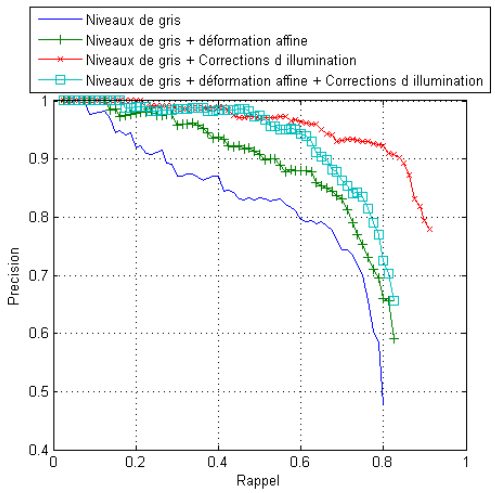
Si la correction des variations d'illumination et, dans une moindre mesure, des déformations affines des images en Niveaux de Gris a permis une amélioration sensible des résultats, nous pouvons constater que le gain de performances le plus important est apporté par l'association des deux mesures de similarité basées sur la corrélation d'images dont nous avons extrait des informations différentes (Contours et images en Niveaux de Gris).



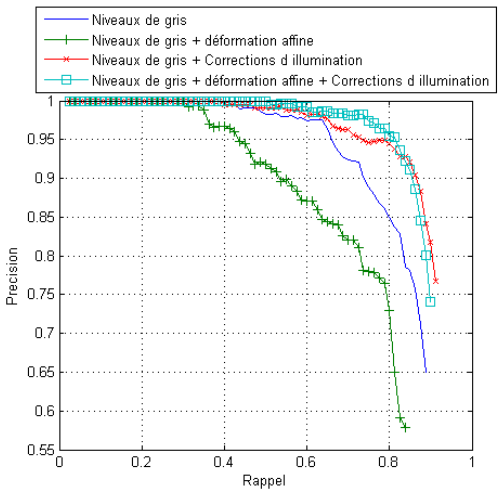
(a) 5 exemples, base (a)



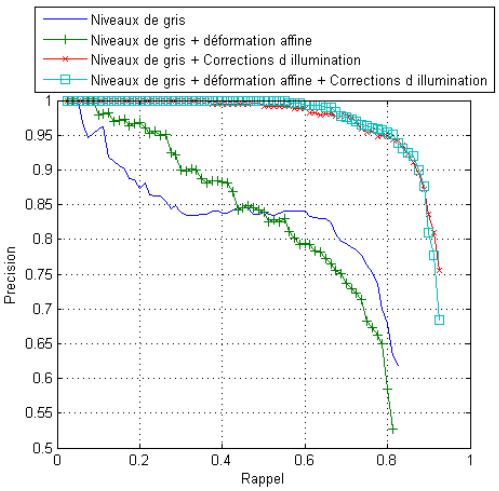
(b) 10 exemples, base (a)+(b)



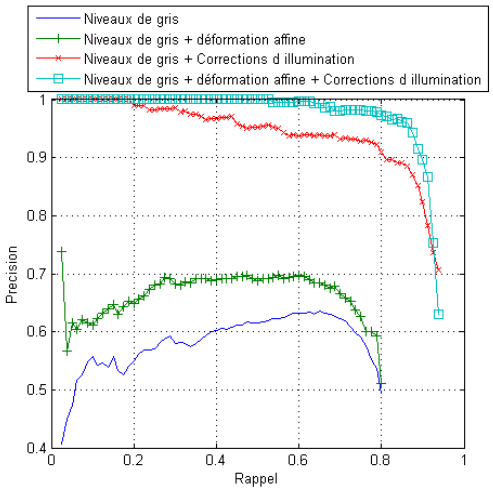
(c) 10 exemples, base (a)+(d)



(d) 20 exemples, base (i ... l)



(e) 40 exemples, base (i ... p)



(f) 80 exemples, base (a ... p)

FIGURE 3.23 – Courbes Rappel Précision du système de détection par corrélation croisée. Comparaison des résultats pour différents traitements des images en Niveaux de Gris, avec et sans l'utilisation de la méthode de déformation affine. On observe que la méthode de déformation affine a tendance à améliorer la Précision au détriment du Rappel.

3.6 Corrélation croisée avec utilisation de filtres de contours orientés

Dans les sections précédentes, nous avons pu constater que si une simple corrélation croisée ne permet pas d'effectuer une détection efficace d'objets complexes, l'utilisation de combinaisons de corrélations des contours et des images en Niveaux de Gris permet une très nette amélioration des résultats. Ainsi, l'utilisation de plusieurs corrélations apportant chacune une information différente permet l'utilisation d'une mesure de similarité aussi simple qu'une corrélation pour détecter des objets aussi complexes que des visages. Dans cette section, nous proposons d'utiliser les filtres de contours orientés afin de combiner non plus seulement une image des contours et l'image en Niveaux de Gris, mais plusieurs images de contours.

3.6.1 Méthode d'association des mesures de similarité

Le système précédent associait deux mesures de similarité (corrélation normée des contours des images et corrélation normée centrée des images en Niveaux de Gris) en divisant le système en deux parties. La première mesure de similarité étant utilisée pour effectuer une prédétection, *i.e*, donner les positions et échelles probables de l'ensemble des objets que nous souhaitons détecter. La seconde mesure de similarité consistant à donner un score nous permettant de décider si la prédétection est correcte. Autrement dit, le système de détection détecte un objet si, à une certaine position et échelle, le score de similarité donné par le système de prédétection est supérieur à un seuil s_1 et le score de similarité donné par le système de décision est supérieur à un seuil final s_f ; la valeur du seuil s_f permettant de choisir un compromis entre le Rappel et la Précision du système de détection. Afin de généraliser cette méthode à un nombre quelconque de mesures de similarité, nous avons modifié le fonctionnement du système de détection comme indiqué sur la figure 3.24. Ainsi, chacune des n mesures de similarité entraîne un score de détection s_i. Le score final du système de détection est alors le score de la mesure de similarité ayant donnée la valeur la plus basse. Autrement dit, le score final du système de détection est donné par la mesure de similarité j telle que, quel que soit i, $s_j \leq s_i$. Ainsi, si l'on fixe le seuil de détection final à une valeur fixe s_f, un objet sera détecté si l'ensemble des mesures de similarité donne un score supérieur au seuil s_f. De cette façon, comme pour le système basé sur une phase de prédétection et une phase de décision, les temps de calculs ne seront que peu impactés par le nombre de mesures utilisées. En effet, un tel système constitue en fait une cascade classifieurs, à l'image du système de détection d'objets de Viola-Jones [131]. La seconde mesure de similarité n'est effectuée que si la première est supérieure au seuil s_f, la troisième si les deux premières sont supérieures à ce seuil et de même jusqu'à la dernière mesure de similarité.

Finalement, comme précédemment, nous éliminons les détections superposées en ne gardant que les détections non superposées à une autre détection avec un score supérieur.

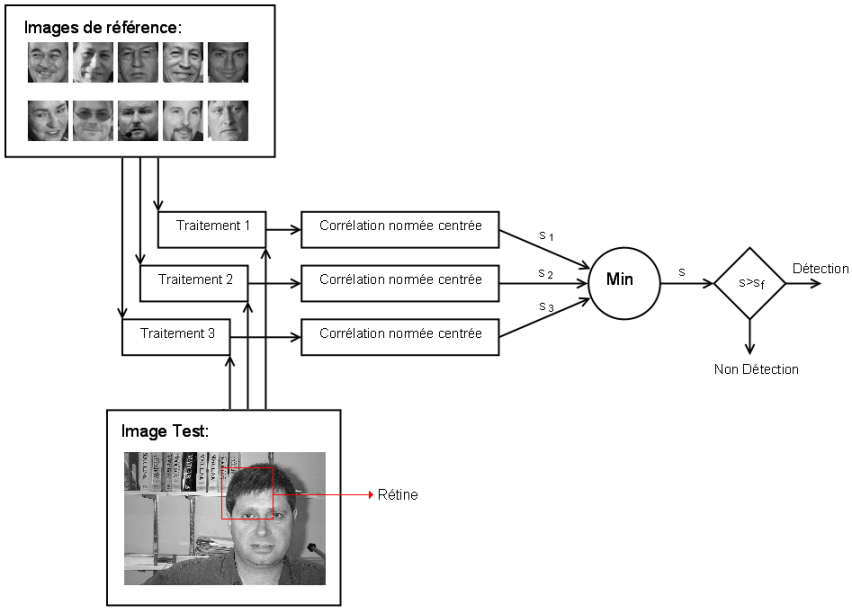


FIGURE 3.24 – Système de détection par association de corrélations (exemple avec trois corrélations). Chaque image de référence et rétines extraites de l'image de test est traitée selon trois méthodes donnant trois images distinctes ; par exemple, l'image en Niveaux de Gris, les contours verticaux et les contours horizontaux. Le traitement d'image associé à la corrélation normée centrée constitue alors une mesure de similarité. Si l'ensemble des scores de similarité s_i est supérieur au seuil s_f, alors nous détectons un objet à l'emplacement et l'échelle correspondant à la rétine. Finalement, nous éliminons les détections superposées par la même méthode que précédemment.

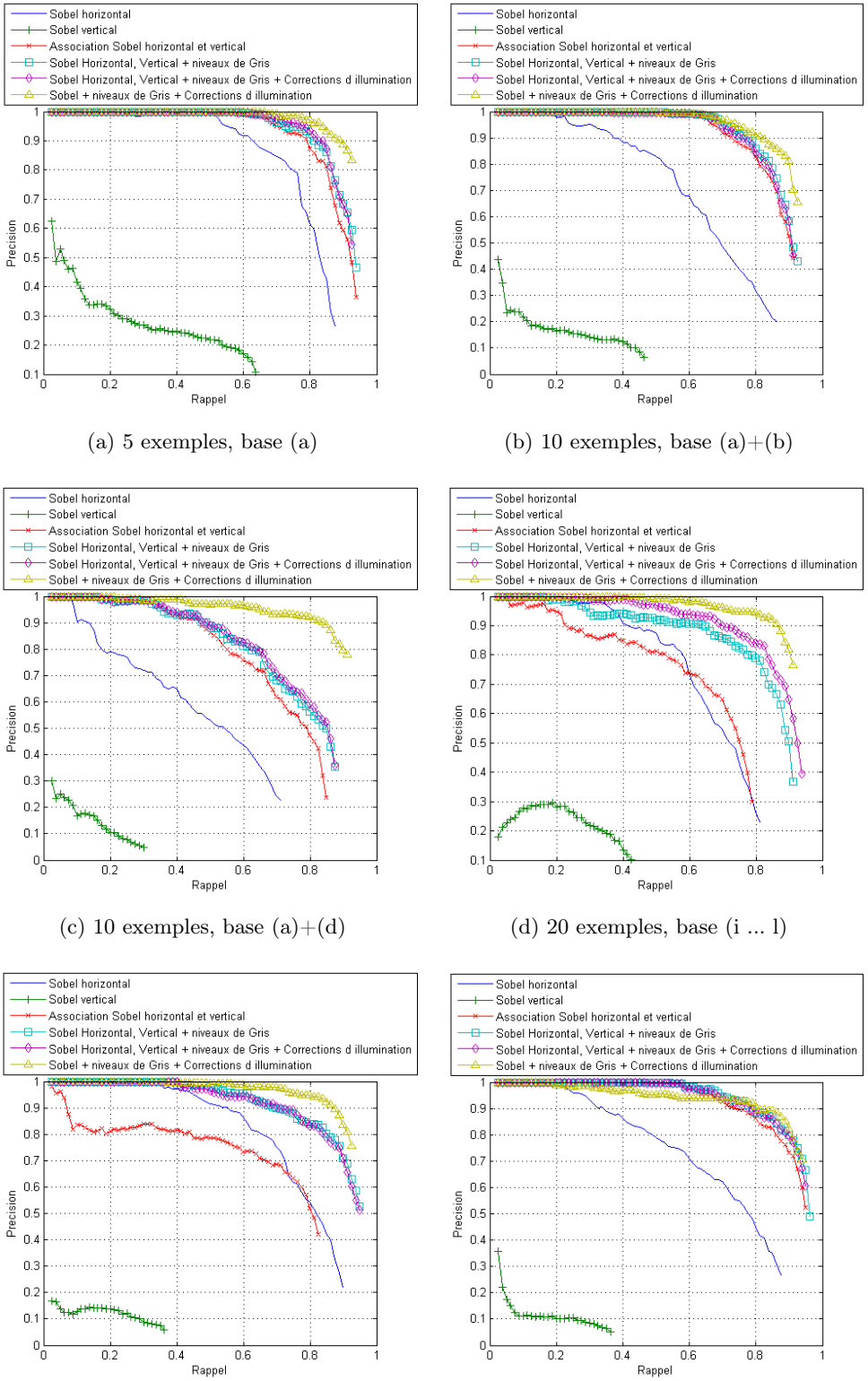
3.6.2 Utilisation des filtres de contours horizontaux et verticaux de Sobel

Nous avons remarqué que la corrélation des contours des images extraits à partir de la méthode de Sobel permet de rendre le système de détection par corrélation plus performant par rapport à l'utilisation des images en Niveaux de Gris. Nous avons ensuite associé la corrélation de ces contours avec la corrélation des images en Niveaux de Gris ce qui nous a permis une nette amélioration du système de détection. Dans cette section nous proposons, non plus d'utiliser les images de contours extraits par la méthode de Sobel, mais les filtres horizontaux et verticaux utilisés par Sobel. Ainsi, le premier et le second traitement du système de détection consisteront à extraire les contours horizontaux et verticaux des images par convolution avec les filtres utilisés par Sobel. Un troisième traitement consistant à utiliser directement l'image en Niveaux de Gris ou l'image en Niveaux de Gris avec correction des variations d'illumination sera aussi évalué. A cause de la sensibilité du système de détection à la base de référence, nous avons, comme pour les expérimentations précédentes, utilisé différentes bases de référence.

La figure 3.25 montre les résultats obtenus sous forme de courbes Rappel Précision. Afin de montrer l'intérêt de l'association des mesures de similarité, nous avons reporté les résultats obtenus avec la seule extraction des contours horizontaux puis verticaux, puis l'association des deux. Nous avons ensuite associé un troisième traitement consistant dans un premiers temps à la simple utilisation de l'image en Niveaux de Gris, puis le traitement permettant de corriger les variations d'illumination que nous avons utilisé pour le système précèdent. Enfin, afin d'avoir un point de comparaison, nous avons fourni les résultats obtenus par le système de détection précédent basée sur la phase de prédétection et la phase de décision avec la correction des variations d'illumination de l'image en Niveaux de Gris. La première constatation que l'on peut faire et sans doute la plus évidente est que, quel que soit la base d'images utilisée l'association des mesures de similarité apporte un gain de performance très important. En effet, l'association de la corrélation normée centrée des contours horizontaux et verticaux entraîne presque systématiquement une nette amélioration des résultats qui s'approchent de ceux obtenus avec le système précédent comportant un traitement complexe de correction d'illumination ainsi que la nécessité de régler un seuil de prédétection. Cependant, nous pouvons constater que si la corrélation des contours horizontaux sans associations entraîne des résultats satisfaisants, ce n'est pas le cas des contours verticaux qui donnent des résultats très variables et souvent bien inférieurs à ceux obtenus par la corrélation des contours horizontaux. Ainsi, dans certains cas ou les résultats obtenus par la corrélations des contours verticaux et horizontaux sont trop différents, alors l'association des deux corrélations peut entraîner une stagnation des résultats (figure : 3.25d), ou même une baisse (figure : 3.25e). Cependant, si l'on combine les corrélations des contours horizontaux et verticaux ainsi que l'image en Niveaux de Gris, nous obtenons une nette amélioration de l'ensemble des résultats. Ainsi, l'association des trois mesures de similarité permet de rendre le système plus robuste à la base utilisée. Dans le cas ou la combinaison des deux premiers traitements entraîne une amélioration des résultats, l'utilisation des images en Niveaux de Gris n'entraîne qu'une légère amélioration (figure : 3.25a,3.25b,3.25c,3.25f). Nous avons ensuite utilisé la correction du gradient d'illumination et l'égalisation d'histogramme sur les images en Niveaux de Gris de la troisième corrélation. Nous constatons que contrairement au système précédent, l'amélioration des résultats est très faible rendant de tels traitements presque inutiles.

En conclusion, nous pouvons dire que l'association de la corrélation des contours horizontaux et verticaux obtenus à partir d'un simple filtrage par convolution permet d'atteindre des résultats proches de ceux obtenus en utilisant les corrections d'illumination relativement complexes telles que l'égalisation d'histogramme et la correction du gradient d'illumination. De plus, on remarque que si la corrélation des contours horizontaux donne des résultats satisfaisants, généralement supérieurs à ceux de la corrélation normée centrée des contours obtenus par la méthode de Sobel, ce n'est pas le cas de la corrélation des contours verticaux. L'utilisation d'autres filtres pourrait ainsi permettre d'améliorer les résultats de ce système de détection.

Dans la prochaine section, nous proposons une méthode permettant de déterminer des filtres adaptés aux objets que nous souhaitons détecter.



(a) 5 exemples, base (a)

(b) 10 exemples, base (a)+(b)

(c) 10 exemples, base (a)+(d)

(d) 20 exemples, base (i ... l)

(e) 40 exemples, base (i ... p)

(f) 80 exemples, base (a ... p)

FIGURE 3.25 – Courbes Rappel Précision du système de détection par association de corrélations croisées des images de contours horizontaux et verticaux. Nous observons que cette association permet d'obtenir un système de détection robuste aux variations d'illumination en donnant des résultats proches de ceux obtenus par l'utilisation des traitements permettant, dans une certaine mesure, de corriger ces variations.

3.6.3 Analyse en Composantes Principales Convolutionelle

Dans cette section, nous proposons une méthode basée sur la PCA permettant de déterminer des filtres correspondant aux formes les plus présentes dans les images de l'objet recherché. Ainsi, si l'on trouve de nombreux contours verticaux cette méthode privilégiera la détermination d'un filtre détecteur de contours verticaux. Cette méthode que nous avons nommé Analyse en Composantes Principales Convolutionelle (C-PCA pour Convolutional Principal Composant Analysis) est une généralisation de la PCA classique et de la 2D-PCA (Two Dimensional Principal Component Analysis) [138]. Bien que nous utilisions la C-PCA pour déterminer des filtres, cette méthode comme la PCA classique ou la 2D-PCA peut aussi être utilisée afin de compresser l'information contenue dans une image. Dans les sections suivantes, nous proposons une description de la méthode de la 2D-PCA avant de décrire la C-PCA. Enfin, nous décrirons l'utilisation de la C-PCA pour déterminer des filtres que nous utiliserons en lieu et place de ceux utilisés par la méthode de Sobel.

3.6.3.1 Analyse en Composantes Principales 2D

L'analyse en composantes principales 2D (2D-PCA) a été introduite par Yang *et al* [138] dans le cadre de la reconnaissance de visages. Le but était d'introduire une méthode de représentation inspirée de la PCA mais tenant compte de l'aspect bidimensionnel des images. La PCA a été très largement utilisée en reconnaissance de visages [98, 45, 28] car elle permet par une simple projection linéaire de très nettement réduire la dimension des vecteurs représentant les images de visages. Cependant, la PCA appliquée à la représentation d'images pose plusieurs difficultés. La première est que les matrices représentant les images doivent être transformées en des vecteurs monodimensionnels de très grande taille. Ainsi, un très grand nombre d'images exemples est nécessaire pour obtenir une évaluation précise de la matrice de covariance et par voie de conséquence, des vecteurs propres représentant l'espace de projection des images. Contrairement à la PCA classique, la 2D-PCA est basée sur l'utilisation de matrice à la place de vecteurs monodimensionnels. Ainsi, il n'est plus nécessaire de transformer l'image originale en vecteur. De plus, les dimensions de la matrice de covariance s'en trouvent très nettement réduites, permettant outre une extraction des vecteurs propres bien plus rapide que pour la PCA, une évaluation précise de ces derniers avec un nombre bien plus réduit d'images exemples.

Pour ce faire la 2D-PCA suit le même raisonnement que la PCA classique qui détermine le sous espace orthogonal de projection de dimension k qui minimise la distance L2 moyenne entre les images originales et les images projetées. Cependant, dans le cas de la 2D-PCA, c'est la matrice A de dimension $m \times n$ représentant l'image qui est projetée dans le sous-espace. Ainsi si $\mathbf{v}$ est un vecteur de la base de projection, alors $\mathbf{s}$ le résultat de la projection s'écrit :

$$\mathbf{s} = A\mathbf{v} \tag{3.36}$$

Ainsi le vecteur $\mathbf{v}$ est de dimension n et le vecteur $\mathbf{s}$ de dimension m. Si nous disposons d'une base de N images représentées par les matrices $\{A_1, \ldots, A_j, \ldots, A_N\}$.

Alors, trouver le vecteur $\mathbf{v}$ qui minimise la distance entre les images originales et les images projetées revient à maximiser le critère $J(\mathbf{v})$ suivant :

$$J(\mathbf{v}) = tr(\mathbf{v}^T \Sigma \mathbf{v}) \tag{3.37}$$

$$\Sigma = \frac{1}{N} \sum_{j=1}^{N} (A_j - \bar{A})^T (A_j - \bar{A}) \tag{3.38}$$

$$\bar{A} = \frac{1}{N} \sum_{j=1}^{N} A_j \tag{3.39}$$

$J(\mathbf{v})$ est la trace de la matrice de covariance des vecteurs $\mathbf{s}$. Σ est la matrice de covariance des matrices A et $\bar{A}$ est l'image moyenne de l'ensemble des N images utilisées. Le vecteur $\mathbf{v}$ qui maximise J est le vecteur propre qui correspond à la plus grande valeur propre de la matrice de covariance Σ [137]. Un seul vecteur de projection est généralement insuffisant. Ainsi comme pour la PCA, le sous-espace de projection est formé des k vecteurs propres de la matrice de covariance correspondant aux plus grandes valeurs propres. Le sous-espace de projection W s'écrit alors :

$$W = [\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}] \tag{3.40}$$

L'image est alors décrite par la matrice :

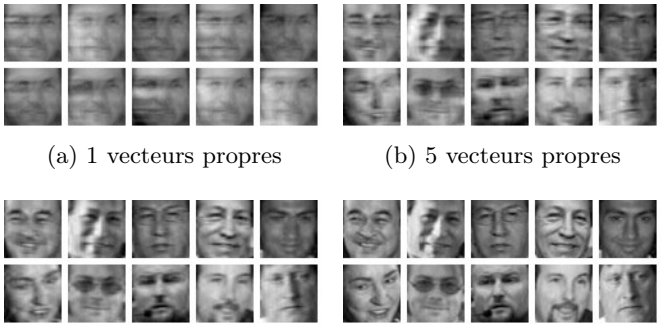
$$S = (A - \bar{A})W \tag{3.41}$$

Reconstruction d'image à partir de la 2D-PCA : La matrice S obtenue par la projection de l'image A dans le sous-espace W permet de décrire l'image originale A de dimension $m \times n$ par la matrice S de dimension $m \times k$ inférieure. A l'inverse, il est possible connaissant la matrice S, le sous-espace W et l'image moyenne $\bar{A}$ de reconstruire plus ou moins précisément en fonction de k l'image originale. Si $\tilde{A}$ est l'image reconstruite, alors, la base W étant orthonormale, nous pouvons écrire :

$$\tilde{A} = SW^T + \bar{A} \tag{3.42}$$

Dans le cas ou $k = n$ alors $\tilde{A} = A$. Plus la valeur de k diminue, plus la reconstruction de $\tilde{A}$ est approximative. Afin de mesurer l'efficacité de la 2D-PCA pour décrire et reconstruire des images, nous avons utilisé une base de données de 80 images de visages (figure : 3.1) de dimension 45×45 afin de calculer la matrice de covariance de la 2D-PCA et de déterminer le sous-espace W de projection. Nous avons ensuite reconstruit des images de visages avec différents nombres de vecteurs propres k (figure : 3.26). Si l'utilisation d'un seul vecteur propre permet de reconnaître que nous avons affaire à des visages, la reconstruction est très approximative et les visages ne sont pas identifiables. Dix vecteurs propres permettent de reconnaître les visages et vingt vecteurs entraînent une reconstruction presque parfaite.

Afin de comparer la 2D-PCA à la PCA, nous avons effectué les mêmes expérimentations pour la PCA (figure : 3.27). Nous constatons que la 2D-PCA nécessite

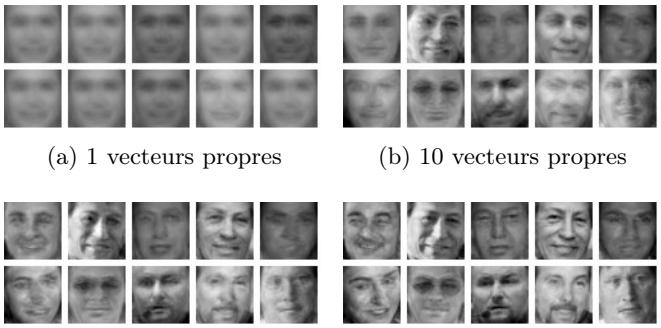


(a) 1 vecteurs propres (b) 5 vecteurs propres

(c) 10 vecteurs propres (d) 20 vecteurs propres

FIGURE 3.26 – Reconstruction d'images avec la 2D-PCA. La base de projection W est calculée à partir de 80 images de dimension 45×45. Une reconstruction parfaite nécessite donc 45 vecteurs propres. Un seul vecteur propre nous permet de reconnaître que nous avons affaire à un visage. Vingt vecteurs propres permettent une reconstruction presque parfaite des images originales.

moins de vecteurs propres pour obtenir une reconstruction avec un niveau de qualité équivalent. Cependant, les erreurs de reconstruction sont assez différentes entre la 2D-PCA et la PCA et il est ainsi difficile de comparer la qualité de reconstruction d'une image par rapport à l'autre.



(a) 1 vecteurs propres (b) 10 vecteurs propres

(c) 20 vecteurs propres (d) 40 vecteurs propres

FIGURE 3.27 – Reconstruction d'images avec la PCA. La base de projection W est calculée à partir de 80 images de dimension 45×45. Une reconstruction parfaite nécessite donc $45 \times 45 = 2025$ vecteurs propres. Une quarantaine de vecteurs propres est nécessaire pour une très bonne reconstruction des images de visages. Une vingtaine de vecteurs propres permet de reconnaître les visages mais entraîne des images relativement floues.

Afin de comparer la qualité de reconstruction de la 2D-PCA et de la PCA, nous avons utilisé une mesure d'erreur de reconstruction simple qui consiste à calculer le ratio entre l'énergie de l'image originale moins l'image reconstruite et l'énergie de

l'image originale. Ainsi, si A est la matrice représentant l'image originale et $\tilde{A}$ celle représentant l'image reconstruite, alors l'erreur de reconstruction E_c se calcule ainsi :

$$En(A) = \sum_{i=j=0}^{i<m,j<n} A(i,j)^2 \tag{3.43}$$

$$E_c = \frac{En(A - \tilde{A})}{En(A)} \tag{3.44}$$

La figure 3.28 nous montre l'erreur de reconstruction E_c de la PCA et de la 2D-PCA en fonction du nombre de vecteurs propres. Nous mesurons ainsi ce que nous avons pu observer sur les figures 3.26 et 3.27, c'est à dire, que pour l'utilisation d'un même nombre de vecteurs propres, l'erreur de reconstruction pour la 2D-PCA est nettement inférieure à celle de la PCA classique.

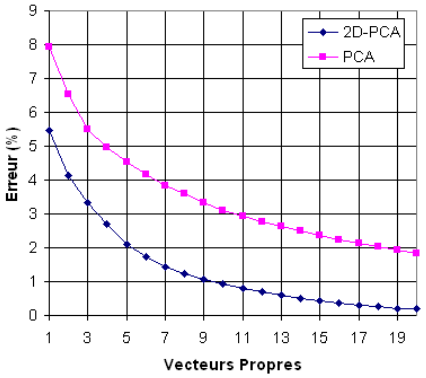


FIGURE 3.28 – Erreur de reconstruction de la PCA et de la 2D-PCA en fonction de la dimension du sous-espace de projection.

Si la 2D-PCA permet de représenter une image à partir d'un sous-espace de dimension plus réduite que la PCA classique, la 2D-PCA reste bien moins performante que la PCA pour la compression d'images. Pour la PCA, le nombre de coefficients permettant de décrire une image est égal au nombre k de vecteurs propres utilisés. Pour la 2D-PCA, le nombre de coefficients nécessaires pour décrire une image de dimension $m \times n = 45 \times 45$ est très supérieur. En effet, A chaque vecteur propre est associé un vecteur $\mathbf{s}$ de m coefficients permettant de décrire chacune des m lignes des images. La projection par la méthode de la 2D-PCA dans le sous-espace W de dimension k entraîne donc $m \times k$ coefficients nécessaires à la description de l'image.

Ainsi, pour atteindre un taux d'erreur de reconstruction de 2%, la 2D-PCA doit utiliser $k = 5$ vecteurs propres et la PCA, $k = 18$ vecteurs propres. Cependant, l'image est alors décrite par 18 coefficients pour la PCA alors que $k \times m = 5 \times 45 = 225$ coefficients sont nécessaires pour la 2D-PCA.

3.6.3.2 Généralisation de l'Analyse en Composantes Principales 2D

Nous avons dans la section précédente brièvement décrit le fonctionnement de la 2D-PCA. Il existe cependant une autre manière d'appréhender cette méthode. En effet, si l'on observe la matrice de covariance Σ de la 2D-PCA (équation 3.38), on remarque qu'elle correspond à la matrice de covariance des lignes de l'ensemble des images exemples centrées (images originales moins l'image moyenne). Si l'on pose $\mathbf{l_{ij}}$ le vecteur correspondant à la i^{ieme} ligne de la j^{ieme} image centrée $(A_j - \bar{A})$ alors :

$$\Sigma = \frac{1}{N}\sum_{j=1}^{N}(A_j - \bar{A})^T(A_j - \bar{A}) \tag{3.45}$$

$$= \frac{1}{N \times m}\sum_{j=1}^{N}\sum_{i=1}^{m}\mathbf{l_{ij}}\mathbf{l_{ij}}^T \tag{3.46}$$

Ainsi, la base de projection W de la 2D-PCA correspond au sous-espace orthogonal obtenu en effectuant l'Analyse en Composantes Principales de l'ensemble des lignes des images centrées. Chaque ligne $\mathbf{l_i}$ des images exemples est projetée dans le même sous espace et est décrite par le vecteur $\mathbf{s_i}$.

$$\mathbf{s_i} = W^T\mathbf{l_i} \tag{3.47}$$

De même, on observe que reconstruire une image par la méthode de la 2D-PCA correspond à reconstruire chaque ligne de cette image de la même façon que pour une PCA classique :

$$\tilde{\mathbf{l_i}} = W\mathbf{s_i} \tag{3.48}$$

Afin de généraliser cette méthode, nous proposons non pas de décrire seulement l'ensemble des lignes d'une image en les projetant dans le même sous-espace orthogonal W, mais de décrire tout rectangle de dimension $(p \times q)$ appartenant à une image A. Ainsi, si nous choisissons $p = 1$ et $q = n$ nous effectuons une 2D-PCA, alors que si nous choisissons $p = m$ et $q = n$ nous effectuons une PCA classique.

Description de la C-PCA : La C-PCA consiste à calculer le sous-espace orthogonal W qui puisse décrire au mieux l'ensemble des imagettes ou régions de dimension $p \times q$ des N images exemples. Pour ce faire, on extrait de chaque image A_j l'ensemble des régions de dimension $p \times q$ que l'on exprime sous forme de vecteurs $\mathbf{x_j}$ et l'on en calcule la matrice de covariance Σ (figure : 3.29). Si nous prenons le cas particulier de nos 80 images de visages de dimension 45×45 et que nous décidons d'appliquer la C-PCA avec $p = q = 5$, alors, chaque image exemple A_j génère $(m - p + 1) \times (n - q + 1) = 1681$ vecteurs $\mathbf{x_j^{uv}}$, (u, v) représentant les coordonnées supérieures gauches des régions X_j^{uv} extraites dans l'image exemple A_j. La matrice de covariance Σ se calcule ainsi :

$$\Sigma = \sum_{j=1}^{N} \sum_{u,v} \mathbf{x_j^{uv}} \mathbf{x_j^{uv}}^T \tag{3.49}$$

Si $p = q = 5$ alors, la matrice Σ est de dimension 25×25. Chaque vecteur $\mathbf{x}$ de $p \times q = 25$ éléments peut alors être décrit par un vecteur $\mathbf{s}$ de k éléments, projection du vecteur $\mathbf{x}$ dans le sous-espace orthogonal W^T formé des k premiers vecteurs propres de Σ.

$$\mathbf{s} = W^T \mathbf{x} \tag{3.50}$$

Nous avons nommé cette méthode PCA Convolutionelle car les descripteurs d'une images A sont alors extraits par convolution. En effet, les éléments s_i du vecteur $\mathbf{s}$ sont le produit scalaire du i^{ieme} vecteur propre avec le vecteur $\mathbf{x_j^{uv}}$ extrait de l'image A_j. Ceci est équivalent à corréler l'imagette X_j^{uv} avec l'image reconstituée V_i du vecteur propre $\mathbf{v_i}$. Ainsi, obtenir la valeur de s_i pour toute coordonnée (u, v) de l'image A revient à convoluer l'image originale avec le filtre V_i. L'image résultante S_i contenant alors en tout point (u, v) la valeur de s_i correspondante.

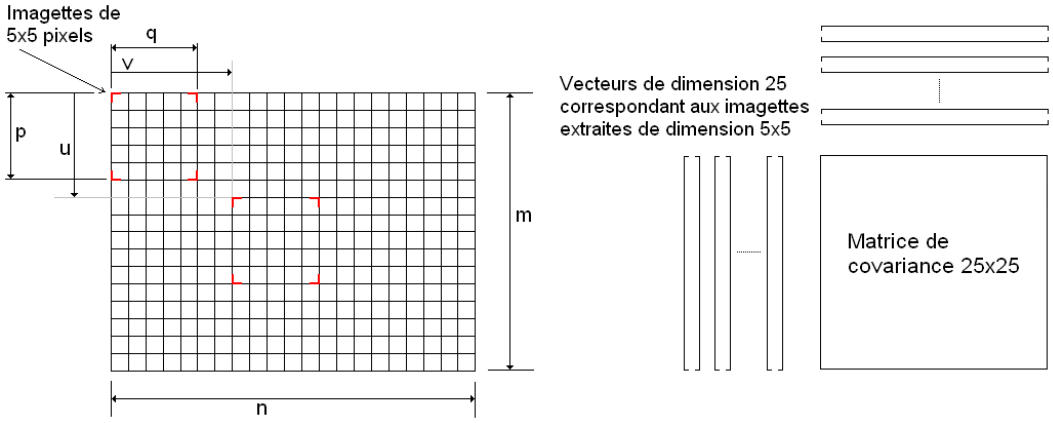


FIGURE 3.29 – Détermination de la matrice de covariance par la méthode de la C-PCA pour des imagettes de dimension $p = q = 5$. Chaque imagette définie par ses coordonnées (u, v) est extraite sous forme de vecteur afin de déterminer la matrice de covariance Σ.

Afin d'illustrer la méthode, nous avons appliquée la C-PCA à la base d'images de 80 visages utilisées précédemment (figure : 3.1). Nous avons choisi les variables $p = q = 5$. La figure 3.30 montre les quatre filtres V_i correspondant aux quatre premiers vecteurs propres obtenus avec la C-PCA. Puis, nous avons appliqué ces filtres à six images de visages représentant ainsi la projection de chaque imagette de dimension 5×5 sur un vecteur propre.

C-PCA et reconstruction d'images : La C-PCA permet de décrire des images par un simple filtrage par convolution en projetant chaque imagette de dimension

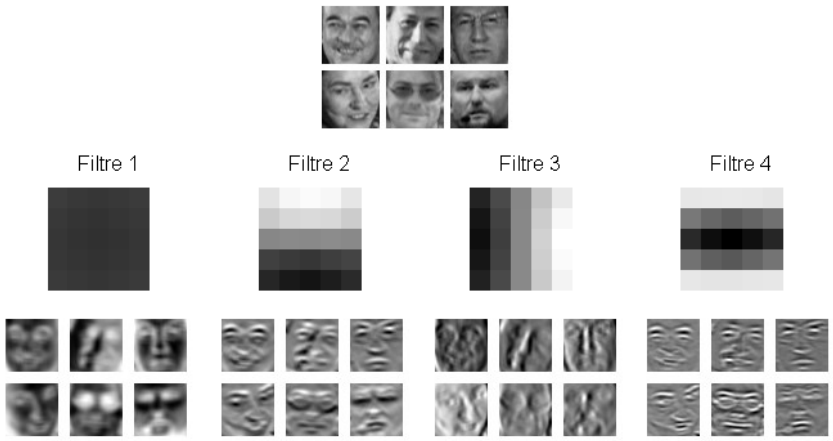


FIGURE 3.30 – Exemples des quatre premiers filtres 5×5 obtenus par la C-PCA, calculés à partir d'une base de 80 images de visages. Ces filtres sont appliqués sur six images de visages. Le premier filtre est un filtre moyenneur. Le second et le quatrième filtre sont des détecteurs de contours horizontaux et le troisième un détecteur de contours verticaux.

(p, q) sur une base de vecteurs propres orthogonaux. Comme pour la 2D-PCA il est évident que cette méthode ne réduit pas autant l'espace de description des images que la PCA classique. Cependant, la compression d'images n'est pas le but de la C-PCA. L'idée de la C-PCA est d'apporter une méthode flexible pour décrire des images. Ces descripteurs pouvant alors comme pour la 2D-PCA être utilisés avec des systèmes de classification pour des problèmes comme la reconnaissance de visages, ou dans notre cas, la détection d'objets. Le choix de la dimension $p \times q$ des filtres que nous extrayons avec la C-PCA permet de décider le type d'information que nous souhaitons extraire des images. Afin de montrer l'influence des variables p et q ainsi que le nombre k de vecteurs propres utilisés, nous avons appliqué la C-PCA en utilisant la même base de 80 images exemples que pour la PCA et la 2D-PCA, puis reconstruit les mêmes visages (figure : 3.31).

Si la reconstruction d'images se fait de façon évidente pour la 2D-PCA et la PCA, certains choix sont nécessaires pour reconstruire les images originales à partir de la C-PCA.

En effet, chaque coefficient de la matrice S_i extraite de l'image A_i par filtrage convolutionnel permet de reconstruire une portion de dimension $p \times q$ de l'image originale. Ainsi, si nous utilisons l'ensemble des coefficients extraits par la C-PCA, nous devons alors tenir compte de la superposition des régions de l'image reconstruite. Nous pouvons imaginer de nombreuses solutions à ce problème. La méthode de reconstruction la plus simple consiste à sous-échantillonner l'image S_i. Si nous sous-échantillonnons S_i horizontalement avec un facteur q et verticalement avec un facteur p, alors nous pouvons reconstruire chaque région de l'image originale sans aucune superposition. De cette manière, nous diminuons aussi nettement le nombre de coefficients nécessaire pour décrire l'image originale. Si nous prenons le cas d'une

(a) p=q=3 k=1 (b) p=q=3 k=2 (c) p=q=3 k=3

(d) p=q=5 k=1 (e) p=q=5 k=3 (f) p=q=5 k=5

(g) p=q=15 k=1 (h) p=q=15 k=5 (i) p=q=15 k=15

FIGURE 3.31 – Reconstruction d'images avec la C-PCA. La base de projection W^T est calculée à partir de 80 images de dimension 45×45. Une reconstruction parfaite nécessite $p \times q$ vecteurs propres. On constate que plus la dimension $p \times q$ des filtres utilisés est importante, plus le nombre k de vecteurs propres nécessaires pour décrire une image augmente.

image de $m \times n = 45 \times 45$ pixels divisée en régions de $p \times q = 5 \times 5$ pixels non superposées, décrites par projection sur $k = 5$ vecteurs propres orthogonaux, alors le nombre de coefficients nécessaires à la reconstruction de l'image avec la méthode de la C-PCA est égal à $\left(\frac{m}{p}\right) \times \left(\frac{n}{q}\right) \times k = 9 \times 9 \times 5 = 405$, contre les 2025 coefficients de la matrice A décrivant l'image originale. Lorsque l'on observe les images de visages reconstruites avec la C-CPA, nous pouvons déjà remarquer que lorsque le nombre de vecteurs propres utilisés est insuffisant on observe aisément l'intersection entre chaque région reconstruite. D'une manière générale, on remarque expérimentalement que pour reconstruire correctement une image de visage, nous devons choisir $k \approx \sqrt{(p \times q)}$. Si nous prenons les cas particuliers de la PCA, *i.e*, $p = m$ et $q = n$, alors, nous obtenons $k = 45$ et pour la 2D-PCA $k \approx 7$ ce qui correspond approximativement à ce que nous observons sur les figures 3.27 et 3.26. De cette règle expérimentale, nous pouvons déduire que le nombre de coefficient nécessaire pour reconstruire correctement une image avec la C-PCA est proportionnel à $\frac{1}{\sqrt{pq}}$. Ainsi, cela confirme que dans le cadre de la compression d'images, la PCA donne les meilleurs résultats.

3.6.3.3 Corrélation avec filtres générés par la C-PCA

Si la PCA permet de décrire les images de visages avec un minimum de descripteurs, elle n'est que difficilement utilisable directement pour des problèmes de détections. Le premier obstacle est le nombre et la dimension des vecteurs propres

de la PCA. En effet, nous savons que pour décrire correctement un visage de 45×45 pixels, nous devons utiliser environ 45 vecteurs propres de dimension $45 \times 45 = 2025$. Projeter chaque image à classifier dans un tel sous-espace est possible mais très coûteux en terme de temps de calculs. Etant donné le très grand nombre d'images à classifier pour les problèmes de détection, cette méthode est difficilement applicable. Une autre difficulté de l'utilisation la PCA pour représenter les images dans un problème de détection est que les descripteurs utilisés doivent permettre de différencier les 'objet' des 'non objet', *i.e*, dans notre cas, un visage du reste du monde. Si la PCA permet de décrire avec très peu de descripteurs les images de visages, ce n'est pas le cas des images du 'reste du monde'. Ainsi deux images de visage et de 'non visage' pourraient avoir les mêmes descripteurs dans l'espace vectoriel de la PCA.

La C-PCA utilise un nombre restreint de vecteurs propres de dimension réduite ($p \times q$). De plus, il suffit pour décrire une image avec la C-PCA d'effectuer une convolution avec les filtres V_i de dimension $p \times q$ de la C-PCA. Nous avons vu dans les sections précédentes que si nous associons les résultats de corrélations normées centrées d'images ayant subit différents traitements, nous améliorons nettement les taux détections. Nous proposons dans cette section d'utiliser les filtres obtenus par la C-PCA en lieu et place des filtres de Sobel que nous avons arbitrairement choisis. Les filtres de la C-PCA possèdent en effet toutes les propriétés que nous recherchons. La première est l'orthogonalité des filtres qui nous permet d'extraire pour chaque filtre une information différente de l'image en Niveaux de Gris. La seconde est que les filtres utilisés correspondent aux formes prédominantes présentes dans les régions de dimension $p \times q$ dans la base d'images exemples.

Comme nous souhaitons nous focaliser sur des filtres extracteurs de contours, nous n'utiliserons pas le premier filtre extrait par la C-PCA qui est un simple filtre moyenneur. En effet, il est évident que la forme la plus répandue est l'image homogène avec très peu de variations. Les autres vecteurs propres correspondent alors aux variations de forme les plus présentes dans l'image, comme les contours horizontaux et verticaux. A titre d'exemple, la figure 3.32, présente les cinq premiers filtres correspondant aux cinq premiers vecteurs propres de la C-PCA pour la base de 80 images de visages (figure : 3.1) pour $p = q =$ 3,4,5 et 6.

Nous pouvons constater que les seconds et troisièmes filtres extraient respectivement les contours horizontaux et verticaux des images. Si nous regardons le cas des filtres 3×3 nous obtenons des filtres aux formes très proches des filtres de Sobel que nous avons précédemment utilisés.

Afin de vérifier l'utilité de tels filtres pour notre système de détection, nous avons remplacé les filtres de Sobel par les filtres de la C-PCA pour différentes valeurs de p et q. Nous avons commencé par les valeurs $p = q = 3$ afin de comparer les résultats de la C-PCA pour des filtres de même dimension que les filtres de Sobel (figure : 3.33).

Nous pouvons alors vérifier que les résultats avec le second et troisième filtres de la C-PCA sont très proches de ceux avec le filtre horizontal et le filtre vertical de Sobel. La combinaison des filtres donne aussi des résultats très légèrement inférieurs mais comparables à ceux obtenus avec Sobel.

Nous avons ensuite comparé la combinaison des seconds et troisièmes filtres de

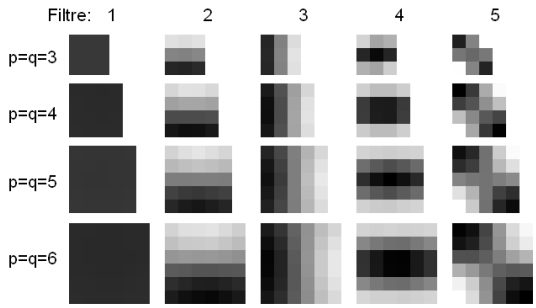
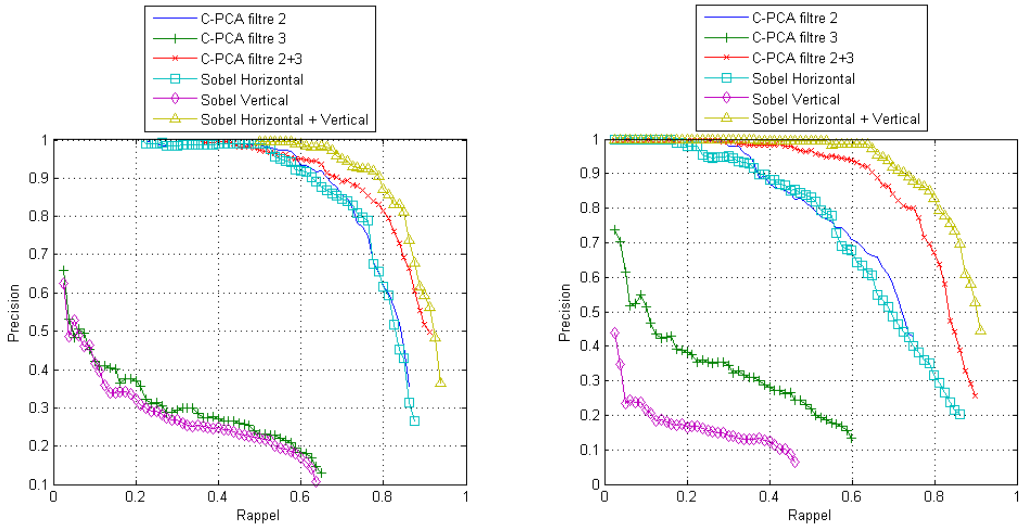


FIGURE 3.32 – Cinq premiers filtres extraits par la C-PCA pour différentes valeurs de p et q. On remarque que nous obtenons les mêmes formes pour les filtres, indépendamment de la dimension des régions extraites. Le premier filtre est un filtre moyenneur, le deuxième et le quatrième filtre extraient des contours horizontaux. Le troisième filtre extrait les contours verticaux et le dernier filtre correspond aux contours diagonaux.

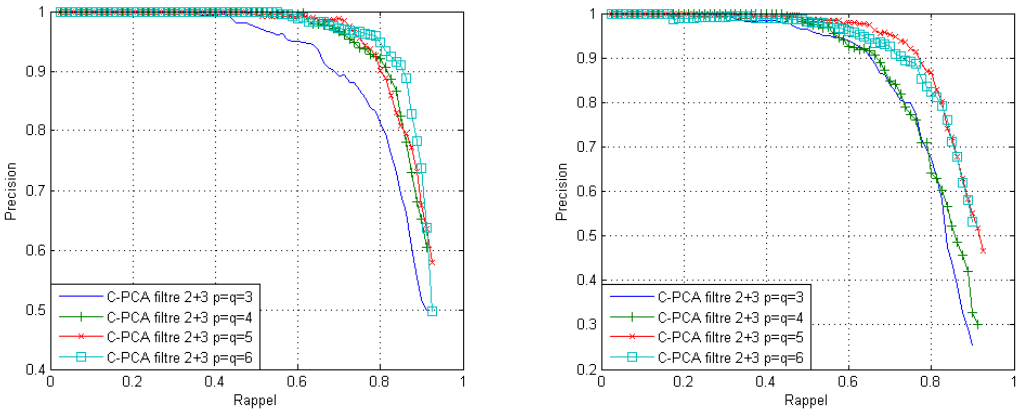


(a) 5 exemples, base (a)

(b) 10 exemples, base (a)+(b)

FIGURE 3.33 – Comparaison des filtres de dimension 3×3 de la C-PCA aux filtres de Sobel pour la détection.

la C-PCA pour des valeurs de $p = q = \{3, 4, 5, 6\}$ (figure : 3.35). L'influence de la dimension des filtres semble assez limité, les résultats sont comparables pour toutes les dimensions, une dimensions de $p = q = 5$ donne des résultats légèrement supérieurs à ceux avec $p = q = 3$ et 4 et équivalents à ceux obtenus avec des dimensions supérieures. Le temps nécessaire au filtrage d'une image étant proportionnels à $p \times q$, le meilleur rapport taux de détection sur puissance de calcul est obtenu pour des filtres de la C-PCA de dimension 5×5.



(a) 5 exemples, base (a)

(b) 10 exemples, base (a)+(b)

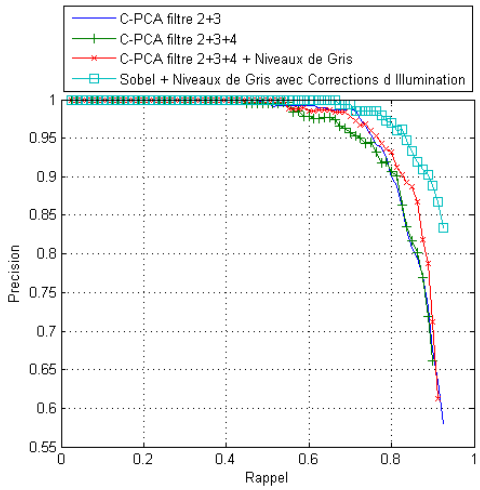
FIGURE 3.34 – Comparaison des filtres de dimension $p = q = 3,4,5,6$ de la C-PCA appliquée au système de détection. Les filtres de dimension $p = q = 5$ sont un bon compromis entre la puissance de calcul nécessaire et les taux de détection obtenus

Enfin, nous avons utilisé les filtres 5×5 de la C-PCA avec différentes bases d'exemples dans le but de mesurer l'efficacité et la sensibilité au nombre d'exemples du détecteur basé sur la C-PCA, de mesurer l'influence du nombre de filtres de la C-PCA sur les taux de détection et enfin, de permettre de comparer les résultats aux autres méthodes basées sur la corrélation (figure : 3.35).

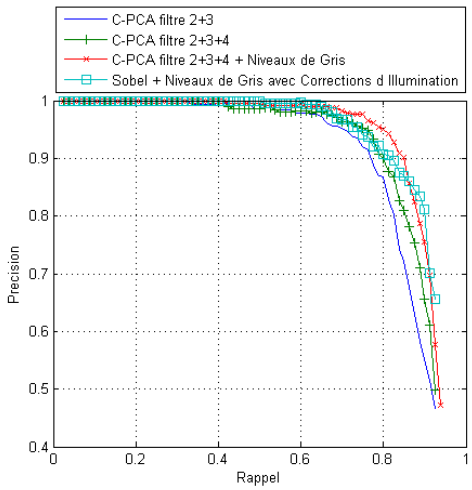
Nous constatons d'abord que plus nous utilisons de filtres de la C-PCA, meilleurs sont les résultats, et ceci quel que soit le nombre d'exemples. L'utilisation des images en Niveaux de Gris en plus des images filtrées permet une légère amélioration. L'utilisation de seulement deux filtres donne des résultats assez variables en fonction de la base d'exemples. L'utilisation d'un troisième filtre permet de rendre le système beaucoup plus robuste à la base utilisée. En effet, lorsque les résultats avec deux filtres sont assez faibles (figure : 3.35e, 3.35) le troisième filtre permet une très nette amélioration des résultats. Lorsque les résultats avec deux filtres sont plus proches de ceux obtenus avec les systèmes de détection précédents, l'apport du troisième filtre sur les taux de détection est bien plus faible. Enfin, l'utilisation de l'image en Niveaux de Gris, qui nous emmène donc à utiliser une combinaison de quatre mesures de similarité, obtient les meilleurs résultats que nous ayons obtenus avec les bases de 40 et 80 exemples.

Au final, nous pouvons dire que l'utilisation des filtres générés par la C-PCA permet d'obtenir un système de détection basé sur la corrélation, particulièrement robuste à la base d'exemples utilisée, et ceci sans effectuer de traitements d'image complexes comme une déformation affine ou des corrections d'illumination. Les filtres générés étant assez proches de ceux utilisés par Sobel, nous obtenons des résultats similaires, avec l'avantages que nous ne sommes pas limités à seulement deux filtres pour la C-PCA, ce qui nous permet une légère amélioration des résultats par rapport

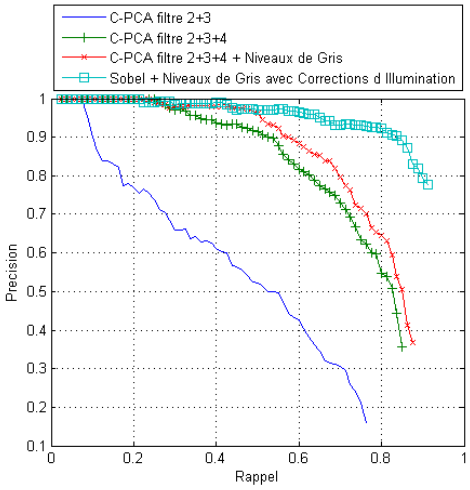
à l'utilisation des filtres de Sobel.



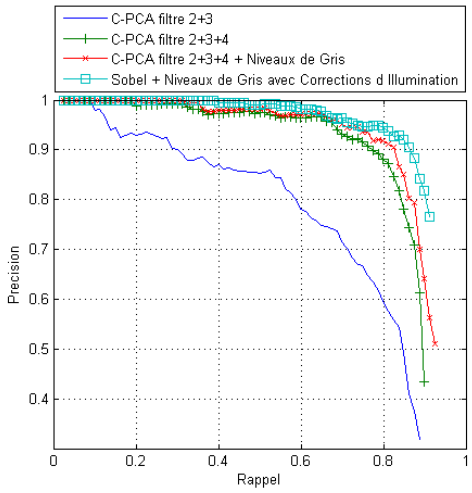
(a) 5 exemples, base (a)



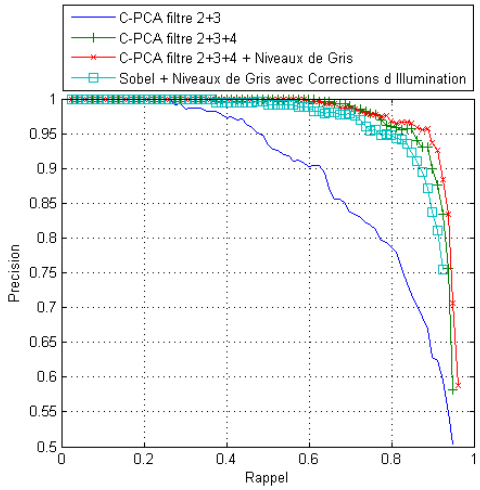
(b) 10 exemples, base (a)+(b)



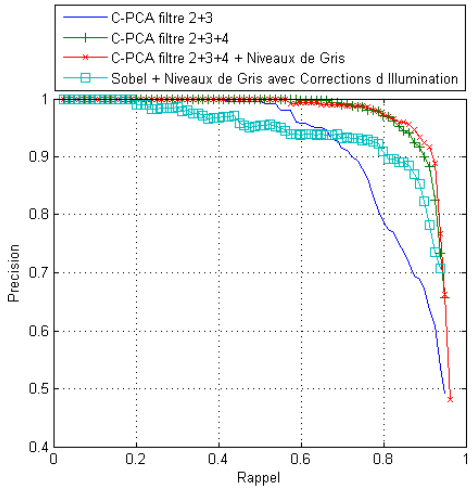
(c) 10 exemples, base (a)+(d)



(d) 20 exemples, base (i ... l)



(e) 40 exemples, base (i ... p)



(f) 80 exemples, base (a ... p)

FIGURE 3.35 – Courbes Rappel Précision du système de détection par association de corrélations croisées des images filtrées par les filtres de la C-PCA. Nous observons que cette association permet d'obtenir un système de détection fonctionnel avec très peu d'exemples.

3.7 Evaluation du système de détection sur la base de test CMU

De part la complexité de la tâche consistant à détecter des visages avec une mesure de similarité basée sur la corrélation, nous avons choisi d'évaluer notre système de détection sur une base de test plus simple que les bases de test standards. Cependant, nous avons montré qu'en associant plusieurs mesures de similarité basées sur la corrélation, nous pouvons obtenir d'assez bons résultats. Nous proposons dans cette section d'appliquer les systèmes de détection basés sur l'association de corrélations à la base de test la plus couramment utilisée en détection de visages, *i.e*, CMU.

Dans ce chapitre nous pouvons distinguer trois méthodes qui ont permis d'obtenir des résultats satisfaisants.

- La première méthode que nous nommerons 'Association 1' présentée dans la section 3.5 est l'association de la corrélation d'images de contours extraits par la méthode de Sobel et d'images en Niveaux de Gris traitées de façon à minimiser l'effet des variations d'illumination.
- La seconde méthode que nous nommerons 'Association 2' présentée dans la section 3.6.2 utilise les filtres de Sobel verticaux et horizontaux afin d'associer la corrélation des contours verticaux, horizontaux et les Niveaux de Gris des images.
- La troisième méthode que nous nommerons 'Association 3' présentée dans la section 3.6.3.3 remplace les filtres de Sobel par les seconds, troisièmes et quatrièmes filtres déterminés à partir de la méthode de la C-PCA.

La première difficulté est que pour effectuer ces tests, notre système de détection doit être capable de détecter des visages de dimension 25×25 alors que nous nous sommes jusqu'à présent limités à des images exemples de dimension 44×44. Nous avons vu dans les sections 3.3.3 et 3.4.2 que plus la dimension des images exemples est réduite, plus les taux de détection sont faibles. Nous avons ensuite fixé la dimension des images exemples à 44×44 pour les expérimentations sur les systèmes de détection basés sur l'association de corrélations. C'est pourquoi, nous commençons dans cette section par évaluer sur la base de test 'Face 1999', l'influence de la diminution de la dimension à 25×25 pixels des images de référence. Nous utilisons une base de 5 et 80 exemples afin de déterminer le comportement du système avec très peu d'exemples puis avec un nombre important. Les systèmes 'Association 2' et 'Association 3' sont utilisés exactement de la même manière que précédemment en ne changeant que la dimension des images exemples. Pour le système 'Association 1', nous modifions la valeur du seuil de prédétection s_1. En effet, nous avons vu que pour cette méthode ce seuil doit être réglé de manière assez précise. Il était égal à 0.72 pour les exemples de 44×44 pixels et est réglé à une valeur de 0.79 pour les images de 25×25 pixels.

Nous constatons sur la figure 3.36 que si les trois systèmes testés donnaient des résultats du même ordre pour des images exemples de 44×44 pixels, ceci n'est pas le cas avec les images exemples de tailles plus réduites (25×25). En effet, les systèmes basés sur l'association de filtres détecteurs de contours orientés 'Association 2 et 3'

donnent des résultats supérieurs au système 'Association 1' basé sur la corrélation d'images traitées par l'algorithme de Sobel et des images en Niveaux de Gris traitées de manière à minimiser l'effet des variations d'illumination. Ainsi, l'association de plusieurs mesures de similarité basées sur la corrélation et des filtres convolutionnels permet d'obtenir des résultats supérieurs aux systèmes basés sur des traitements d'image plus complexes tels que la correction du gradient d'illumination ou l'égalisation d'histogramme.

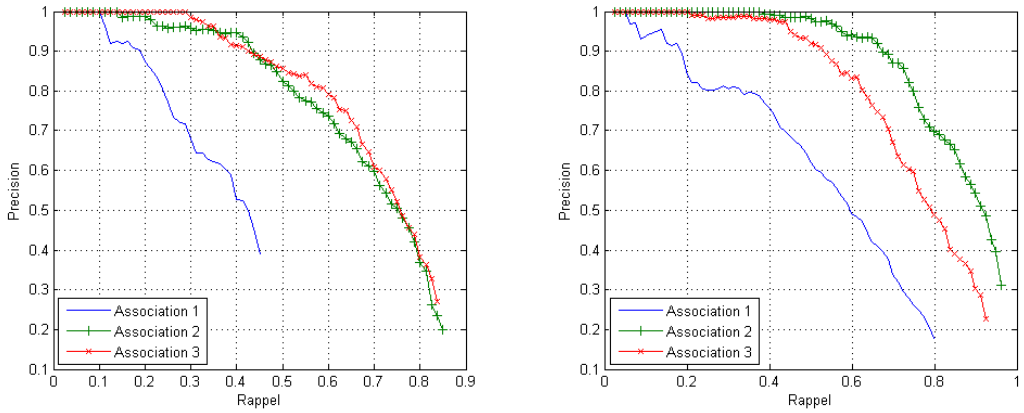
Dans une certaine mesure, nous pouvons rapprocher un tel résultat avec le fonctionnement des réseaux de neurones convolutionnels. En effet, le système de détection de Garcia et Delakis [43] effectue sur la première couche du réseau de neurones plusieurs filtrages convolutionnels avec des filtres de dimension 5×5. Ce système est le système de détection de visages le plus performant tout en étant le seul système de détection qui ne nécessite aucun prétraitement d'image corrigeant les variations d'illumination.

Nous pouvons aussi constater que les taux de détection sont très nettement inférieurs à ceux obtenus avec des images exemples de 44×44 pixels (figures : 3.19, 3.25 et 3.35). Il semble donc difficile d'atteindre les même taux de détection que l'état de l'art avec une mesure de similarité basée sur la corrélation.

Lorsque nous appliquons les trois systèmes de détection à la base de visages CMU, nous pouvons déjà constater, que comme sur la base 'Face 1999', les systèmes 'Association 2 et 3' sont plus performants que le système 'Association 1'. La principale différence entre les résultats obtenus sur les deux bases de test est l'influence du nombre d'exemples de la base de référence. En effet, sur la base de test 'Face 1999' il est presque équivalent d'utiliser 5 exemples (figure : 3.37a) ou 80 exemples (figure : 3.37b). Sur la base de test CMU, les résultats sont nettement inférieurs avec la base de référence de 5 exemples comparés à ceux obtenus avec la base de 80 exemples. Ceci s'explique par la plus grande diversité des visages présents dans la base CMU qui ne peuvent être correctement représentés par seulement 5 exemples.

L'intérêt de la base de test CMU est qu'elle nous permet de nous comparer aux systèmes de détection de l'état de l'art. La plupart de ces méthodes ne donnent que le Rappel pour un nombre donné de fausses détections. Nous reportons dans le tableau 3.1 les résultats publiés pour les principaux systèmes de l'état de l'art, ainsi que le nombre d'exemples utilisés dans la phase d'apprentissage et incluons les résultats obtenus par nos trois systèmes de détection basés sur la corrélation. Nos résultats sont très loin de ceux de l'état de l'art. Cependant, il faut noter que nous utilisons beaucoup moins d'exemples d'apprentissage que ces méthodes et aucun classifieur complexe. Ainsi notre système fonctionne avec près de cinquante fois moins d'exemples que les systèmes traditionnels et est capable de détecter des visages de dimension réduite dans des environnements complexes. La figure 3.38 montre des exemples de détections sur la base CMU et met en évidence à la fois, la capacité de la corrélation à détecter des visages dans un environnement complexe, mais aussi les limites des méthodes utilisant la corrélation.

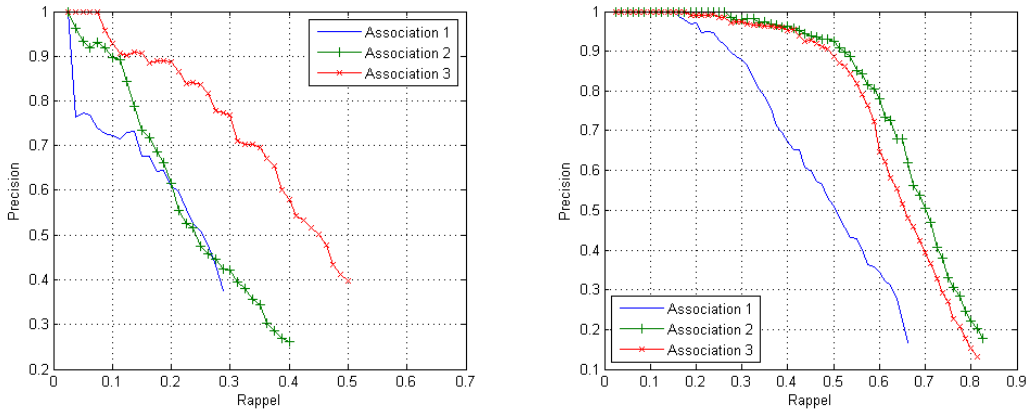
Un autre avantage de la méthode basée sur la corrélation et les plus proches voisins, est que comme nous pouvons le voir sur la figure 3.38, nous somme capables de repérer assez précisément la position de points d'intérêt tels que le nez, les yeux



(a) 5 exemples, base (a)

(b) 80 exemples, base (a ... p)

FIGURE 3.36 – Courbes Rappel Précision des trois systèmes de détection par association de corrélations sur la base 'Face 1999' avec des images exemples de 25×25 pixels.



(a) 5 exemples, base (a)

(b) 80 exemples, base (a ... p)

FIGURE 3.37 – Courbes Rappel Précision des trois systèmes de détection par association de corrélations sur la base CMU avec des images exemples de 25×25 pixels.

ou la bouche. En effet la position de ces points d'intérêt ayant manuellement été annotée sur chaque image exemple, nous pouvons la reporter sur chaque détection correspondante.

Ainsi l'Association de plusieurs corrélations permet d'obtenir des résultat intéressants, en particulier si on les compare à ceux obtenus par la corrélation croisée normée sans association (figure : 3.11) qui ne semblait pas permettre l'application de la corrélation pour détecter des visages sur des bases de test complexes telles que

Système de détection	Fausses détections				
	0	10	31	65	167
Rowley *et al* [47] (4000)	-	83,2%	86,0%	-	90,1%
Viola-Jones [131] (4916)	-	78,3%	85,2%	89,8%	91,8%
CFF [43] (3702)	88,8%	90,5%	91,5%	92,3%	93,1%
Association 1 (80)	14,7%	24,7%	30,7%	35,2%	44,4%
Association 2 (80)	25,2%	40,9%	51,7%	56,2%	64,2%
Association 3 (80)	17,0%	40,1%	48,1%	54,6%	58,7%

TABLE 3.1 – Comparaison du Rappel des systèmes de détection pour divers nombre de fausses détections sur la base de test CMU. Les nombres entre parenthèses correspondent au nombre d'exemple de la base d'apprentissage.

CMU.

Dans le chapitre suivant, nous proposons de garder le principe de l'association de mesures de similarité en remplaçant la corrélation par une mesure plus complexe basée sur les réseaux de neurones qui ont montré leur efficacité en détection d'objets.

FIGURE 3.38 – Exemples de détections effectuées sur la base CMU à partir du système basée sur la C-PCA et l'association de corrélations croisées normées centrées

CHAPITRE 4

Détection par mesures de similarité discriminatives

4.1 Introduction

Les systèmes de détection d'objets et en particulier les systèmes de détection de visages de l'état de l'art sont tous basés sur des mesures de similarité discriminatives. Malgré la difficulté induite par la nécessité de représenter la classe 'non objet', ces méthodes semblent permettre une meilleure classification que les méthodes génératives. La corrélation, associée à la méthode des plus proches voisins, a permis d'obtenir un système de détection fonctionnel mais dont les résultats semblent limités par la méthode de classification choisie. Afin d'améliorer les taux de détection tout en utilisant toujours une base d'exemples de dimension réduite, nous proposons, dans ce chapitre, d'utiliser les méthodes mises au point pour les systèmes de détection par corrélation en les adaptant à une mesure de similarité discriminative plus performante. Pour ce faire, nous avons remplacé la corrélation par un Perceptron Multicouche. Nous avons ensuite appliqué les deux idées qui ont donné les meilleures améliorations à la méthode par corrélation, c'est à dire, l'association de classifieurs et l'utilisation de filtres détecteurs de contours. Nous montrerons à la fois l'efficacité de l'utilisation d'un Perceptron Multicouche en lieu et place d'une simple corrélation, mais aussi que nos conclusions sur les méthodes que nous utilisons avec la corrélation se généralisent à des classifieurs plus complexes et performants. Nous commencerons par décrire un système de détection basé sur un MLP et montrerons l'influence de la forme du MLP choisie, du nombre d'exemples et de la méthode de BootStrapping. Nous appliquerons ensuite les méthodes que nous avons mises au point précédemment et montrerons comment elles influencent les résultats sur le problème particulier de la détection de visages.

4.2 Système de Détection basée sur un MLP avec peu d'exemples

Dans cette section nous utilisons un Perceptron Multicouche appliqué aux images en Niveaux de Gris, au problème de la détection de visages. Nous commencerons par décrire le fonctionnement d'un tel système, puis nous donnerons différents résultats expérimentaux sur la base de test de détection de visages CMU. Nous nous intéresserons à l'influence du nombre d'exemples, à la forme du MLP ainsi qu'à la normalisation des images.

4.2.1 Fonctionnement du système de détection

Le système de détection que nous employons ici est un système de détection 'classique' (figure : 1.2) où nous utilisons un MLP directement appliqué aux images en Niveaux de Gris comme classifieur. Nous nous servons d'une rétine de 25×25 pixels afin que notre système puisse fonctionner sur les bases de test standards de détection de visages. La pyramide d'images permettant la détection multi-échelle utilise un facteur de sous-échantillonnage de 1.2.

Nous commencerons par utiliser un MLP à trois couches entièrement connecté (figure : 2.15) avec n_2 neurones dans la couche cachée. Le nombre n_1 de neurones de la première couche est directement lié à la dimension de la rétine puisque l'entrée du réseau est un vecteur représentatif de l'image contenue dans la rétine. La troisième et dernière couche comporte un seul et unique neurone. Le réseau de neurones est entraîné de façon à retourner la valeur 1 si l'image d'entrée représentée par le vecteur $\mathbf{x}$ est un visage (ou plus généralement l'objet à détecter) et -1 dans le cas contraire. Les fonctions d'activation utilisées sont les fonctions tangentes hyperboliques (figure : 2.16).

4.2.1.1 Entraînement du perceptron

Les N premiers exemples de la base de visages 'Caltech WebFaces' sont utilisés comme images de référence. Une base de 450 images ne contenant pas de visage est utilisée pour l'algorithme de BootStrapping. Afin de minimiser les risques de sur-apprentissage liés au faible nombre d'exemples ainsi que d'obtenir un système d'apprentissage plus rapide qu'avec l'utilisation classique de la méthode de BootStrapping, nous avons mis au point une variante de cette méthode.

Comme pour l'algorithme de BootStrapping classique, nous utilisons les fausses détections obtenues sur les images ne contenant aucun visage pour générer la classe 'non visage'. Cependant les conditions d'arrêt de l'algorithme, ainsi que la méthode d'apprentissage utilisée différent de la méthode originale. L'algorithme classique de BootStrapping consiste à obtenir un certain nombre d'exemples de 'non visage' puis d'utiliser un algorithme d'apprentissage, généralement l'algorithme de Backpropagation pour les MLP. Cette opération étant répétée un certain nombre de fois, généralement jusqu'à ce que l'ajout de nouveaux exemples n'améliore plus le taux de détection du système.

Ici, l'apprentissage et l'algorithme de BootStrapping sont plus imbriqués que dans l'algorithme classique. L'algorithme de Backpropagation Stochastique utilisé ne diverge de l'algorithme classique (Algo : 5) que par les conditions d'arrêt et la sélection des exemples de 'visage' et de 'non visage'.

La condition d'arrêt consiste à stopper l'apprentissage lorsque l'on atteint un nombre donné d'itérations max_{iter} sans vérifier la convergence de l'algorithme de Backpropagation. La valeur de max_{iter} est relativement faible, généralement égale au nombre d'exemples utilisés pour l'apprentissage.

De façon à donner autant d'importance à la classe 'visage' qu'à la classe 'non visage' dont les dimensions peuvent être très différentes, l'algorithme alterne systématiquement entre un exemple de 'visage' aléatoirement choisi et un exemple de 'non visage'.

L'algorithme de Backpropagation stochastique est alors défini comme suit :

Algorithm 8 Stochastic Backpropagation V2

initialition : η, $\mathbf{w}$, n_2, θ, max_{iter}, $m = 0$
while $m < max_{iter}$ **do**
  $m = m + 1$
  if m pair **then**
    $\mathbf{x} \leftarrow$ échantillon 'visage' exemple choisi aléatoirement
  else
    $\mathbf{x} \leftarrow$ échantillon 'non visage' exemple choisi aléatoirement
  end if
  $\delta_k = (t_k - z_k)\,\varphi'_k\,(net_k)$
  $\delta_j = [\sum_{k=1}^{c} \delta_k w_{kj}]\,\varphi'_j\,(net_j)$
  $w_{kj} = w_{kj} + \eta \delta_k y_j$
  $w_{ji} = w_{ji} + \eta \delta_j x_i$
end while
return $\mathbf{w}$

Ainsi, une seule utilisation de l'algorithme de Backpropagation est insuffisante pour converger vers une solution de classification optimale au sens des moindres carrés.

Le but de notre algorithme de BootStrapping que nous nommerons Online BootStrapping est de minimiser le taux de fausses détections (FAR) sur les images de la base ne contenant aucun visage. Pour ce faire, chaque erreur détectée par le classifieur est ajoutée à la base de 'non visage'. Lorsque le nombre de fausses détections atteint un nombre N_{fdmax} donné (nous avons choisi $N_{fdmax} = 100$), nous entraînons le réseau de neurones à l'aide de l'algorithme de Backpropagation (Algo : 8). Afin de se focaliser sur les exemples de 'non visage' les plus significatifs, nous limitons le nombre total N_{ft} d'exemples de 'non visage' à $N_{ftmax} = 3000$. Si le nombre d'exemples de 'non visage' est supérieur à N_{ftmax}, alors nous supprimons les échantillons les moins pertinents, *i.e*, ceux dont le score donné par le classifieur s'approche le plus de la valeur souhaitée (-1). L'algorithme de BootStrapping s'arrête lorsque le taux de fausses détections (ou fausses alarmes) FAR est inférieur à un seuil FAR_{min} donné.

Le taux de fausses détections est calculé tous les $N_{test} = 2 \times 10^6$ tests et correspond au rapport du nombre de fausses détections effectuées N_{fd} sur le nombre de tests, *i.e*, $FAR = \frac{N_{fd}}{N_{test}}$.

Le fonctionnement de l'algorithme d'Online BootStrapping se résume donc ainsi :
N_{ftmax} : nombre maximum d'échantillons de 'non visage'.
N_{ft} : nombre total d'échantillons de 'non visage' utilisés pour l'apprentissage.

1. Réunir un petit nombre d'échantillons exemples de la classe 'non visage' et un ensemble d'échantillons représentatif de la classe 'visage'
2. Entraîner le MLP à partir de la base d'échantillons exemples courante par l'algorithme de Backpropagation.
3. Supprimer les échantillons de 'non visage' dont le score de classification est le plus proche de '-1' de façon à ce que $N_{ft} \leq N_{ftmax}$.
4. Utiliser le système de détection sur une base d'images ne contenant pas la classe à détecter. Obtenir N_{fdmax} fausses détections supplémentaires.
5. Retourner à l'étape 2 jusqu'à ce que $FAR = \frac{N_{fd}}{Ntest} < FAR_{min}$

Cette méthode permet ainsi, non seulement une convergence rapide du classifieur en le focalisant au maximum sur les exemples les plus significatifs mais permet en plus, grâce à la condition d'arrêt utilisée de minimiser les risques de sur-apprentissage, ce qui est nécessaire à notre système qui doit être capable de fonctionner avec peu d'exemples.

Afin de mesurer l'efficacité de la méthode d'apprentissage, nous observerons trois paramètres qui nous permettrons de caractériser les performances de cet algorithme.

1. Le Rappel (R) : le nombre de visages exemples dont le MLP renvoie un score supérieur à zéro sur le nombre total d'exemples.
2. Le taux de fausses alarmes (FAR) : $FAR = \frac{N_{fd}}{Ntest}$. $N_{test} = 2 \times 10^6$
3. Le nombre de fichiers images ne contenant pas de visages utilisés pour converger vers une solution ($NbIm$) (cela nous permet de caractériser la vitesse de convergence de la méthode)

4.2.1.2 Regroupement des résultats

Afin d'éviter les détections superposées, nous utiliserons une méthode qui a montré son efficacité, notamment dans le cadre du détecteur de visages de Garcia et Delakis.

Chaque rétine 25×25 dans la pyramide d'images pour laquelle le classifieur renvoie un score supérieur à zéro est considérée comme une détection. Le système de regroupement choisit la détection avec le score le plus proche de 1. L'ensemble des détections superposées à cette dernière est alors regroupé en une seule détection dont l'échelle E et la position $\mathbf{p}$ sont les moyennes pondérées des détections superposés. Le score s_r, correspond à la somme des scores de détection. Ainsi, plus le nombre de

superpositions avec un score proche de 1 est important, plus le score de détection final sera élevé. L'opération est ensuite réitérée jusqu'à ne plus avoir de superpositions.

Si nous avons K détections superposées, E_i, $\mathbf{p_i}$ et s_i représentant respectivement l'échelle, la position et le score de la i^{ieme} détection superposée, alors :

$$s_r = \sum_{i=0}^{i<K} s_i \tag{4.1}$$

$$E = \frac{1}{s_r} \sum_{i=0}^{i<K} s_i E_i \tag{4.2}$$

$$\mathbf{p} = \frac{1}{s_r} \sum_{i=0}^{i<K} s_i \mathbf{p_i} \tag{4.3}$$

4.2.2 Résultats expérimentaux

Comme pour la corrélation, de nombreux paramètres peuvent influer sur les résultats d'un système détection basé sur un MLP. Nous pouvons citer en tout premier lieu, la forme du réseau de neurones, les traitements appliqués à l'image en Niveaux de Gris ou encore le nombre d'exemples de la base d'apprentissage. Nous proposons dans cette section d'observer les influences de ces différents paramètres sur notre système de détection. Nous commencerons comme pour la corrélation, par observer l'influence de la normalisation des vecteurs représentant les images en Niveaux de Gris, puis nous observerons les effets de la variation du nombre de neurones cachés du MLP, du nombre d'exemples d'apprentissage et finalement de la forme du MLP.

Afin d'effectuer ces tests nous utilisons la base 'Caltech WebFaces' comme base d'images de visages exemples. Plus exactement, les N échantillons exemples utilisés pour l'apprentissage correspondent aux N premiers visages extraits de cette base.

4.2.2.1 Influence de la normalisation

Nous avons vu, pour la corrélation que l'utilisation directe des images en Niveaux de Gris, sans normalisation, était impossible avec une mesure de similarité aussi simple. Dans cette section, nous proposons d'observer l'influence de la normalisation sur les taux de détection d'un système utilisant un réseau de neurones comme mesure de similarité. Pour ce faire, nous effectuerons les tests de détection de visages, sur la base CMU.

D'une certaine manière, un MLP est un système de détection par corrélations. En effet, ce dernier classe les images d'entrée grâce à diverses corrélations effectuées par chaque neurone du réseau. La normalisation $L2$ centrée donnant de meilleurs résultats que la normalisation simple pour la corrélation, c'est donc la première que nous utiliserons sur les vecteurs d'entrée du MLP.

Ainsi, si $\mathbf{x} = (x_1 \dots x_d)^T$ est le vecteur de dimension d qui représente l'image en Niveaux de Gris, le vecteur d'entrée normalisé $\tilde{x}$ s'écrit :

$$m = \frac{1}{d}\sum_{i=1}^{d} x_i \tag{4.4}$$

$$\sigma = \sqrt{\sum_{i=1}^{d}(x_i - m)^2} \tag{4.5}$$

$$\tilde{x} = \frac{\mathbf{x} - m}{\sigma} \tag{4.6}$$

$$\mathbf{x} - m = (x_1 - m, \ldots, x_d - m)^T \tag{4.7}$$

Les paramètres du système de détection n'étant pas indépendants les uns des autres, nous proposons d'effectuer les tests avec peu d'exemples, *i.e*, 80, puis 1000 exemples ainsi qu'avec 10 puis 40 neurones cachés. De cette manière, nous obtiendrons des résultats pour des valeurs extrêmes des paramètres du système et en tirerons des conclusions assez générales sur l'influence de la normalisation.

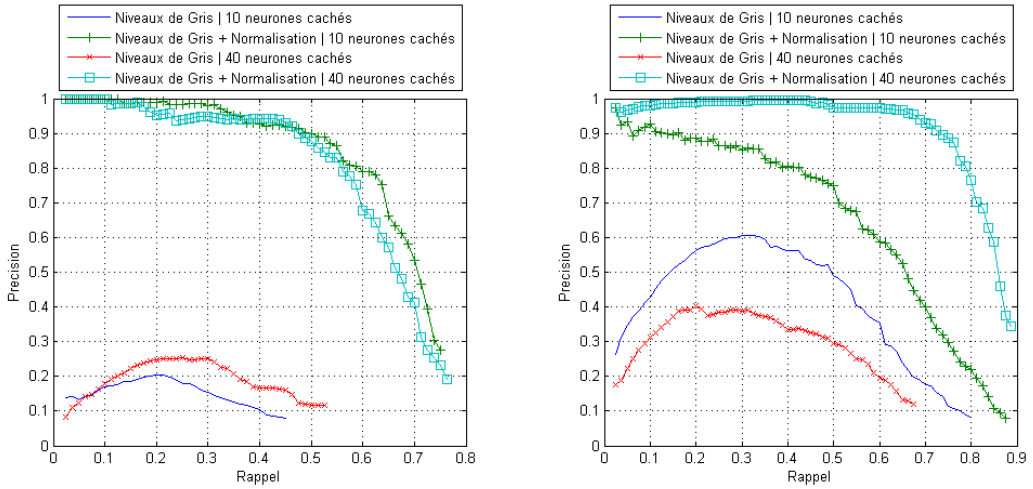
Les courbes Rappel Précision obtenues (figure : 4.1) nous montrent que comme pour la corrélation, la normalisation est indispensable au bon fonctionnement du système de détection. En effet, sans la normalisation des images en Niveaux de Gris, les résultats de notre système de détection sont très nettement inférieurs à ceux obtenus avec la normalisation. Nous pouvons aussi constater que l'utilisation d'un plus grand nombre d'exemples d'apprentissage améliore les résultats, avec ou sans l'utilisation de la normalisation. Cependant, l'utilisation de plus de neurones cachés ne garantit pas une amélioration des résultats, en particulier, avec 1000 exemples, les résultats sans normalisation sont supérieurs avec 10 neurones cachés qu'avec 40.

Le tableau 4.1 résume les caractéristiques concernant la convergence du MLP en fonction des paramètres du système de détection (nombre de neurones cachés n_2, nombre d'exemples N et normalisation). Sans normalisation, nous constatons que pour 80 exemples, le Rappel est très bon puisque supérieur à 90% et même égal à 1 dans le cas où nous utilisons dix neurones cachés. Cependant, étant donné le faible nombre d'exemples d'apprentissage, ce taux n'est pas très significatif et nous pouvons constater que les résultats du système de détection sont assez modestes. Lorsque nous utilisons milles exemples d'apprentissage, le Rappel n'est plus que de l'ordre de 70% alors qu'il est de plus de 95% avec la normalisation, ce qui explique la différence entre les résultats du système de détection avec et sans la normalisation.

En ce qui concerne le temps nécessaire à la convergence de l'algorithme d'Online BootStrapping, nous pouvons remarquer une tendance générale à l'augmentation du nombre de fichiers images ($NbIm$) nécessaires avec l'augmentation du nombre d'exemples d'apprentissage et avec le nombre de neurones cachés. Cependant, la valeur de ($NbIm$) reste assez variable et aléatoire.

4.2.2.2 Influence du nombre de neurones cachés

Nous avons vu, dans la section précédente que la normalisation est indispensable pour que l'algorithme de Backpropagation converge vers une solution qui per-



(a) 80 exemples (b) 1000 exemples

FIGURE 4.1 – Courbes Rappel Précision du système de détection basé sur un MLP et les images en Niveaux de Gris. Influence de la Normalisation. La normalisation apporte un gain substantiel aux résultats du système de détection.

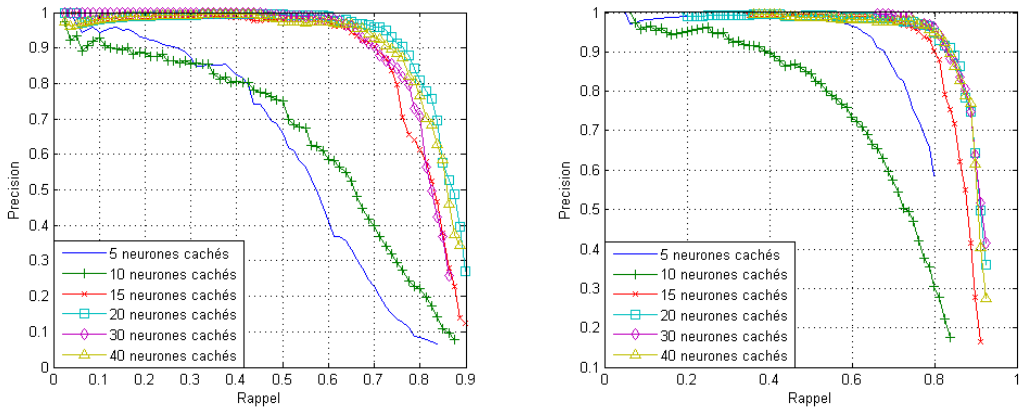
	Niveaux de Gris				Normalisation			
Neurones Cachés	10		40		10		40	
Visages Exemples	80	1000	80	1000	80	1000	80	1000
Rappel (%)	100	71,2	92,5	74,3	99,4	99,4	93,5	97,5
FAR ($\times 10^{-6}$)	46	47	21	46	48	42	46	40
NbIm	137	456	414	1126	26	460	110	330

TABLE 4.1 – Performances de l'algorithme d'apprentissage du MLP en fonction de la normalisation des vecteurs d'entrée, du nombre de neurones cachés et du nombre d'exemples de la base de référence.

met d'obtenir un système de détection fonctionnel. Avec 80 exemples, nous obtenons les mêmes résultats en utilisant 10 ou 40 neurones cachés. Cependant, nous voyons qu'avec 1000 exemples, l'augmentation du nombre de neurones cachés apporte un gain substantiel au niveau des taux de détection. Dans cette section, nous souhaitons mettre en évidence l'influence du nombre n_2 de neurones cachés. Pour ce faire, nous utiliserons 1000 puis 4000 exemples dans la base d'apprentissage afin de ne pas limiter les taux de détection par la faiblesse du nombre d'exemples. Nous donnons figure 4.2 et 4.3 les courbes Rappel Précision ainsi que le Rappel du système de détection pour 31 fausses détections pour un nombre de neurones cachés compris entre 5 et 40. Pour n_2 compris entre 5 et 15, l'augmentation du nombre de neurones cachés entraîne globalement une amélioration des résultats. Nous pouvons cependant constater que lorsque nous utilisons 4000 Exemples de référence, les résultats du système de

détection sont meilleurs avec 5 neurones cachés qu'avec 10. Au delà de 15 neurones cachés, les résultats restent sensiblement les mêmes, démontrant ainsi l'efficacité de l'algorithme d'Online BootStrapping pour éviter le sur-apprentissage.

Ainsi, si nous utilisons suffisamment de neurones cachés, les résultats sont alors limités par d'autres facteurs comme les traitements appliqués à l'image en Niveaux de Gris, et surtout le nombre d'exemples de la base de référence. A l'inverse, si le MLP est sous-dimensionné, le système d'apprentissage ne peut converger vers une solution qui permet d'obtenir à la fois un Rappel et une Précision élevés.



(a) 1000 exemples (b) 4000 exemples

FIGURE 4.2 – Courbes Rappel Précision du système de détection basé sur un MLP en fonction du nombre de neurones cachés. Que ce soit pour 1000 ou 4000 exemples nous constatons une amélioration puis une stagnation des taux de détection avec l'augmentation du nombre n_2 de neurones cachés.

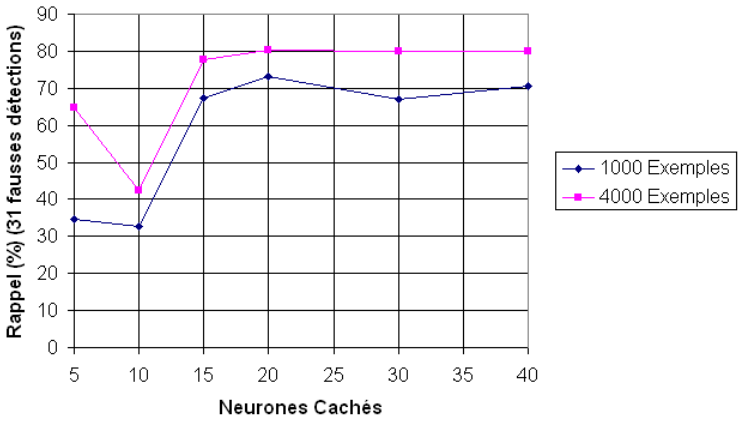


FIGURE 4.3 – Rappel du système de détection basé sur un MLP pour 31 fausses détections sur la base CMU en fonction du nombre de neurones cachés.

4.2.2.3 Influence du nombre d'images exemples

Nous avons pu constater grâce aux expérimentations précédentes que si nous dimensionnons convenablement le MLP, alors c'est le nombre d'exemples d'apprentissage qui limite la qualité de notre système de détection. Afin de mesurer les effets du nombres d'exemples d'apprentissage, nous avons appliqué le système de détection basé sur un MLP à 40 neurones cachés, à la base de test CMU, en faisant varier le nombre d'exemples d'apprentissage entre 80 et 5000. Les courbes Rappel Précision correspondantes sont visibles sur la figure 4.4. Nous pouvons constater une amélioration continue des résultats avec l'augmentation du nombre d'exemples. Afin de montrer la progression des résultats, nous avons reporté sur la figure 4.5 le Rappel du système de détection pour 31 fausses détections en fonction de la dimension de la base d'exemples de référence. Si 80 exemples ne permettent d'atteindre qu'un Rappel de 47% pour 31 fausses détections, ce qui est comparable aux résultats obtenus par le système de détection basé sur la corrélation, l'augmentation du nombre d'exemples améliore très rapidement les résultats, puisque nous atteignons un Rappel de 64% pour 300 exemples et de plus de 80% lorsque nous dépassons les 3000 exemples. Ainsi, malgré le fait que nous n'utilisons aucun traitement d'image complexe, nous constatons que notre système obtient des résultats qui se rapprochent de l'état de l'art lorsque nous utilisons un grand nombre d'exemples d'apprentissage. En effet, le système de Rowley *et al* obtient un Rappel de 86% et celui de Viola-Jones 85% pour 31 fausses détections sur la base CMU, contre un Rappel de 82% pour notre système de détection. La plupart des systèmes de l'état de l'art utilisent environ 4000 exemples d'apprentissage afin d'entraîner un classifieur discriminatif, ces résultats nous montrent que les performances des systèmes de détection sont principalement dues au nombre d'exemples et au type de classifieur utilisé.

4.2.2.4 Influence de la forme du MLP

Nous avons jusqu'à présent utilisé un MLP à trois couches entièrement connecté. Rowley *et al* dans [47] obtiennent des résultats au niveaux de l'état de l'art en détection de visages avec un MLP à trois couches partiellement connecté. La forme du réseau de neurones est décrite section 2.3.7.1 et figure 4.6. Cette forme de réseau de neurones permet de réduire le nombre de connections et donc de diminuer à la fois les temps d'apprentissage et de détection. Ainsi, là ou un MLP entièrement connecté à 26 neurones cachés compte 16276 connections, le MLP partiellement connecté de Rowley en compte 2078 pour le même nombre de neurones cachés et une rétine de 25×25 pixels, c'est à dire, près de huit fois moins de connections. Dans cette section, nous utilisons la forme de MLP de Rowley *et al* et comparons les résultats à ceux obtenus avec un MLP entièrement connecté.

Dans [47] les neurones cachés sont copiés deux, voire trois fois afin d'augmenter le nombre de paramètres du réseau de neurones. Nous obtenons alors trois réseaux de neurones distincts avec 26, 52 et 78 neurones cachés que nous nommerons respectivement Rowley1, Rowley2 et Rowley3. En premier lieu, nous avons comparé les résultats du système de détection utilisant le réseau de neurones de Rowley avec

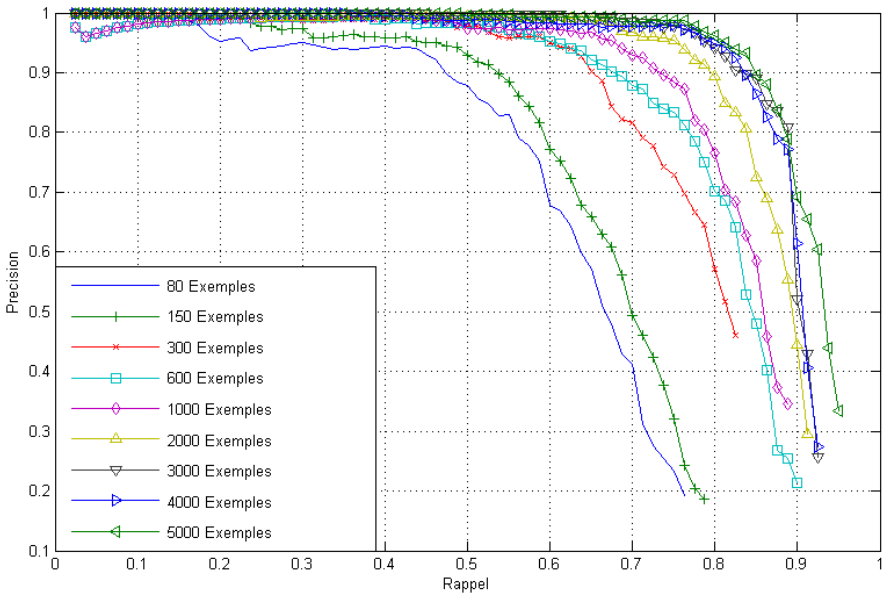


FIGURE 4.4 – Courbes Rappel Précision du système de détection basé sur un MLP entièrement connecté à 40 neurones cachés en fonction du nombre d'exemples de la base de référence. Plus nous utilisons d'exemples, meilleurs sont les taux de détection.

150, 1000 et 5000 exemples d'apprentissage et ceci pour 26, 52 et 78 neurones cachés. Nous pouvons constater (figure : 4.7) que les meilleurs résultats sont obtenus par les réseaux Rowley2 et Rowley3 qui comptent un total de $2078 \times 2 = 4156$ et $2078 \times 3 = 6234$ connections. Si l'utilisation de 78 neurones cachés semble donner de meilleurs résultats, l'utilisation du réseau Rowley2 semble un bon compromis entre la complexité du réseau de neurones et les résultats du système de détection. En effet, nous obtenons avec un MLP possédant 4156 connections des résultats équivalents à un MLP à 40 neurones cachés comptant 25040 connections. Ainsi, nous en déduisons que c'est l'information locale qui est la plus pertinente pour distinguer un visage du 'reste du monde'.

En ce qui concerne le nombre de fichiers nécessaire à la convergence de l'algorithme ($NbIm$), il oscille entre 100 et 200 et est donc au moins deux fois inférieur au nombre nécessaire pour un MLP entièrement connecté. Cependant, comme pour un MLP entièrement connecté, $NbIm$ varie de façon assez aléatoire et il est difficile de conclure sur l'influence du nombre d'exemples ou du nombre de connections.

Dans la suite des expérimentations, nous utiliserons le MLP partiellement connecté Rowley2, qui est un bon compromis entre les résultats et les temps de calculs nécessaires à l'apprentissage et à l'évaluation du système de détection.

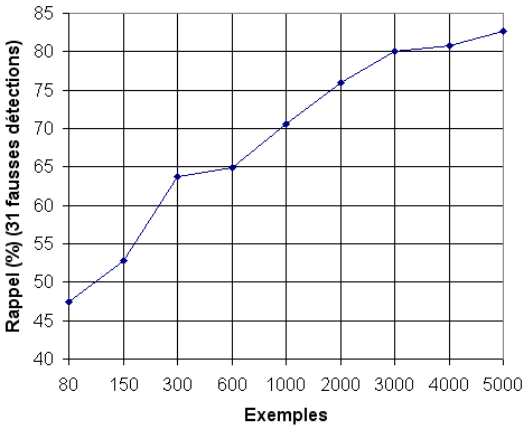


FIGURE 4.5 – Rappel du système de détection basé sur un MLP pour 31 fausses détections sur la base CMU en fonction du nombre d'exemples de la base de référence. On constate une augmentation du Rappel avec le nombre d'exemples.

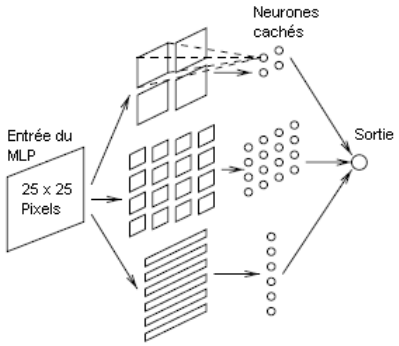
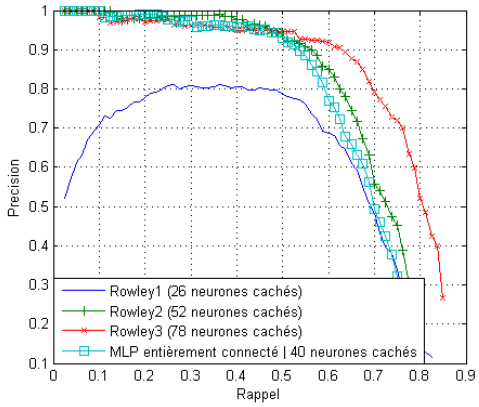
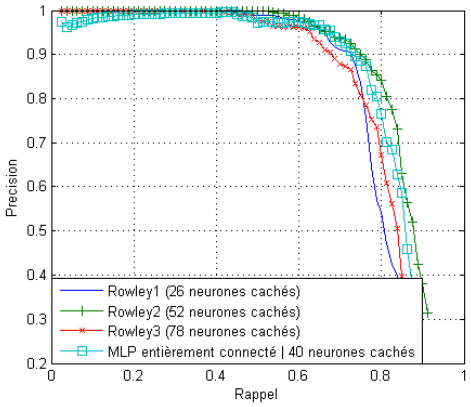


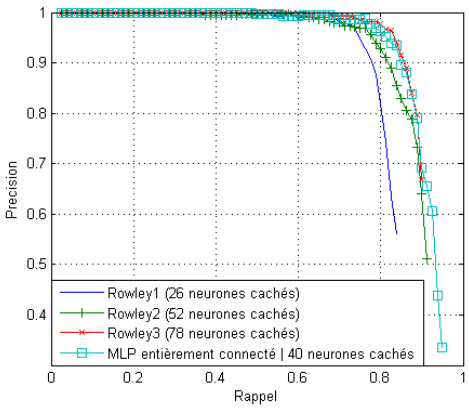
FIGURE 4.6 – Forme du MLP partiellement connecté utilisé, inspiré par Rowley *et al.*



(a) 150 exemples



(b) 1000 exemples



(c) 5000 exemples

FIGURE 4.7 – Influence du nombre de neurones cachés sur le réseau de neurones de Rowley *et al* avec 150 , 1000 et 5000 exemples d'apprentissage. Comparaison des résultats sur la base CMU par rapport à un MLP entièrement connecté à 40 neurones cachés. Nous pouvons constater que le réseau de neurone de Rowley est capable d'obtenir des résultats équivalents voire supérieurs a ceux du MLP entièrement connecté tout en possédant bien moins de connections.

4.3 C-PCA et association de MLP

L'utilisation d'un système d'apprentissage performant basé sur un MLP et la méthode d'Online BootStrapping permettent d'obtenir des résultats qui se rapprochent de l'état de l'art, à condition d'utiliser plusieurs milliers d'exemples. Ainsi, un tel système de détection, s'il s'avère bien plus performant que notre système basé sur la corrélation, nécessite l'utilisation de cinquante fois plus d'exemples d'apprentissage. Si nous utilisons 80 exemples, alors nous obtenons des résultats comparables à ceux obtenus par notre système basé sur l'association de corrélations. Le but de cette section est d'appliquer les méthodes employées avec la détection par corrélation au système de détection utilisant un MLP. Nous souhaitons ainsi obtenir un système capable d'atteindre des taux de détection au niveau de l'état de l'art mais utilisant une base d'apprentissage réduite. Nous avons constaté dans le chapitre précédent que l'association de plusieurs corrélations et de traitements d'image distincts permettait une nette amélioration des taux de détection. Dans cette section, nous décrivons de quelle manière nous avons transposé la méthode d'association de corrélations à l'association de MLP et en particulier comment nous avons adapté l'algorithme d'apprentissage à l'utilisation d'une association de MLP.

4.3.1 méthodes d'association des MLP

Nous avons dénombré trois méthodes simples d'association des classifieurs : la méthode 'ET', la méthode 'OU' et enfin la méthode 'vote'. Ces trois méthodes sont utilisées et comparées dans le système de détection de Rowley *et al.* En effet, afin de diminuer le nombre de fausses détections, ce système entraîne plusieurs MLP dont les résultats sont arbitrés de la façon suivante.

1. La méthode 'ET' consiste à ne garder que les résultats pour lesquels chacun des MLP donne un score supérieur à 0. Autrement dit, un visage est détecté s'il l'est pour l'ensemble des classifieurs.
2. La méthode 'OU' consiste à garder tous les résultats pour lesquels au moins un des classifieurs donne un résultat supérieur à 0. C'est à dire, garder toutes les détections qu'au moins un classifieur classe dans la catégorie 'visage'.
3. La méthode du 'vote' consiste à utiliser trois classifieurs et à classer une image comme 'visage', si au moins deux des trois classifieurs classent l'image dans la catégorie 'visage'. Cette méthode peu être considérée comme une solution intermédiaire entre le 'ET' et le 'OU'.

Dans le cas du système de Rowley *et al*, l'entrée de chaque classifieur ainsi que les exemples de visages utilisés sont les mêmes. Chaque classifieur est entraîné indépendamment par une méthode de BootStrapping. Ainsi, les résultats de chaque classifieur ne se distinguent que par les variations des conditions initiales de chacun des MLP. Malgré cela, l'association de classifieurs permet une nette diminution du nombre des fausses détections, particulièrement pour les méthodes 'ET' et 'vote'. Un seul classifieur permet d'obtenir un taux de détection de $91,7\%$ mais pour un total de

484 fausses détections avec la base de test CMU. Deux classifieurs similaires associés selon la méthode 'ET' permettent d'obtenir un Rappel de $86,6\%$ avec 79 fausses détections. La méthode de 'vote' avec trois classifieurs permet d'atteindre $88,4\%$ avec 99 fausses détections. La méthode 'OU' semble moins efficace et atteint un Rappel assez élevé de $90,3\%$ mais pour un nombre assez important de 185 fausses détections.

Dans le chapitre précédent, nous avons associé les résultats de plusieurs corrélations selon une méthode basée sur le 'ET'. En effet, pour qu'une détection soit valide, il est nécessaire que chaque corrélation de l'association renvoie un score supérieur à un seuil s_f (section 3.6.1). Nous avons fait ce choix pour plusieurs raisons :

- La première est que la probabilité que la rétine testée dans une image ne soit pas un objet est 'infiniment' plus grande que le contraire. Ainsi, un système de détection doit favoriser en priorité l'élimination des fausses détections. L'association selon la méthode 'ET' permet de limiter au maximum ces fausses détections puisque, si une seule corrélation de l'association renvoie un score indiquant que la rétine testée n'appartient pas à la catégorie 'objet', le système considère que l'image testée appartient alors à la catégorie 'non objet'.
- La seconde est liée à la complexité de calcul. Comme nous l'avons vu (section 3.6.1) l'association par la méthode 'ET' permet de limiter au minimum le nombre de corrélations nécessaires à la détection. En effet, si une seule des corrélations de l'association renvoie un score inférieur au seuil s_f, il est inutile d'effectuer les autres corrélations de l'association.
- La troisième est liée à l'état de l'art. En effet, le système de détection d'objets le plus couramment utilisé actuellement, c'est à dire le détecteur de Viola-Jones [131] est basé sur une cascade de classifieurs dont le principe revient à associer une multitude de classifieurs simples selon la méthode 'ET' (un objet est détecté s'il est classé dans la catégorie 'objet' par l'ensemble des classifieurs de la cascade).

Dans cette section, nous nous inspirons du système de détection mis au point pour la corrélation pour adapter les méthodes d'association 'ET', 'OU' et 'vote' à notre système de détection utilisant un MLP. Le fonctionnement du système de détection utilisant l'association de MLP est décrit par la figure 4.8. Afin que chaque MLP de l'association utilise une information différente, l'image en Niveaux de Gris correspondant à la rétine dans l'image test est traitée de manière différente pour chaque MLP, puis normalisée. Les scores s_i de sortie de chaque MLP sont associés de façon à renvoyer un score s permettant de classer l'image correspondant à la rétine dans la catégorie appropriée. Plus le score s sera proche de 1 plus l'image correspondante aura de chance d'appartenir à la catégorie 'objet', à l'inverse, plus le score s'approche de -1 et plus l'image traitée par l'association de MLP aura de chance d'appartenir à la catégorie 'non objet'. Contrairement au système de détection de Rowley *et al*, nous souhaitons obtenir, en sortie de l'association de classifieurs, un score de détection et non simplement l'appartenance à une des deux classe considérées. Pour ce faire, les méthodes 'ET', 'OU', et 'vote' sont adaptées comme suit :

1. La méthode 'ET' : le score s de classification correspond au score le plus faible

de l'ensemble des classifieurs, $s = \min(s_i)$.

2. La méthode 'OU' : le score s de classification correspond au score le plus élevé de l'ensemble des classifieurs, $s = \max(s_i)$.
3. La méthode 'vote' que nous nommerons '+' : le score s correspond à la valeur moyenne des scores s_i de l'ensemble des classifieurs, $s = \frac{1}{M}\sum_{i=1}^{M} s_i$. Cette méthode est un intermédiaire entre le 'ET' et le 'OU'.

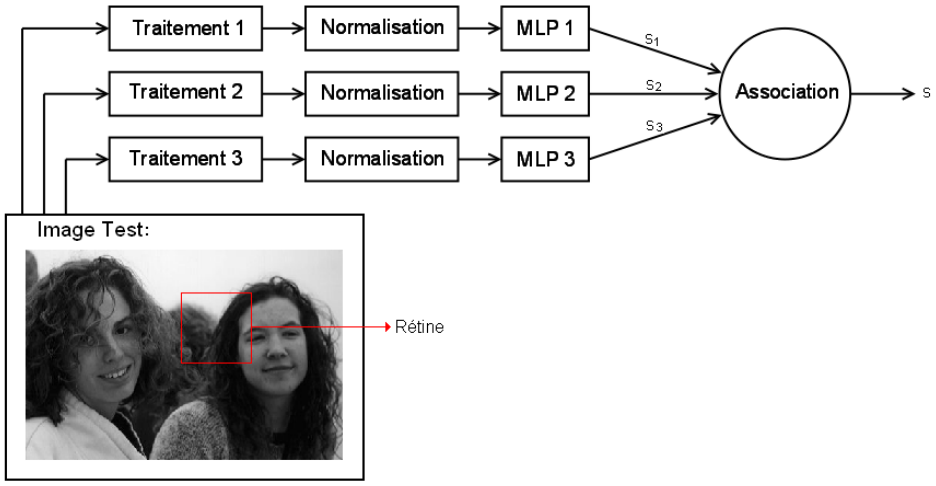


FIGURE 4.8 – Système de détection par association de MLP (exemple avec trois MLP). La rétine extraite de l'image de test est traitée selon trois méthodes donnant trois images distinctes qui sont ensuite normalisées. Le traitement d'image associé au MLP constitue alors une mesure de similarité dont le score correspondant est noté s_i. Les scores de similarité sont ensuite associés de façon à obtenir un score final s permettant de classer au mieux la rétine extraite de l'image test.

4.3.1.1 BootStrapping appliqué à l'association de MLP

Nous avons vu dans la section précédente, de quelle manière nous associons plusieurs MLP. Nous avons cependant passé sous silence la méthode utilisée pour entraîner les MLP de l'association. Nous proposons dans cette section, de décrire la méthode d'apprentissage de l'association de MLP. La méthode utilisée est une simple variation de l'algorithme d'Online BootStrapping (section : 4.2.1.1), ou plus exactement une adaptation de cette méthode à l'association de classifieurs. Elle permet d'effectuer l'apprentissage pour l'ensemble des MLP et non indépendamment pour chaque MLP. Pour ce faire, nous modifions l'algorithme d'Online BootStrapping comme suit :
N_{ftmax} : nombre maximum d'échantillons de 'non visage'.
N_{ft} : nombre total d'échantillons de 'non visage' utilisés pour l'apprentissage.

1. Réunir un petit nombre d'échantillons exemples de la classe 'non visage' et un ensemble d'échantillons représentatif de la classe 'visage'

2. Entraîner chacun des M MLP à partir de la base d'échantillons exemples courante par l'algorithme de Backpropagation.

3. Pour chacun des M MLP, supprimer les échantillons de 'non visage' dont le score de classification s_i est le plus proche de '-1' de façon à ce que $N_{ft} \leq N_{ftmax}$ pour chaque MLP.

4. Utiliser le système de détection sur une base d'images ne contenant pas la classe à détecter. Obtenir N_{fdmax} fausses détections supplémentaires.

5. Retourner à l'étape 2 jusqu'à ce que $FAR = \frac{N_{fd}}{Ntest} < FAR_{min}$

Ainsi, chaque MLP est entraîné de la même façon que lorsque nous n'en utilisons qu'un. Cependant, les fausses détections sélectionnées correspondent aux fausses détections obtenues par l'association de classifieurs. De cette manière, le système se focalise sur les erreurs commises par l'association de classifieurs et non sur les erreurs commises par chaque MLP de l'association. De plus, chaque MLP conserve les N_{ftmax} échantillons de 'non visage' qu'il a le plus de difficultés à classer dans la catégorie adéquate. Ainsi, si les échantillons 'visage' sont les mêmes pour chaque MLP (à l'exception du traitement appliqué aux images), ce n'est pas le cas des échantillons de 'non visage'. De cette manière, chaque MLP se focalise sur les échantillons de 'non visage' qui entraînent des erreurs de classification.

4.3.1.2 Utilisation de la C-PCA

Afin que chaque MLP utilise une information différente, chaque imagette présentée à chaque MLP subit un traitement d'image distinct. Nous avons vu dans le chapitre concernant la corrélation que la C-PCA permet de déterminer des filtres adaptés aux images à traiter. Ces filtres, nous ont permis d'obtenir une très nette amélioration des résultats par l'association des corrélations normées centrées des images filtrées par la C-PCA. Comme dans la section 3.6.1, nous proposons d'utiliser les filtres de la C-PCA afin d'effectuer les traitements d'image appliqués aux imagettes d'entrée de chaque MLP, excepté pour le premier MLP qui utilise la totalité de l'information disponible, c'est à dire l'image en Niveaux de Gris.

Ainsi le 'traitement 1' consiste à garder l'image en Niveaux de Gris, le 'traitement 2' effectue une convolution entre le second filtre de la C-PCA et l'image en Niveaux de Gris, le 'traitement 3' effectue une convolution avec le troisième filtre de la C-PCA et ainsi de suite en fonction du nombre M de MLP.

Les filtres de la C-PCA de dimension 5×5 ayant montré de bons résultats avec la corrélation, nous conserverons cette dimension pour notre système basé sur les MLP. La figure 4.9 montre le résultat des traitements d'image correspondant aux trois premiers MLP appliqués à un visage en Niveaux de Gris. Nous pouvons constater que chacune des trois images obtenues est visuellement assez différente et permet de focaliser chaque MLP sur des informations distinctes.

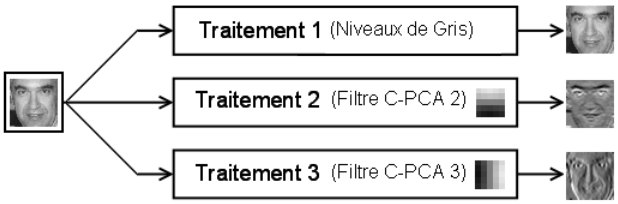


FIGURE 4.9 – Exemples de traitement d'image utilisés pour le système de détection par l'association de MLP utilisant la C-PCA. La première image correspond à l'image en Niveaux de Gris, la seconde à l'image filtrée par le second filtre de la C-PCA et la troisième par le troisième filtre de la C-PCA.

4.3.2 Résultats expérimentaux

Nous proposons dans cette section, de mesurer l'efficacité et l'utilité du système de détection par association de MLP tel que nous l'avons construit et en particulier, les performances de ce système avec une base d'exemples d'apprentissage réduite. Pour ce faire, nous utiliserons toujours la base de test CMU et les exemples tirés de la base 'Caltech WebFaces'. Le MLP partiellement connecté 'Rowley2' comportant 52 neurones cachés ayant montré les meilleurs compromis performance temps de calculs, nous utiliserons toujours ce dernier dans les expérimentations suivantes, excepté si nous précisons le contraire.

Nous commencerons par mesurer l'influence de l'utilisation des filtres de la C-PCA sur le système de détection. Nous observerons ensuite l'influence de la méthode d'association utilisée ('ET', 'OU', '+') ainsi que du nombre de MLP. Finalement, nous utiliserons la méthode d'association la plus performante et mesurerons l'influence du nombre N d'exemples d'apprentissage. Nous comparerons ensuite ces résultats à ceux obtenus avec un seul MLP puis avec l'état de l'art.

4.3.2.1 Influence des filtres de la C-PCA

Nous avons, jusqu'à présent, mesuré l'efficacité du système de détection utilisant l'image en Niveaux de Gris normalisée comme 'entrée' d'un MLP. Avant de s'intéresser aux résultats de l'association de MLP utilisant les images traitées par convolution avec les filtres de la C-PCA, nous avons appliqué chaque mesure de similarité de notre système à la base de test CMU afin de pouvoir convenablement mesurer l'influence de l'association des MLP. Comme nous souhaitons que notre système soit fonctionnel avec peu d'exemples, nous effectuons ces mesures avec 150 exemples d'apprentissage. La figure 4.10 montre les courbes Rappel Précision obtenues avec les seconds, troisièmes et quatrièmes filtres de la CPA correspondant respectivement aux 'traitements 2, 3 et 4'. Nous avons ajouté les résultats avec l'image en Niveaux de Gris correspondant au 'traitement 1' comme point de comparaison. Afin de donner des résultats chiffrés, nous donnons aussi dans le tableau 4.2, le Rappel du système de détection en fonction du traitement d'image appliqué pour 31 et 65 fausses détections.

Le 'traitement 3' semble donner les meilleurs résultats alors que les taux de détection diminuent légèrement avec l'utilisation des filtres 2 et surtout 4 de la C-PCA. Cependant, les résultats obtenus restent du même ordre, quel que soit le traitement appliqué ; nous pouvons ainsi espérer que l'association de ces traitements d'image permettra une amélioration des résultats, aucun des traitements utilisés ne risquant de faire chuter les taux de détection. De même, lorsque l'on observe le nombre de fichiers $NbIm$ nécessaire à la convergence de l'algorithme de BootStrapping, il reste du même ordre quel que soit le traitement appliqué, c'est à dire entre 100 et 200.

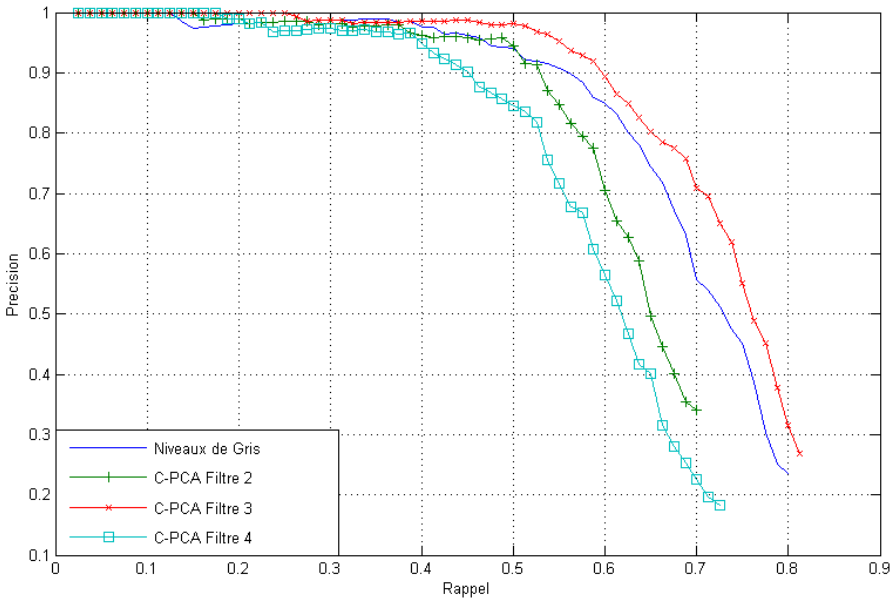


FIGURE 4.10 – Courbes Rappel Précision du système de détection basé sur le MLP Rowley2. Comparaison des résultats en fonction du traitement d'image appliqué aux images classées par le MLP. Nous pouvons constater que les résultats restent comparables, que l'on utilise l'image en Niveaux de Gris, ou les filtres de la C-PCA

	Traitement			
fausses détections	1	2	3	4
31	55,0%	51,8%	58,1	44,7%
65	60,1%	55,4%	62,3%	51,4%

TABLE 4.2 – Comparaison du Rappel du système de détection basé sur un MLP en fonction du traitement d'image appliquée pour 31 et 65 fausses détections sur la base de test CMU.

4.3.2.2 Influence de la méthode d'association et du nombre de MLP

Comme nous l'avons vu section 4.3.1, nous pouvons associer les mesures de similarité selon trois différentes méthodes, *i.e*, 'ET', 'OU' et '+'. De plus, nous pouvons varier le nombre de mesures de similarité utilisées. Le nombre de combinaisons étant alors assez important, nous commencerons par observer l'influence du nombre M de mesures de similarité pour chacune des trois associations. Une fois déterminé le nombre M optimal, nous comparerons alors les résultats de chacune des trois associations. Nous utilisons toujours 150 exemples d'apprentissages et évaluons notre système sur la base de test CMU.

Méthode 'ET' : La figure 4.11 présente les courbes Rappel Précision pour une seule mesure de similarité utilisant l'image en Niveaux de Gris ($M = 1$), puis pour deux mesures combinant l'image en Niveaux de Gris et le second filtre de la C-PCA ($M = 2$) jusqu'à combiner quatre mesures de similarité ($M = 4$) associant ainsi, l'image en Niveaux de Gris et trois filtres de la C-PCA par la méthode 'ET'. Nous pouvons, en premier lieu, constater que l'ajout d'une mesure de similarité supplémentaire ne détériore pas les résultats et permet à partir de $M = 3$ une sensible amélioration des résultats. Le tableau 4.3 reporte le Rappel pour 31 et 65 fausses détections en fonction de M et montre que l'association de trois mesures de similarité permet, pour un même nombre de fausses détections, une augmentation du Rappel de l'ordre de 5% par rapport à l'utilisation d'une seule mesure de similarité.

Si le gain de performances apporté par l'association 'ET' est relativement faible, le nombre d'images nécessaire $NbIm$ à la convergence de l'algorithme de BootStrapping n'est que de 25 pour $M \geq 2$ contre 137 pour $M = 1$. Ceci permet une diminution appréciable de la durée de la phase d'apprentissage.

	nombre d'associations M			
fausses détections	1	2	3	4
31	$55,0\%$	$53,0\%$	$61,3$	$61,5\%$
65	$60,1\%$	$60,5\%$	$65,7\%$	$65,5\%$

TABLE 4.3 – Comparaison du Rappel du système de détection basé sur les MLP en fonction du nombre M de mesures de similarité utilisées avec la méthode d'association 'ET' pour 31 et 65 fausses détections sur la base de test CMU.

Méthode 'OU' : Comme pour la méthode 'ET', nous reportons les résultats obtenus en fonction de M, sous forme de courbes Rappel Précision (figure : 4.12) et de tableau (tableau : 4.4). Comme nous le pensions, la méthode 'OU' améliore le Rappel au détriment de la Précision. En effet, nous pouvons constater sur la figure 4.12 que la Précision du système de détection pour un Rappel inférieur à 0.5 a tendance à légèrement diminuer avec l'augmentation du nombre M d'associations. A l'inverse, nous constatons que le gain sur le Rappel apporté par l'association 'OU' est d'autant plus important que la Précision diminue. Ainsi, pour $M = 3$ le Rappel est augmenté

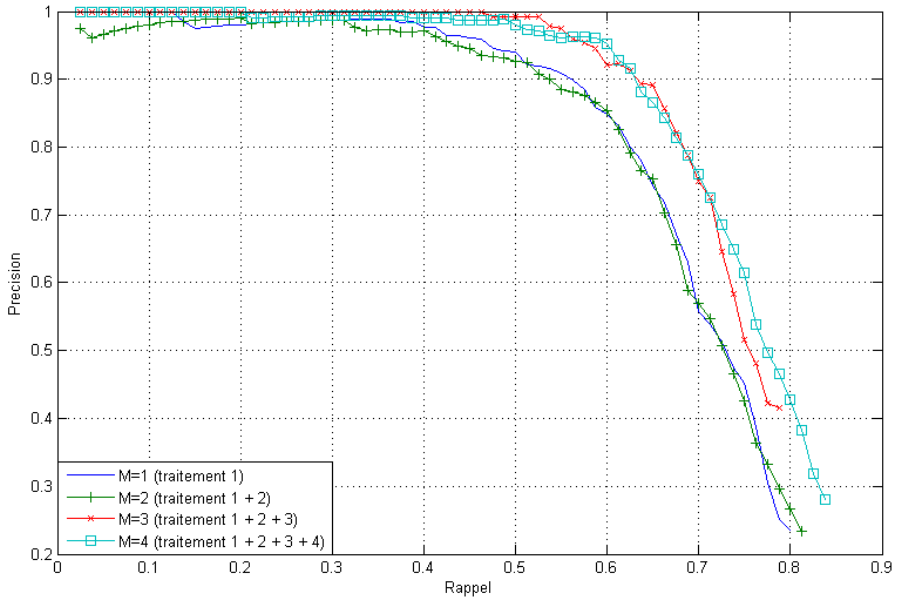


FIGURE 4.11 – Courbes Rappel Précision du système de détection basé sur l'association de MLP 'ET' et la C-PCA. Comparaison des résultats en fonction de M, le nombre de mesures de similarité utilisées.

de 10% pour 31 fausses détections et de 15% pour 65. Comme pour l'association 'ET' l'utilisation d'une quatrième mesure de similarité ne semble pas permettre de réellement améliorer les résultats. Ainsi, l'utilisation de $M = 3$ associations semble le meilleur compromis entre la complexité du système et les performances.

Le nombre de fichiers images $NbIm$ nécessaire à l'apprentissage est lui nettement augmenté dès que $M \geq 2$ puisqu'il est de l'ordre de 450 images, contre 137 pour $M = 1$ et 25 en utilisant la méthode 'ET'.

fausses détections	nombre d'associations M			
	1	2	3	4
31	55,0%	63,9%	65,5	60,3%
65	60,1%	69,4%	72,2%	70,0%

TABLE 4.4 – Comparaison du Rappel du système de détection basé sur les MLP en fonction du nombre M de mesures de similarité utilisées avec la méthode d'association 'OU' pour 31 et 65 fausses détections sur la base de test CMU.

Méthode '+' : Cette méthode constitue un compromis entre la méthode 'ET' qui favorise la Précision du système en minimisant le nombre de fausses détections et

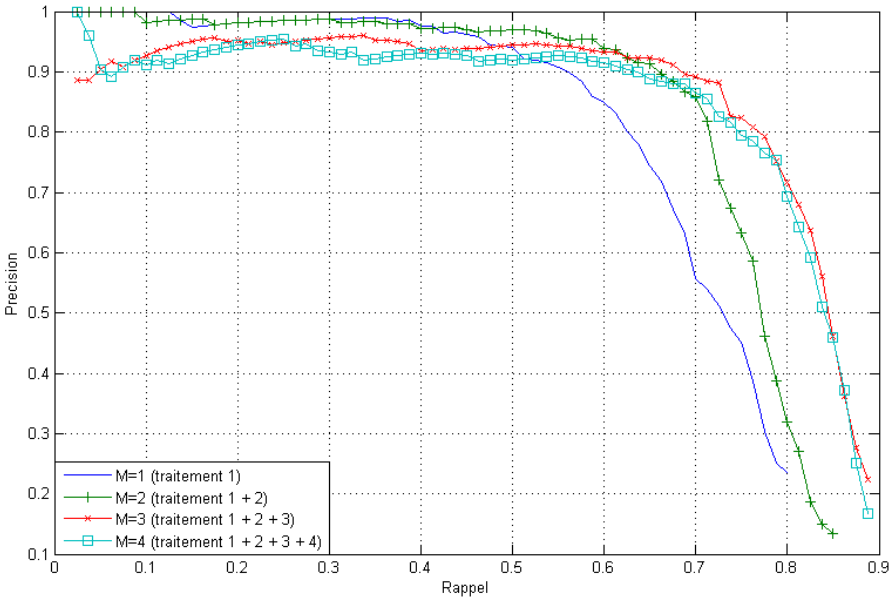


FIGURE 4.12 – Courbes Rappel Précision du système de détection basé sur l'association de MLP 'OU' et la C-PCA. Comparaison des résultats en fonction de M, le nombre de mesures de similarité utilisées.

la méthode 'OU' qui au contraire, favorise l'augmentation du Rappel du système de détection. Comme pour les deux autres méthodes, les résultats obtenus sur la base CMU sont reportés sous forme de courbes Rappel Précision (figure : 4.13) et sous forme de tableau (tableau : 4.5). Nous pouvons constater que l'association '+' permet une nette amélioration des taux de détection avec un gain de l'ordre de 20% du Rappel pour $M = 3$ et 31 et 65 fausses détections. Comme pour les méthodes d'association précédentes, l'augmentation du nombre d'associations M au delà de 3 ne semble pas apporter un gain de performance le justifiant. Ainsi, nous remarquons que pour les trois méthodes d'association ainsi que pour la méthode de détection par association de corrélations (section 3.6.1), l'association de trois mesures de similarité est le meilleur compromis entre les taux de détection du système et les temps de calculs.

La méthode d'association '+' n'apporte par contre aucun gain particulier sur la convergence de l'algorithme de BootStrapping, $NbIm$ restant de l'ordre de 150 quel que soit la valeur de M.

Comparaison des méthodes 'ET', 'OU' et '+' : Les expérimentations précédentes nous ont permis de déterminer que le meilleur nombre d'associations est $M = 3$ pour les trois méthodes utilisées. Afin de déterminer la méthode d'association la plus performante nous reportons (figure : 4.14) les courbes Rappel Précision des trois méthodes 'ET', 'OU' et '+' pour $M = 3$, nous avons en plus reporté la

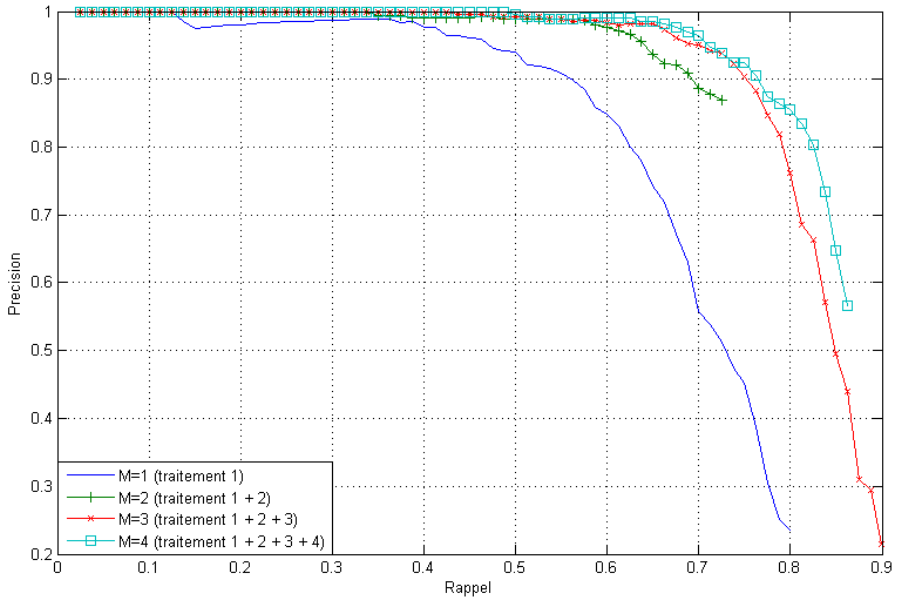


FIGURE 4.13 – Courbes Rappel Précision du système de détection basé sur l'association de MLP '+' et la C-PCA. Comparaison des résultats en fonction de M, le nombre de mesures de similarité utilisées.

	nombre d'associations M			
fausses détections	1	2	3	4
31	55,0%	66,7%	72,6	73,8%
65	60,1%	73,0%	76,2%	78,6%

TABLE 4.5 – Comparaison du Rappel du système de détection basé sur les MLP en fonction du nombre M de mesures de similarité utilisées avec la méthode d'association '+' pour 31 et 65 fausses détections sur la base de test CMU.

courbe Rappel Précision correspondant au système de détection sans association afin de mieux visualiser le gain apporté par la méthode d'association de MLP utilisant la C-PCA. Nous pouvons aisément constater que, quelle que soit la méthode d'association utilisée, nous obtenons un gain substantiel des taux de détection. De plus, nous voyons que la méthode '+' permet de tirer partie des avantages de la méthode 'ET' qui favorise la Précision et de la méthode 'OU' qui favorise le Rappel. Ainsi, la méthode d'association '+' de trois MLP partiellement connectés (Rowley2) permet d'obtenir un système de détection efficace avec seulement 150 visages exemples contre de l'ordre de 4000 pour l'état de l'art.

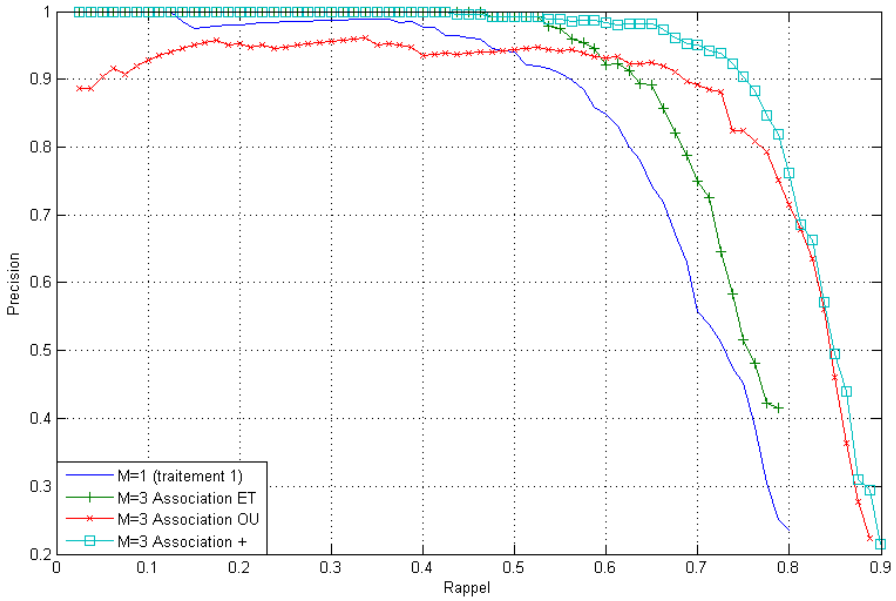


FIGURE 4.14 – Courbes Rappel Précision du système de détection basé sur l'association de MLP et la C-PCA. Comparaison des résultats en fonction de la méthode d'association utilisée.

4.3.2.3 Influence du nombre d'exemples

Les expérimentations précédentes nous ont permis de mettre au point un système de détection capable de détecter des visages dans une base complexe avec une base d'apprentissage réduite à seulement 150 exemples. Ce système consiste à utiliser une association '+' de trois MLP partiellement connectés (Rowley2) avec l'image en Niveaux de Gris et les filtres 2 et 3 de la C-PCA. Cette méthode permet, à partir de seulement 150 exemples d'apprentissage, d'atteindre un Rappel de $72,6\%$ pour 31 fausses détections. Ces résultats restent néanmoins assez éloignés des taux de détection de l'état de l'art qui sont au moins de 85% pour le même nombre de fausses détections. Afin de tester dans quelle mesure notre système est sensible aux variations du nombre N d'exemples d'apprentissage et combien d'exemples sont nécessaires pour obtenir des résultats comparables à l'état de l'art, nous avons évalué notre système sur la base de test CMU avec $N = 80$, 150, 300, 600 et 1000 exemples. Les courbes Rappel Précision sont données figure 4.15. Nous pouvons constater une amélioration rapide des résultats avec l'augmentation du nombre d'exemples jusqu'à 600, puis une stagnation, voire une très légère régression avec 1000 exemples d'apprentissage. Ceci s'explique par le fait que nous avons dimensionné notre système pour fonctionner avec peu d'exemples.

La figure 4.16 compare le Rappel obtenu par l'association de MLP pour 31 fausses détections à celui obtenu sans association (section : 4.2.2.3). Nous remarquons un

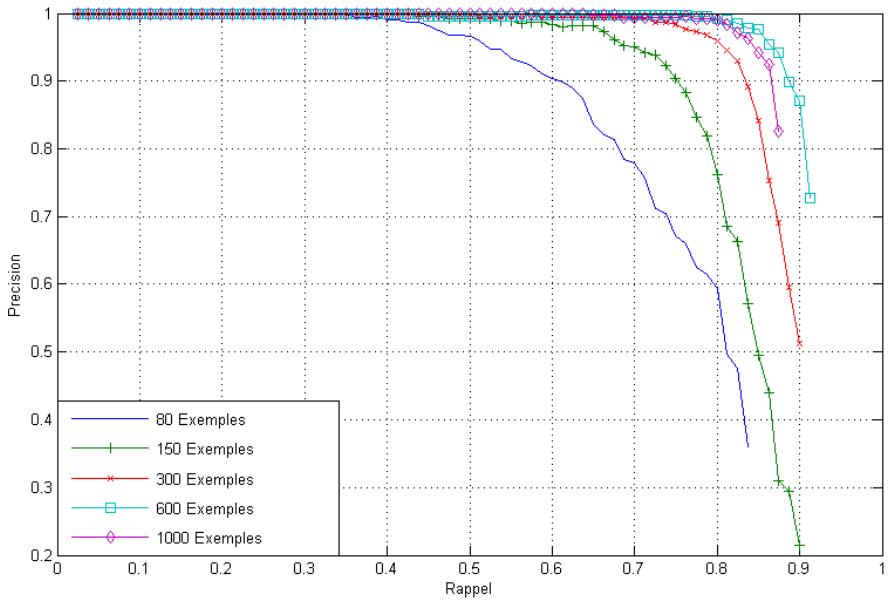


FIGURE 4.15 – Courbes Rappel Précision du système de détection basé sur l'association '+' de MLP et la C-PCA. Comparaison des résultats en fonction du nombre d'exemples d'apprentissage.

écart de prêt de 20% lorsque nous utilisons moins de 1000 exemples d'apprentissage. Nous pouvons aussi constater que l'association de classifieurs permet d'atteindre avec 300 exemples les même résultats qu'un seul MLP avec 5000 exemples (figure : 4.4 et 4.5). Nous démontrons ainsi que l'utilisation de filtres détecteurs de contours et d'associations de classifieurs permet, comme pour la corrélation, une très nette amélioration des résultats lorsque nous utilisons peu d'exemples.

Enfin, si l'on se compare à l'état de l'art (tableau : 4.6), nous constatons que notre système atteint des performances du même ordre que le système de Viola-jones ou celui de Rowley *et al* avec seulement 600 exemples d'apprentissage contre plusieurs milliers pour ces derniers. Nous constatons de plus que les meilleurs systèmes en terme de taux de détections sont ceux utilisant le plus grand nombre d'exemples d'apprentissage.

Afin d'illustrer les résultats de notre système de détection avec 600 exemples, nous montrons figure 4.17 quelques détections effectuées sur la base CMU. Nous pouvons ainsi remarquer que notre système de détection échoue à détecter les visages très peu contrastés ou très flous mais est capable de détecter des visages subissant tout de même de grandes variations d'illumination ou de forme.

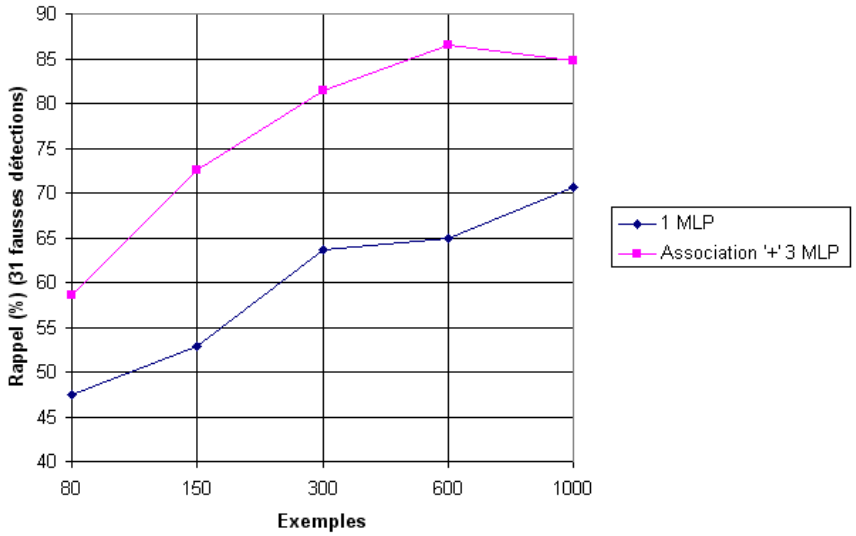


FIGURE 4.16 – Rappel du système de détection basé sur un MLP et l'association '+' de trois MLP pour 31 fausses détections sur la base CMU en fonction du nombre d'exemples de la base de référence.

Système de détection	Fausses détections				
	0	10	31	65	167
Rowley *et al* [47] (4000)	-	83,2%	86,0%	-	90,1%
Viola-Jones [131] (4916)	-	78,3%	85,2%	89,8%	91,8%
CFF [43] (3702)	88,8%	90,5%	91,5%	92,3%	93,1%
Association '+' 3 MLP (300)	49,6%	76,2%	81,4%	82,9%	85,5%
Association '+' 3 MLP (600)	43,1%	84,0%	86,5%	88,3%	89,7%

TABLE 4.6 – Rappel du système de détection basé sur l'association '+' de trois MLP et la C-PCA pour divers nombres de fausses détections sur la base de test CMU. Les nombres entre parenthèses correspondent au nombre d'exemples de la base d'apprentissage.

FIGURE 4.17 – Exemples de détections effectuées sur la base CMU par le système basé sur l'association '+' de trois MLP avec 600 exemples d'apprentissage.

CHAPITRE 5

Conclusion générale

Cette partie est consacrée à faire le point sur les développements effectués jusqu'à présent. Nous commencerons par mettre en avant les principaux enseignements apportés par les différentes expérimentations, puis nous donnerons les perspectives et les possibilités d'amélioration que nous pouvons apporter au système de détection que nous avons mis au point.

5.1 Conclusion

Dans cette thèse, a été développé, une méthode de détection d'objets pouvant fonctionner avec une base d'apprentissage de dimension réduite. Nous avons appliqué ce système à la détection de visages, ce qui nous a permis de pouvoir nous comparer à diverses méthodes de détection et de démontrer que nous pouvions obtenir des résultats du même ordre que l'état de l'art avec près de dix fois moins d'exemples. Pour arriver à ce résultat, nous avons commencé par utiliser une approche basée sur une mesure de similarité utilisant la corrélation. Ceci nous a permis de mettre au point plusieurs outils apportant une très nette amélioration au système de détection basée sur la corrélation ; mais aussi, de mesurer l'influence de plusieurs paramètres sur les méthodes de détection tels que, le nombres d'exemples, l'utilisation de corrections d'illumination ou de méthodes de déformation. Nous avons ainsi déterminé que si l'utilisation de traitements d'image tels que la déformation affine, ou l'égalisation d'histogramme apporte un gain de performance modéré, c'est surtout l'utilisation de filtres détecteurs de contours et l'association de plusieurs mesures de similarité qui apportent les meilleures améliorations de résultats au système de détection par corrélation.

Nous avons alors mis au point une méthode d'extraction de descripteurs dans une image, inspirée de la 2D-PCA et pouvant être considérée comme une généralisation de cette dernière. Cette méthode que nous avons nommée Convolutional PCA (C-PCA) nous permet de déterminer des filtres détecteurs de contours adaptés aux objets que nous souhaitons détecter. Nous avons ensuite associé les mesures de similarité basées

sur la corrélation des images filtrées par la C-PCA. Nous avons alors montré qu'un tel système était capable, malgré la simplicité de la corrélation, de détecter des visages dans des images complexes en utilisant moins d'une centaine d'images exemples.

Nous avons ensuite remplacé la corrélation par une mesure discriminative basée sur un Perceptron Multicouche, bien plus performante en détection. Nous avons d'abord montré qu'un système de détection utilisant un MLP appliqué aux images en Niveaux de Gris était capable avec suffisamment d'exemples, et en dimensionnant convenablement le réseau de neurones, d'atteindre des résultats proches de l'état de l'art en détection de visages. Nous avons ainsi démontré que les performances des systèmes de détection modernes sont en très grande partie dues à l'utilisation d'un classifieur discriminatif efficace et à l'utilisation d'un très grand nombre d'exemples d'apprentissage. Les traitements correcteurs d'illumination ainsi que les différents descripteurs d'images possibles, influent finalement beaucoup moins sur les résultats que la forme du classifieur utilisé ou le nombre d'exemples d'apprentissage.

Nous avons ensuite associé plusieurs MLP et les images filtrées par la C-PCA, dans le but de permettre à notre système de détection de fonctionner avec une base d'exemples d'apprentissage de dimension plus réduite, tout en obtenant des résultats comparables à l'état de l'art. Afin de pouvoir entraîner une association de MLP avec une base d'apprentissage réduite sans risquer de sur-apprentissage, nous avons mis au point une variante de l'algorithme de BootStrapping qui nous a permis, à la fois d'obtenir un système convergeant rapidement vers une solution, mais aussi capable de fonctionner avec des bases allant de 80 exemples à plusieurs milliers, sans aucune modification des paramètres du système. Nous avons alors montré que l'association de mesures de similarité permet, comme pour la corrélation, une très nette amélioration des résultats, surtout lorsque nous utilisons peu d'exemples. Ainsi, nous avons pu obtenir des résultats au niveaux de l'état de l'art tout en utilisant près de dix fois moins d'exemples, l'association de classifieurs montrant des améliorations de Rappel de l'ordre de 20% par rapport à un MLP seul.

Pour résumer, nous avons montré que nous pouvons obtenir un système de détection d'objets complexes avec seulement quelques centaines d'exemples, non pas en utilisant des méthodes de déformation ou des traitements d'image complexes afin de minimiser la variabilité des exemples d'apprentissage, mais en associant plusieurs classifieurs et en utilisant des filtres de contours orientés.

5.2 Perspectives

Le sujet traité, étant assez vaste et les possibilités très nombreuses, nous avons écarté certaines voies qui méritent cependant sans doute d'être explorées et qui pourraient permettre à la fois une amélioration des taux de détection, l'utilisation de moins d'exemples d'apprentissage mais aussi l'obtention d'un système de détection bien plus rapide.

En premiers lieu, il est possible de remplacer les MLP de notre système par des réseaux convolutionnels [43]. Ces derniers étant à la fois plus performants en détection et plus rapides qu'un simple MLP, il serait interessant d'observer les résultats d'une

association de CFF.

Une autre solution que nous n'avons que peu explorée, est l'utilisation de méthodes de co-apprentissages [68, 110]. Ceci consiste à utiliser les détections effectuées par une association de classifieurs dans le but d'augmenter la dimension de la base d'apprentissage et constitue, sans doute, une des meilleures perspectives pour encore réduire la dimension de la base d'apprentissage initiale. En effet, nous avons montré [90] qu'il était possible avec la détection par corrélation d'utiliser les visages détectés pour augmenter le nombre d'exemples d'apprentissage et améliorer les performances du système de détection. De plus, l'utilisation d'une association de classifieurs s'avère particulièrement adaptée au co-apprentissage.

Enfin, nous nous sommes limité dans cette étude au problème de la détection. Cependant, il serait intéressant d'appliquer les outils que nous avons mis au point à d'autres domaines, notamment, la reconnaissance de visages ou nous ne disposons souvent que d'un nombre limité d'exemples. La 2D-PCA ayant par exemple montré de bons résultats utilisés en reconnaissance de visages, il serait intéressant d'observer les performances de la C-PCA dans ce domaine.

Bibliographie

[1] O. Agazzi, S. Kuo, E. Levin, and R. Pieraccini. Connected and degraded text reconition using planar hidden markov models. *In Proc. of the IEEE Int. Conf. on Accoustics Speech and Signal Processing, ICASSP*, 5 :113–116, 1993.

[2] S.A. Aivazian, V.M. Buchstader, I.S. Yenyukov, and L.D. Meshalkin. Applied statistics : Classification and reduction of dimensionality. *Finansy i Statistika (Reference Edition)*, 1989.

[3] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, pages 821–837, 1964.

[4] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1-3) :42–54, 2000.

[5] P. Baldi and K. Hornik. Neural networks and principal component analisys : Learning from examples without local minima. *IEEE transactions on Neural Networks*, pages 53–58, 1988.

[6] S. Baluja. Face detection with in-plane rotation : Early concepts and preliminary results. *Technical Report JPRC-1997-001-1, Justsystem Pittsburg Research Center*, 1997.

[7] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *International Conference on Computer Vision*, 2 :383–390, 2001.

[8] R. Battiti. Accelerated backpropagation learning : Two optimizations methods. *Complex systems*, 3 :331–342, 1989.

[9] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs fisherfaces : Recognition using class specific linear projection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17 :711–720, 1997.

[10] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1) :1–127, 2009.

[11] D. Beyner and T. Poggio. Face recognition from one example view. *Computer Vision, Proceedings, Fifth International Conference on*, 1995.

[12] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[13] H. Boulard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, pages 291–294, 1988.

[14] Lubomir Bourdev and Jonathan Brandt. Robust object detection via soft cascade. In *CVPR '05 : Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Paul S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML '98 : Proceedings of the Fifteenth International Conference on Machine Learning*, pages 82–90, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[16] R. Brunelli and T. Poggio. Face recognition : Feature versus templates. *IEEE Trans. Pattern Anal. Mach Intel*, pages 1042–1052, 1993.

[17] G. Burel and D. Carel. Detection and localization faces on digital images. *Pattern Recognition*, pages 963–967, 1994.

[18] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, pages 121–167, 1998.

[19] F. Camastra, E. Cepollina, and A.M. Colla. Word recognition by MLP-based character spotting and dynamic programming. *9th Workshop of Italian Neural Network Society (WIRN97)*, 1997.

[20] R. Chellappa, C.L. Wilson, and S. Sirohey. Human and machine recognition of faces : A survey. *Proc. IEEE*, pages 384–388, 1997.

[21] Xue-wen Chen and Jong Cheol Jeong. Minimum reference set based feature selection for small sample classifications. In *ICML '07 : Proceedings of the 24th international conference on Machine learning*, pages 153–160, New York, NY, USA, 2007. ACM.

[22] J. Choi. Realtime on-road vehicle detection with optical flows and haar-like feature detector. *University of Illinois*, 1007.

[23] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active Appearence Models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6), 2001.

[24] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active Shape Models - their training and application. *Computer Vision and Image Understanding*, 61(1), 1995.

[25] T. F. Cootes, K. Walker, and C. J. Taylor. View-based active appearence models. *Automatic Face and Gesture Recognition, Proceedings. Fourth IEEE International Conference*, 2000.

[26] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20 :1–25, 1995.

[27] G. Cottrel and M. Fleming. Face recognition using unsupervised feature extraction. *Proceedings of the International Conference on Neural Networks*, pages 322–325, 1990.

[28] G.W. Cottrel and M.K. Fleming. Face recognition using unsupervised feature extraction. *Proc, inteligence Neural Network Conf*, pages 322–325, 1990.

[29] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Srar. Soc.*, 39 :1–38, 1977.

[30] Richard O. duda, Peter E. Hart, and David G.Stork. *Pattern Classification.* Wiley-Interscience, 2001.

[31] S. Duffner and C. Garcia. A neural scheme for robust detection of transparent logos in tv programs. *Artificial Neural Networks - ICANN*, pages 14–23, 2006.

[32] J L. Dugelay and H. Sanson. Differential methods for the identification of 2D and 3D motion models in image sequences. *Signal Processing : Image Communication 7*, 1995.

[33] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Face recognition using active appearance models. *Proceedings of the 5th European Conference on Computer Vision*, 1998.

[34] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Advances in active appearence models. *Computer Vision. The Proceedings of the Seventh IEEE International Conference on*, 1 :137–142, 1999.

[35] H.M. El-Bakry and M.A. Abo Elsoud. Human face recognition using neural networks. *The Sixteenth National Radio Science Conference, Cairo*, 1999.

[36] M.H. Ahmad Fadzil and H. Abu Bakar. Human face recognition using neural networks. *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 3 :936–939, 1994.

[37] O.D. Faugeras and G. Toscani. The calibration problem for stereo. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, CVPR-86*, pages 15–20, 1986.

[38] R. A. Fisher. The use of multiple measures in taxonomic problems. *Annals of Eugenics*, 17 :179–188, 1936.

[39] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Second International Conference on Computational Learning Theory*, 1995.

[40] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, pages 119–139, 1997.

[41] K. Fukushima. Cognitron : A self-organizing multilayered neural network. *Biological Cybernetics*, 20(6) :121–136, 1975.

[42] K. Fukushima and S. Miyake. Neocognitron : a new algorithm for pattern recognition tolerent of deformations and shifts in position. *Pattern Recognition*, 154 :455–469, 1982.

[43] C. Garcia and M. Delakis. Convolutional Face Finder, a neural architecture for fast and robust face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11) :1408–1423, 2004.

[44] B. Girod and R. Thoma. Motion-compensating interpolation from interlaced and non-interlaced grids. *Proc. SPIE Image Coding*, 594 :186–193, 1985.

[45] M.A. Grudin. On internal representations in faces recognition systems. *Pattern Recognition*, 33(7) :1161–1177, 2000.

[46] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3) :389–422, 2002.

[47] and S. Baluja H.A. Rowley and T. Kanade. Neural network-based face detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[48] and S. Baluja H.A. Rowley and T. Kanade. Rotation invariant neural network-based face detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[49] H. J. Hotellinh. Analysis of complex of statistical variables into principal components. *Journal of Educational Psychology*, 24 :417–441, 1933.

[50] Chang Huang, Haizhou Ai, Bo Wu, and Shihong Lao. Boosting nested cascade detector for multi-view face detection. *Pattern Recognition, International Conference on*, 2 :415–418, 2004.

[51] Y.L Huang and R.F Chang. MLP interpolation for digital image processing using wavelet transform. *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 3217–3220, 1999.

[52] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction, and functional architecture in cat's visual cortex. *Journal of Physiology (London)*, 160 :106–154, 1962.

[53] Ken ichi Funashi. On the approximate realization of identity mappings by three-layer neural networks. *Technical report, Toyohashi University of Technology, Department of Information and computer Sciences*, 1990.

[54] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1 :295–307, 1988.

[55] N. Japkowicz, S. Hanson, and A. Gluck. NonLinear autoassociation is not equivalent to PCA. *Neural Computation*, 12(3) :531–545, 2000.

[56] John R. Deller Jr., John G. Proakis, and John H. L. Hansen. Discrete-time processing of speech signals. *Macmillian Publishing Company*, 1993.

[57] P. Juell and R. Marsh. A hierarchical neural network for human face detection. *Pattern Recognition*, pages 781–787, 1996.

[58] M. Kirby and L. Sirovich. Application of the karhunen-loève procedure for the characterization of human faces. *Journal of Cognitive Neuroscience*, 12(1), 1990.

[59] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Proceeding of annual ACM Symposium on the Theory of Computing*, 1995.

[60] M. Kolsh and M. Turk. Robust hand detection. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 614–619, 2004.

[61] M. A. Kramer. Non linear principal component analysis using autoassociative neural networks. *AIChE Journal*, pages 233–243, 1996.

[62] S. Kuo and O. Agazzi. Keyword spotting in poorly printed documents. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 842–848, 1994.

[63] Jorg Lange, Christoph V. D. Malsburg, Rolf P. Wurtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Trans. Comput*, 42 :300–311, 1993.

[64] Steve Lawrence, C. Lee Giles, and Ah Chung Tsoi. Lessons in neural network training : Overfitting may be harder than expected. In *In Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, pages 540–545. AAAI Press, 1997.

[65] Y. LeCun. *Generalization and network design strategies, in Connectionism in Perspective*. Elsevier Science Inc., New York, NY, USA, 1989.

[66] Y. LeCun, E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch. Cognitron : A self-organizing multilayered neural network. *Disordered systems and biological organizaton*, pages 233–240, 1986.

[67] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems 2*, pages 396–404, 1990.

[68] Anat Levin, Paul Viola, and Yoav Freund. Unsupervised improvement of visual detectors using co-training. *Proceedings of the Nineth IEEE International Conference on Computer Vision (ICCV)*, 2003.

[69] S. Li, Z. Zhang, H. Shum, and H. Zhang. Floatboost learning for classification. 2002.

[70] R. Lienhart, A. Kuranov, and V. Pisarevsky. An extended set of haar-like features for rapid object detection. *IEEE International Conference on Image Processing*, 2002.

[71] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, 2002.

[72] N. Littlestone. Learning quickly when irrelevant attributes abound : A new linear threashold learning using winnow. *Machine Learning*, 9 :385–318, 1988.

[73] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using winnow. *Proceeding of fourth annual Workshop on Computational Learning Theory*, pages 147–156, 1991.

[74] Aleksander Lodwich, Yves Rangoni, and Thomas Breuel. Evaluation of robustness and performance of early stopping rules with multi layer perceptrons. *Neural Networks, IEEE - INNS - ENNS International Joint Conference on*, 0 :1877–1884, 2009.

[75] W J. MacLean and J K. Tsotsos. Fast pattern recognition using normalized grey-scale correlation in a pyramid image representation. *Machine Vision & Applications*, 2007.

[76] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7) :1769–1796, 1999.

[77] A. Mahmood. Structure-less object detection using adaboost algorithm. *Machine Vision. ICMV 2007*, pages 85–90, 2007.

[78] A. ManField and J.Wayman. Best practices in testing and reporting performance of biometric devices. 2002.

[79] J. Mao and A.K. Jain. A note on the effective number of parameters in nonlinear learning systems. *Neural Networks,1997., International Conference on*, 2 :1045–1050.

[80] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. *Proc. of the ISCA European Conf on Speech Communication and Technology (EUROSPEECH)*, pages 1895–1898, 1997.

[81] A. Martinez and M. Kak. PCA versus LDA. *IEEE Trans on Pattern Analysis and Machine Intelligence (PAMI)*, 23 :228–233, 2001.

[82] U. Matecki and V. Sperschneider. Automated feature selection for MLP networks in SAR image classification. *Image Processing and Its Applications, Sixth International Conference*, 2 :676–679, 1997.

[83] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, pages 415–446, 1909.

[84] M. L. Minsky and S. A. Papert. Perceptrons. *MIT Press, Cambridge, MA*, 1969.

[85] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(7) :696–710, 1997.

[86] M. C. Mozer. *The perception of multiple objects : a connectionist approach.* MIT Press, Cambridge, MA, USA, 1991.

[87] A. Namphol, M. Arozullah, and S. Shin. Higher order data compression with neural networks. *Proceeding of the IJCNN*, pages 155–159, 1991.

[88] A.N. Netravali and J.D. Robbins. Motion comensated television coding - part i. *Bell Systems Technol*, 58(3) :629–668, 1979.

[89] S. Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. *Proceedings of the International Conference on Artificial Neural Networks*, pages 737–745, 1991.

[90] S. Onis, H. Sanson, and C. Garcia. Iterative unsupervised object detection system. *Systems, Signals and Image Processing. IWSSIP*, pages 397–400, 2008.

[91] M. Oravec and J. Pavlovièova. Face recognition methods based on feedforward neural networks, principal component analysis and self-organizing map. *RADIOENGINEERING*, 16(1), 2007.

[92] O. Osuna, R. Freund, and F. Girosit. Training support vector machines : an application to face detection. *Computer Vision and Pattern Recognition*, pages 130–136, 2002.

[93] S. Palanivel, B.S. Venkatesh, and B. Yegnanarayana. Real time face recognition system using autoassociative neural network models. *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, page 836, 2003.

[94] V. P. Palgianakos, M. N. Vrahatis, and G. D. Magoulas. Nonmonotone methods for backpropagation training with adaptative learning rate. *International Joint Conference on Neural Networks*, 3 :1762–1767, 1999.

[95] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. *Computer Vision, 1998. Sixth International Conference*, pages 555–562, 1998.

[96] D. Paul, S. Nowlan, and G. E. Hinton. Experiments on learning by backpropagation. *Technical Report TR CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University*, 1986.

[97] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2 :559–572, 1901.

[98] A. Pentland. Looking at people : Sensing for ubiquitous and wearable computing. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(1) :107–119, 2000.

[99] H. Permuter, J. Francos, and I. H. Jermyn. Gaussian mixture models of texture and colour for image database retrieval. *Acoustics, Speech, and Signal Processing, ICASSP*, 3 :569–572, 2003.

[100] F. Perronnin. A probabilistic model of face mapping applied to person recognition. *PHD Thesis, Ecole polytechnique fédérale de Lausane*, 2004.

[101] M. Propp and A. Samal. Artificial neural network architecture for human face detection. *Intelligence Ingeneering. Systems Artificial Neural Netwoks*, pages 535–540, 1992.

[102] L. R. Rabiner. A tutorial on hidden markov models and selecteds applications in speech recognition. *Proceeding of the IEEE*, 77 :257–285, 1989.

[103] S.J. Raudis and A.K. Jain. Small sample size effect in statistical pattern recognition : Recommendation for practitioners. *IEEE Transactions on Pattern Analysis and Intelligence*, 13(3) :252–264, 1991.

[104] D.A. Reynolds and R.C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1), 1995.

[105] P. Réfrégier. *Théorie du Signal*. Paris, France, 1993.

[106] F. Rosenblatt. The perceptron : a probabilistic model for information storage and retrieval in the brain. *Psychological Review*, pages 386–408, 1958.

[107] F. Rosenblatt. *Principles of Neurodynamics : Perceptrons and Theory of Brain Mechanisms*. Cornell Aeronautical Laboratory, 1961.

[108] D. Roth. Learning to resolve natural language ambiguities : A unified approach. *Proceeding of the fifteenth National Conference On Artificial Intelligence*, pages 806–813, 1998.

[109] D. Roth, M. H. Yang, and N. Ahuja. A SNOW-based face detector. *Advances in Neural Information Processing Systems 12 (NIPS 12), MIT Press, Cambridge*, 2000.

[110] P. M. Roth, H. Grabner, H. Bischof, D. Skocaj, and A. Leonardist. On-line conservative learning for person detection. *ICCCN '05 : Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 223–230, 2005.

[111] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing : Exploration in the Microstructures of Cognition*, pages 318–362, 1986.

[112] Zohra Saidane and Christophe Garcia. Automatic scene text recognition using a convolutional neural network. *Second International Workshop on Camera-Based Document Analysis and Recognition*, pages 100–106, 2007.

[113] T. Sakai, M. Nagao, and T. Kanade. Computer analysis and classification of photographs human faces. *Proc. First USA-Jaân Computer Conference*, pages 2–7, 1972.

[114] R. Salomon. Improved comvergence rate of back-propagation with dynamic adaptation of the learning rate. *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 269–273, 1990.

[115] F. Samaria and A. Harter. Parametrisation of a stochastic model for human face identification. *2nd IEEE Workshop on Applications of computer vision*, pages 138–142, 1994.

[116] R. Santiago-Mozos, JM. Leiva-Murillo, F. Perez-Cruz, and A. Artes-Rodriguez. Supervised-PCA and SVM classifiers for object detection in infrared images. *IEEE Conference on Advanced Video and Signal Based Surveillance*, 1999.

[117] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Proceedings of the eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1995.

[118] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT' 98 : Proceedings of the eleventh annual conference on Computational learning theory*, pages 80–91, New York, NY, USA, 1998. ACM.

[119] Henry Schneiderman. Feature-centric evaluation for efficient cascaded object detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2 :29–36, 2004.

[120] M.I. Sezan and R.I. Lagendijk. Frame rate conversion of motion picture films for progressive-scan HDTV. *Proc. EUSIPCO*, pages 1279–1282, 1992.

[121] W.G. Shadeed, D.I. Abu-Al-Nadi, and M.J. Mismar. Road traffic sign detection in color images. *10th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2 :890–893, 2003.

[122] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *unpublished but often cited*, 1968.

[123] Q. Song and J. Robinson. A feature space for face image processing. *Memorial University of Newfoundland*, 2 :97–100, 2000.

[124] Jie Sun, James M. Rehg, and Aaron Bobick. Automatic cascade training with perturbation bias. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2 :276–283, 2004.

[125] K-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1), 1998.

[126] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[127] S. Usui, S. Nakauchi, and M. Nakano. Internal color representation acquired by five-layer neural network. *Proceeding of the International Conference on Articial Neural Networks*, pages 867–872, 1991.

[128] D. Valentin, H. Abdi, A. J. O'Toole, and G. Cottrell. Connectionist models of face processing : A survey. *Pattern Recogition*, pages 1209–1230, 1994.

[129] V. Vapnik. Estimation of dependances based on empirical data [in russian]. *Nauka Russia (English Translation : Springer Verlag, New York, 1982)*, 1979.

[130] V. Vapnik. *The nature of Statistical Learning Theory*. Springer, 1995.

[131] P. Viola and M. Jones. Robust real-time object detection. *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning and Sampling*, 2001.

[132] T. Wakahara, Y. Kimura, and A. Tomono. Affine-invariant recognition of gray-scale characters using global affine transformation correlation. *IEEE Transactions on pattern analysis and machine intelligence*, 23(4), 2001.

[133] P. Werbos. Backpropagation : Past and future. *Proceedings of the international Conference on Neural Networks*, pages 281–284, 2002.

[134] Rong Xiao, Long Zhu, and Hong-Jiang Zhang. Boosting chain learning for object detection. In *ICCV '03 : Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 709, Washington, DC, USA, 2003. IEEE Computer Society.

[135] Shengye Yan, Shiguang Shan, Xilin Chen, and Wen Gao. Locally assembled binary (lab) feature with feature-centric cascade for fast and accurate face detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0 :1–7, 2008.

[136] Shengye Yan, Shiguang Shan, Xilin Chen, Wen Gao, and Jie Chen. Matrix-structural learning (msl) of cascaded classifier from enormous training set. In *CVPR*, 2007.

[137] J. Yang and J.Y. Yang. From image vector to matrix : A straightforward image-projection technique-IMPCA vs PCA. *Pattern Recognition*, 35(9) :1997–1999, 2002.

[138] Jian Yang, David Zhang, A. F. Frangi, and Jing yu Yang. Two-dimensional pca : A new approach to appearance-based face representation and recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(1) :131–137, 2004.

[139] M. H Yang, N. Ahuja, and D. Kriegman. Face detection using mixtures of linear subspaces. *Proceedings, Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[140] S.J. Young, N.H. Russell, and J.H.S Thornton. Token passing : a simple conceptual model for connected speech recognition systems. *Technical Report CUED/F-INFENG/TR38, Cambridge University Engineering Dept*, 1989.

ANNEXE A

Base de données d'images

A.1 Face 1999

450 images de visages provenant de 27 personnes distinctes avec un arrière plan complexe (figure : A.1). Nous en faisons usage pour tester les versions les plus basiques de notre système de détection car elle constitue une base de détection relativement simple au regard des bases de détection de visages standards.

FIGURE A.1 – Exemples d'images de la base de données Face 1999

A.2 CMU

130 images contenant 507 visages de face (figure : A.2). Cette base d'images est la plus couramment utilisée pour tester les systèmes de détection de visages. Elle est notamment formée par 23 images utilisées par Sung et Poggio dans [125] et de 14 images ne contenant aucun visage. On retrouve souvent une base d'image nommée CMU-125 contenant de 487 visages. Elle correspond à la base CMU classique moins cinq images contenant des visages dessinés.

FIGURE A.2 – Exemples d'images de la base de données CMU avec les visages annotés à détecter

A.3 Caltech WebFaces

7092 images contenant 10524 visages de face (figure : A.3) collectées sur Internet à l'aide de Google Image. La position des yeux, du nez et de la bouche a été manuellement annotée. On peut remarquer cependant quelques rares visages présents mais non annotés, sans doute pour des raisons d'erreurs humaines. Cette base reste cependant à la fois un bon moyen d'évaluation puisque représentative des images que l'on peut trouver sur Internet, mais peut aussi servir de base de visages exemples de par la variété et le nombre des visages disponibles.

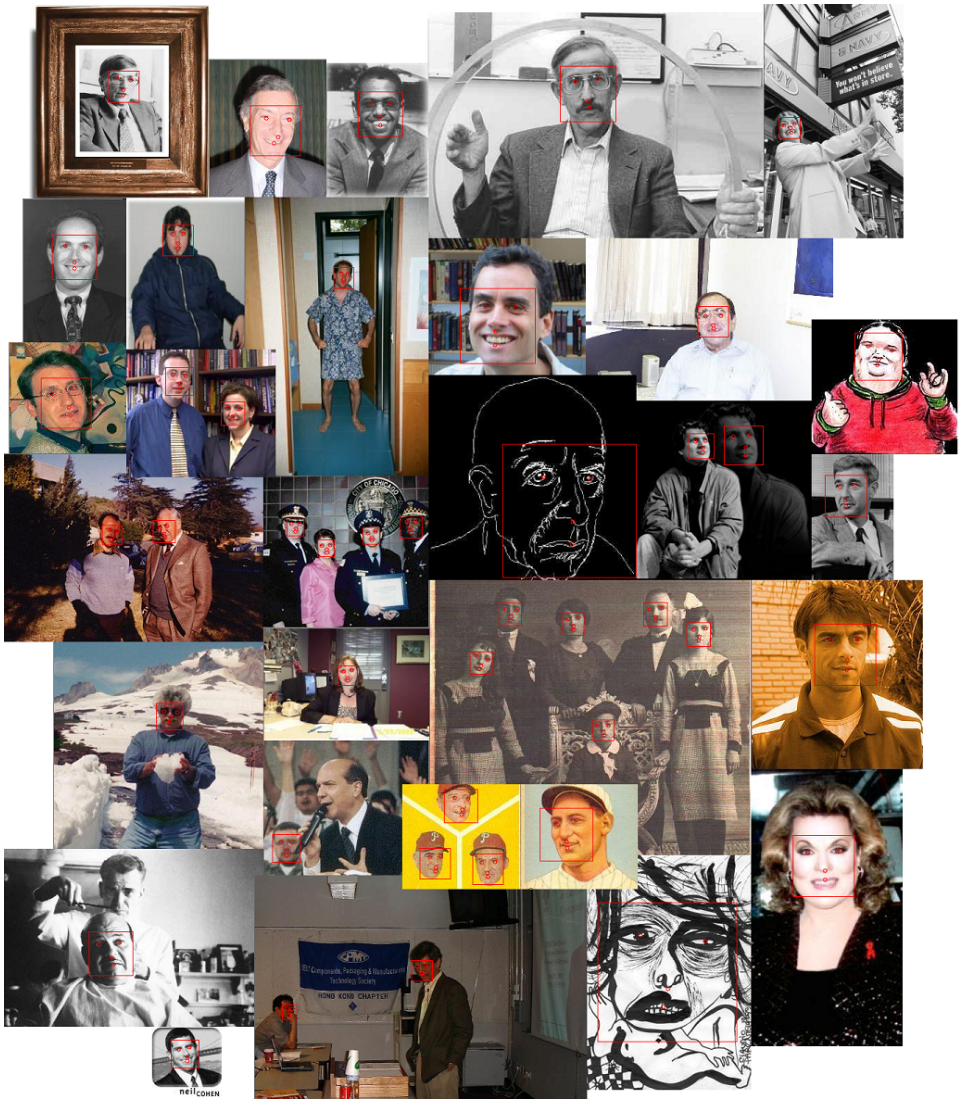


FIGURE A.3 – Exemples d'images de la base de données Caltech WebFaces avec les visages annotés