Reconnaissance d'objets en vision artificielle : application à la reconnaissance de piétons

Laetitia Leyrit

N° d'ordre : D.U : 2071
EDSPIC : 498

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

Ecole Doctorale
Sciences Pour L'Ingénieur De Clermont-Ferrand

Thèse
présentée par :
LAETITIA LEYRIT

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Vision pour la robotique

Reconnaissance d'objets en vision artificielle : Application à la reconnaissance de piétons

---

Soutenue publiquement le 22 novembre 2010 devant le jury :

M.	Laurent	TRASSOUDAINE	Pr Univ. Blaise Pascal	Président
Mme	Catherine	ACHARD	MCF Univ. Pierre et Marie Curie	Rapporteur
Mme	Sylvie	TREUILLET	MCF Univ. d'Orléans	Rapporteur
Mme	Alice	CAPLIER	Pr Univ. Joseph Fourier	Examinatrice
M.	Thierry	CHATEAU	MCF Univ. Blaise Pascal	Examinateur
M.	Jean-Thierry	LAPRESTÉ	Pr Univ. d'Auvergne	Directeur de thèse

Remerciements

Lorsque je parcoure une thèse, je lis toujours la page des remerciements, même si force est de constater que je n'y ai jamais trouvé de quoi développer le moindre algorithme ! Mais c'est dans cet unique feuillet que transparaît le plus souvent le dur labeur d'un thésard, ses années de travail, d'échecs parfois, mais aussi de réussites, couronnées par ce dernier, mais ô combien important, diplôme !
Je me soumettrais donc bien volontiers à la coutume et j'espère que je saurais remercier ici toutes les personnes qui, de près ou de loin, ont participé à la réalisation de ma thèse.

Je commence par m'adresser à mon jury de thèse. Je remercie M. Laurent Trassoudaine pour avoir accepté la présidence du jury, et Mme Alice Caplier pour sa participation. Je remercie particulièrement Mme Sylvie Treuillet et Mme Catherine Achard pour avoir accepté de rapporter mes travaux. J'ai apprécié la justesse de leurs remarques, mais aussi la sympathie qu'elles m'ont témoignée.
Je m'adresse ensuite à mon encadrement de thèse. M. Jean- Thierry Lapresté a été un directeur de thèse cordial et chacun de ses conseils m'ont été précieux. Quant à M. Thierry Chateau, c'est un encadrant de thèse accessible avec qui j'ai pu parler sans contraintes ni pression de mes travaux et de tout ce qui concernait de près ou de loin ma thèse. Je les remercie donc chaleureusement pour m'avoir suivie pendant ces années.

Je tiens également à remercier le personnel du dépatement GIM de l'IUT où j'ai adoré enseigner tout au long de ma thèse, notamment M. Jean-Marc Lavest, M. François Collange, M. Flavien Paccot, M. Nicolas Deltroy et M. Eric Royer. Leur soutien ne m'a jamais fait défaut et la bonne ambiance du département a été pour moi une véritable bulle de décompression.

Je remercie toutes les personnes du laboratoire LASMEA qui m'ont accompagnée tout au long de ma thèse. Je pense en particulier à mes collègues du bureau 4016, Manu, Bertrand, Alex, Pierre B. et Eric, Christophe (à qui mon ordinateur doit certainement sa survie à ces années de thèse) et tous mes compagnons de galère, Pierre A., Jonathan, Hala, Réda, Ahmed, Franck, Ismaïl, Fabio, Laurence, Pierre L., Baptiste, ... La liste ne sera pas exhaustive mais le cœur y est. Ils ont tous contribué à traverser cette thèse dans une atmosphère de travail agréable ; et j'ai toujours trouvé chez l'un ou l'autre une oreille attentive et un soutien opportun.

Une pensée va également à Manue et Charlène, avec qui j'ai passé d'agréables soirées. Elles ont supporté, avec une bonne humeur sans faille, beaucoup d'histoires du laboratoire, qui ont dû être, je le crains, peu passionnantes pour elles !
Et je fais un clin d'œil à nos week-ends canoë, qui resteront de mythiques « envers du décor »de ces années de thèse.

Je remercie enfin toute ma famille, qui a toujours été là pour moi durant ma thèse mais aussi tout au long de mes études. Je pense en particulier à mes grands-parents et surtout à mes parents que j'embrasse très fort.
J'envoie de gros bisous à mes seusœurs : Charlène, ma colocataire de choc et confidente, Morgane et Chloé, pour leur tendresse et bonne humeur. Les virées shopping, cinéma, piscine ou autres m'auront permis de faire des pauses très appréciables durant cette thèse et de relativiser beaucoup de choses.

Enfin, je remercie de tout mon cœur François, sans qui cette thèse n'aurait pas été la même.

Résumé

Ce mémoire présente les travaux réalisés dans le cadre de ma thèse. Celle-ci a été menée dans le groupe GRAVIR [1] du LASMEA [2] au sein de l'équipe ComSee [3] qui se consacre à la vision par ordinateur. Ces travaux s'inscrivent dans le cadre d'un projet de l'Agence Nationale pour la Recherche s'intitulant « Logiciels d'Observation des Vulnérables ». Son but est de concevoir des logiciels détectant des piétons en danger et d'améliorer ainsi la sécurité routière.

Ma thèse a pour but de détecter et de reconnaître les piétons dans les images. Celles-ci proviennent d'une caméra embarquée dans un véhicule circulant en milieu urbain. Ce cahier des charges implique de nombreuses contraintes. Il faut notamment obtenir un système fonctionnant en temps réel pour être capable de détecter les piétons avant un éventuel impact. De plus, ces piétons peuvent être sujets à de nombreuses variations (taille, type de vêtements...), ce qui rend la tâche de reconnaissance d'autant plus ardue. La caméra étant mobile, aucune information ne pourra être extraite du fond. Dans ma thèse, nous mettons en œuvre différentes méthodes de vision par ordinateur, toutes basées apprentissage, qui permettent de répondre à ces attentes. Le problème se traite en deux phases.

Dans un premier temps, une étape de traitement hors ligne nous permet de concevoir une méthode valide pour reconnaître des piétons. Nous faisons appel à une base d'apprentissage. Tout d'abord, un descripteur d'images est employé pour extraire des informations des images. Puis, à partir de ces informations, un classifieur est entraîné à différencier les piétons des autres objets. Nous proposons l'utilisation de trois descripteurs (ondelettes de Haar, histogrammes de gradients et descripteur binaire). Pour la classification, nous avons recours à un algorithme de *Boosting* (AdaBoost) et à des méthodes à noyaux (SVM, RVM, moindres carrés). Chaque méthode a été paramétrée, testée et validée, tant au niveau description d'images que classification. La meilleure association de toutes ces méthodes est également recherchée.

Dans un second temps, nous développons un système embarqué temps réel, qui soit capable de détecter les piétons avant une éventuelle collision. Nous exploitons directement des

1. acronyme de « Groupe d'Automatique, VIsion et Robotique ».
2. acronyme de « LAboratoire des Sciences et Matériaux Et d'Automatique ».
3. acronyme de « Computers that See ».

images brutes en provenance de la caméra et ajoutons un module pour segmenter l'image, afin de pouvoir intégrer les méthodes de description et classification précédentes et ainsi répondre à la problématique initiale.

Mots-clés : reconnaissance d'objets, détection, apprentissage, classification, description d'images, base d'apprentissage, caméra embarquée, temps-réel, piétons.

Abstract

This thesis has been realized in the group GRAVIR [4] of the LASMEA [5] with the team ComSee [6], which works on computer vision. My research was involved in a projet of the « Agence Nationale pour la Recherche »nammed « Logiciels d'Observation des Vulnérables ». Its goal was to create softwares to detect endangered pedestrians and thus to improve road safety.

My thesis aims to detect and to recognize pedestrians in images. These come from a camera embedded into a vehicle, which is driven in urban areas. These specifications involve many constraints. We have to obtain a real-time system for detect pedestrians before a possible collision. Moreover, pedestrians should be very variable (size, clothes, ...), which make the recognition more complicated. As the camera is moving, no information could be taken from the background. In my thesis, we implement several methods of computer vision, all based on a learning stage, which answer to all theses expectations. The problem is solved in two steps.

Firstly, a off-line stage allows us to design a method able to recognize pedestrians. We use a learning database. First of all, an image descriptor is used to extract informations of the images. Then, from these informations, a classifior is trained to differentiate pedestrians to others objects. We suggest to use three descriptors (Haar wavelets, histograms of oriented gradients and binary descriptor). For the classification task, we use a *Boosting* algorithm (AdaBoost) and kernel methods (SVM, RVM, least squares). We define all the parameters, and each method - of description or classification - is then tested and validated. The best association of these methods is also searched.

Secondly, we realize an embedded real-time system, which is able to detect pedestrians before a possible collision. We directly use raw images coming from the camera et add a segmentation stage, so as to insert previous description and classification méthods and thus to answer to the initial problem.

Key-words : object recognition, detection, learning, classification, image description, learning database, embedded camera, real-time, pedestrians.

4. for « Groupe d'Automatique, VIsion et Robotique ».
5. for « LAboratoire des Sciences et Matériaux et d'Automatique ».
6. for « Computers that See ».

Table des matières

Liste des figures

Liste des tableaux

Notations

$\mathbf{x}$	un vecteur
$\mathtt{X}$	une matrice
$\|\mathbf{x}\|_1$	norme l_1 du vecteur $\mathbf{x}$ (ou norme 1)
$\|\mathbf{x}\|$	norme euclidienne du vecteur $\mathbf{x}$ (ou norme 2)
$\|\mathtt{X}\|$	norme de Frobinus de la matrice $\mathtt{X}$ (ou norme 2)
$\langle \mathbf{x}_1 \cdot \mathbf{x}_2 \rangle$	produit scalaire entre les vecteurs $\mathbf{x}_1$ et $\mathbf{x}_2$
$\mathbf{x} \otimes \mathbf{y}$	produit tensoriel entre les vecteurs $\mathbf{x}$ et $\mathbf{x}_2$
I_i	image numéro i
V_i	vecteur de caractéristiques
$V_i(k)$	composante du vecteur V_i avec $k \in [1, ..., K]$
ψ	ondelette mère
$\psi_{s,\tau}(t)$	ondelette de dilatation s et de translation τ en fonction de t
$I_{\text{intégral}}$	image intégrale
G_x	composante verticale du gradient
G_y	composante horizontale du gradient
θ	un angle
C_R	matrice de covariance d'une région R
$LBP_{R,N}(x, y)$	*Local Binary Pattern* de paramètres $\{R, N\}$ aux coordonnées $\{x, y\}$
p	un point
$I(p)$	intensité lumineuse du point p
Q	matrice de renforcement
$Q(s, a)$	élément de la matrice Q pour l'action a à l'état s
$\mathcal{S}$	un ensemble d'apprentissage
N	nombre d'exemples dans $\mathcal{S}$
y	un scalaire correspondant à la classe d'un objet
h	règle de décision d'un classifieur faible
H	règle de décision finale d'un classifieur fort
C	un classifieur
$\mathbf{p}$	vecteur poids d'un algorithme d'AdaBoost
ϵ	une erreur
α	un coefficient
$sign$	fonction signe

$\mathbf{w}$	vecteur poids d'une machine à noyau
b	biais
K	une fonction noyau
M_f	marge fonctionnelle
M_g	marge géométrique
L	lagrangien
α_i	multiplicateurs de Lagrange
ξ_i	variable ressort
C	constante
$p(\mathbf{x}_1\|\mathbf{x}_2)$	probabilité de l'hypothèse x_1 sachant x_2
$diag$	matrice diagonale
$\boldsymbol{\phi}$	vecteur de fonctions de base
$\mathtt{\Phi}$	matrice des vecteurs de base
Φ^+	pseudo-inverse de Φ
σ^2	un écart-type
$\nabla\nabla$	Hessien
$R(\mathbf{w})$	terme d'amortissement sur $\mathbf{w}$
λ	terme de régularisation
$\mathtt{R}$	matrice de rotation
$\mathbf{T}$	vecteur de translation

Chapitre 1

Introduction

A l'heure actuelle, les systèmes de vision sont de plus en plus répandus (webcam, caméscope...) et les caméras se sont installées partout dans notre quotidien. Elles sont utilisées pour réaliser de la vidéosurveillance (dans les magasins, rues ou aéroports), de l'aide à la conduite (aide au guidage ou détection d'obstacles), et bien d'autres applications encore... Pour l'être humain, voir est une tâche innée et nous ne mesurons souvent pas la difficulté pour obtenir les mêmes performances artificiellement. Malgré les avancées de la vision par ordinateur, les systèmes développés sont très loin d'égaler les performances de l'œil et du cerveau humains.
Mes travaux de thèse se déroulent dans ce cadre complexe mais stimulant. Il s'agit de mettre au point un système capable de détecter des piétons à partir d'images provenant d'une simple caméra. Le problème est d'autant plus difficile que le système de vision est embarqué dans un véhicule se déplaçant en milieu urbain : l'environnement visuel contient de nombreux éléments, est sujet à de multiples variations, et de plus, le processus doit prendre place en temps réel [1]*, pour éviter une éventuelle collision.*
Ce premier chapitre est une introduction à mes travaux ; il s'agit de situer dans quel contexte scientifique ils sont réalisés, en présentant les différents domaines de recherche auxquels ils sont rattachés, et l'état de l'art sur ces questions.

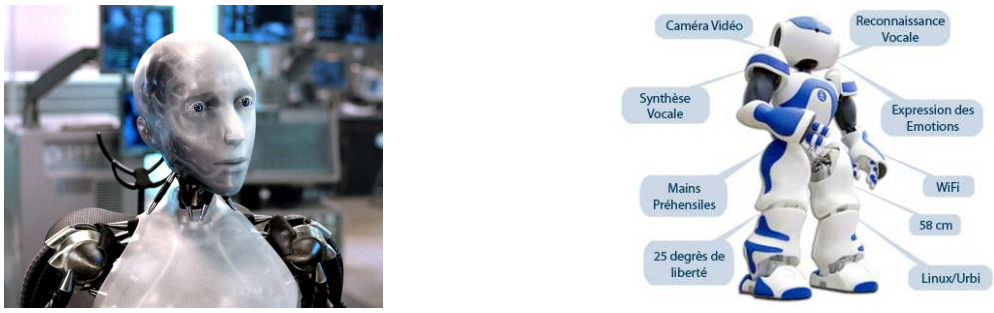
1.1 Reconnaissance des formes

1.1.1 De l'Intelligence Artificielle...

Le terme d'Intelligence Artificielle (IA) désigne toutes les techniques et méthodes qui permettent de doter des systèmes informatiques de capacités proches de celles de l'être humain. La science-fiction s'est appropriée ce domaine en imaginant des robots, dotés d'une intelligence qui serait semblable à celle de l'homme. Toutefois, de nos jours, notre maîtrise de l'IA est

1. En informatique industrielle, un système est dit temps réel lorsqu'il contrôle un procédé physique à une vitesse adaptée à son évolution.

bien plus modeste ; les systèmes développés ne sont capables que d'imiter certaines capacités de l'homme. Prenons par exemple des robots alliant des capacités de perception de l'environnement et de commande qui sont capables de se déplacer de façon autonome en évitant les obstacles (voir figure 1.1).
Le but de ma thèse est de réaliser un système permettant de détecter des piétons devant un véhicule à partir d'une caméra embarquée. Mon travail se situe au niveau de la perception ; effectivement, il s'agit de recréer artificiellement les capacités de perception d'un conducteur qui chercherait de façon active les piétons situés devant son véhicule avec lesquels il pourrait entrer en collision. Le système désiré est donc apparenté à un système d'Intelligence Artificielle.



a b

FIGURE 1.1 – Intelligence Artificielle : de la fiction à la réalité
a) exemple d'un robot imaginé par la science-fiction (extrait du film I-Robots) ayant les mêmes capacités que l'être humain,
b) exemple des capacités d'un robot actuel (NAO de la société Aldebaran Robotics).

1.1.2 ... à la reconnaissance des formes

Mais l'IA est un très vaste domaine et les applications sont diverses et variées. Il est possible de distinguer plusieurs branches. Mes travaux de thèse s'inscrivent dans le cadre de la « reconnaissance des formes » [47]. Cette appellation désigne les méthodes qui, à partir de l'observation d'un objet, lui attribuent une classe. Par exemple, un système qui, à partir d'une observation radar, classifie des objets volants soit dans une classe *avions*, soit dans une classe *non-avions*, réalise une tâche de reconnaissance d'objets.
L'observation d'un objet peut être très différente selon le domaine d'activité dans lequel nous évoluons. Souvent c'est une observation visuelle qui est utilisée (photo, séquence vidéo, image médicale, image satellite, etc.), mais il est également possible d'avoir une analyse radar, laser ou sonore, le champ d'applications étant très large. De même, le motif à reconnaître et à classer peut-être simple ou complexe, et les outils développés sont adaptés à chaque utilisation. Par exemple, un cube sera plus facile à reconnaître qu'un visage puisqu'il a des propriétés physiques constantes dans le temps et l'espace alors qu'un visage est sujet à beaucoup plus de changements.

Dans le cadre de ma thèse, nous cherchons à reconnaître des piétons à partir d'images vidéo. Grâce à une caméra embarquée, une observation visuelle de la scène à l'avant d'un véhicule est obtenue ; la problématique principale est alors de reconnaître les piétons présents dans les images, c'est-à-dire être capable d'indiquer si les objets présents dans l'environnement devant le véhicule appartiennent ou non à la classe *piétons*.
Dans ce manuscrit, nous présentons les différentes étapes permettant de réaliser ce système de détection et de reconnaissance de piétons. Il ne s'agit pas de fournir ici une description exhaustive de l'état de l'art, ce qui ne saurait être réalisable dans ce mémoire tant les méthodes sont diverses et variées suivant les domaines d'applications, mais plutôt de donner une vue d'ensemble des éléments qui permettent de réaliser cette tâche.
Actuellement, il existe deux types de méthodes : les méthodes basées modèle et les méthodes basées apprentissage. Les premières utilisent un modèle du piéton et identifient un nouvel objet en calculant sa distance au modèle [10]. Les secondes font appel à une phase d'apprentissage hors ligne [2] qui génère un classifieur capable de différencier les piétons des autres objets [74]. Les techniques développées dans cette thèse appartiennent à cette catégorie. Le tableau (1.1.2) présente les différentes étapes. Pour réaliser ce système, une phase d'apprentissage préalable est nécessaire (première colonne du tableau) ; la phase de traitement en ligne est alors possible en utilisant les informations de la phase d'apprentissage (deuxième colonne du tableau).
Un glossaire (page 149) donne une définition des principaux mots de vocabulaire utilisés dans la suite du manuscrit.

1.2 Segmentation d'images

Nous avons vu précédemment que la première étape d'une reconnaissance consiste à réaliser une segmentation de l'image. Cette étape n'intervient généralement pas lors de l'apprentissage hors ligne lorsqu'une base d'apprentissage comprenant déjà des images « prédécoupées » autour de l'objet à reconnaître est utilisée. Par contre, lors de la détection et reconnaissance en ligne, les données proviennent d'une caméra dont le champ couvre une large scène et l'objet recherché n'en occupe qu'une partie. Afin de pouvoir détecter puis reconnaître cet objet, il est nécessaire de diviser l'image initiale en une ou plusieurs zones d'intérêt où l'objet est susceptible de se trouver. Cette étape de prétraitement de l'image permet ainsi de limiter les temps de calcul qui suivent puisque seules les zones conservées seront traitées.
Selon les capacités du descripteur qui est ensuite mis en œuvre dans la chaîne de traitement, ces zones pourront être découpées à nouveau en une ou plusieurs imagettes centrées autour de l'objet et mises aux bonnes dimensions afin de mettre en adéquation les données courantes avec les besoins du classifieur. Celui-ci pourra alors attribuer chacune d'entre elles à la classe *piétons*

2. Un processus est dit « en ligne » lorsqu'il s'exécute au fur et à mesure du travail principal qu'il doit réaliser ; un processus hors-ligne réalise des tâches en différé du travail principal, soit avant en prévision d'une opération particulière à effectuer pour le travail principal, soit après pour traiter par exemple des informations récoltées pendant le travail principal.

	1. Traitement hors ligne Apprentissage	2. Traitement en ligne Détection et reconnaissance
Entrées :	un ensemble d'images avec leurs étiquettes de classe : positif ou négatif	une nouvelle image
Segmentation :	segmentation des images en vignettes autour des exemples	découpage de l'image initiale en des zones « intéressantes », c'est-à-dire des imagettes où l'objet recherché est susceptible de se trouver
Description :	analyse via un descripteur adéquat de chaque exemple de la base d'apprentissage pour en extraire des informations	extraction des informations sur chaque imagette en utilisant le même descripteur que pendant la phase d'apprentissage
Classification :	traitement des informations fournies précédemment par le descripteur avec les étiquettes de classe afin de créer un classifieur qui sera capable de distinguer les exemples positifs des négatifs	utilisation du classifieur entraîné pendant l'apprentissage pour que le système indique si l'objet recherché est présent ou non dans chaque imagette

TABLE 1.1 – Les différentes étapes constituant un système de détection d'objets
En premier, l'apprentissage hors ligne et en second, la détection et reconnaissance en ligne.

ou *non-piétons*.
Pour sélectionner ces zones, deux types de segmentation peuvent être mises en œuvre. Dans un premier temps, si des connaissances *a priori* sur la scène sont disponibles, elles peuvent être utilisées ; par exemple, en vidéosurveillance, la recherche du visage d'une personne peut être ciblée sur la porte d'entrée d'un magasin, à une hauteur adéquate. Mais des informations sur la scène ne sont pas toujours connues, et même parfois, cela n'est pas suffisant voire inadaptable, notamment lorsqu'il s'agit d'une caméra mobile.
Dans un deuxième temps, si aucune information n'est connue, des méthodes particulières peuvent être employées pour segmenter les images ; celles-ci regroupent par zones des pixels ayant des propriétés communes (citons par exemple les approches par seuillage, par frontières [14, 49] ou par régions [8]) sachant qu'il est possible de fusionner plusieurs techniques pour segmenter au mieux l'image. Tout cela dépend de l'application et du temps imparti pour la réaliser.

1.3 Description d'images

Lorsque des zones de l'image ont été définies et sélectionnées, il faut analyser l'information qu'elles contiennent. Deux types de traitement sont envisageables.
Le premier type consiste en une amélioration de la qualité de l'image, par exemple pour atténuer les défauts du capteur et les bruits associés aux conditions d'acquisition (flou, mauvaise illumination...). Ces traitements sont parfois réalisés avant la phase de segmentation de l'image mais peuvent l'être après de façon à avoir moins de zones à s'occuper et ainsi réduire les temps de calcul.
Le deuxième type est directement lié à la tâche de reconnaissance d'objets. Il faut extraire des zones d'intérêt des informations relatives à ce qui est contenu dans l'image ; il existe des descripteurs d'images qui permettent de caractériser l'information disponible. Cette étape primordiale est détaillée dans le chapitre 2. Deux façons de procéder sont envisageables :

- soit l'image est décrite en des points particuliers et significatifs : c'est un descripteur local et ces points intéressants sont des points d'intérêt ;
- soit tous les pixels de l'image correspondants à la zone d'intérêt sont pris en compte dans la description : c'est un descripteur global.

Descripteur local En ce qui concerne les descripteurs locaux, des méthodes permettent de détecter les points d'intérêt. Par exemple, le détecteur de points de Harris [39] qui est certainement le plus répandu ; son principe est de détecter les changements brusques d'intensité sur l'image pour ainsi mettre en évidence des coins.

FIGURE 1.2 – Exemple de détecteur de points d'intérêt : le détecteur de Harris
Le détecteur de Harris permet de sélectionner dans l'image initiale (à gauche) un ensemble de points d'intérêt (en rouge dans l'image à droite) qui correspondent à des changements brusques de l'intensité dans 2 *directions.*

Puis un descripteur caractérise chaque point d'intérêt par son voisinage. Le plus souvent, il s'agit de l'information qu'a calculée le détecteur.

L'utilisation de points d'intérêt est difficile à mettre en place dans le cas d'un système de reconnaissance avec caméra embarquée. En effet, le piéton est un objet déformable et de plus en mouvement ; les points d'intérêt risquent de ne pas être les mêmes d'une image à l'autre (par exemple en fonction de la posture de la personne ou des conditions d'illumination) et le classifieur qui suit sera incapable de faire le lien avec les classes d'objets envisagées. Néanmoins, il existe des travaux de catégorisation [54] utilisant des points d'intérêt.

Descripteur global En reconnaissance d'objets, un descripteur global est plus facile à utiliser car il traite l'image dans son intégralité. Le descripteur est ainsi moins sensible aux déformations du piéton d'une image à l'autre. Deux images proches doivent donc conduire à deux descripteurs proches. Un classifieur peut ainsi être entraîné sur ces données pendant l'apprentissage et il pourra associer ces deux images à une même classe grâce à une notion de distance entre les descripteurs (par exemple distance euclidienne). Il en sera de même pour la reconnaissance : en utilisant le même descripteur que pendant l'apprentissage, le classifieur pourra associer un nouvel objet à une classe d'objets apprise.
De nombreuses méthodes existent. Le choix dépend avant tout de l'application visée et du temps de calcul disponible pour la réaliser. En effet, dans le cas de reconnaissance et de détection en temps réel, il faut souvent faire un compromis entre qualité de la réponse et temps d'exécution. Or si les informations fournies par le descripteur ne sont pas assez précises, le classifieur sera difficile à entraîner pendant l'apprentissage et les résultats qu'il fournira en reconnaissance seront donc peu fiables. A l'heure actuelle, les descripteurs les plus utilisés en reconnaissance d'objets sont les ondelettes de Haar [74] et les histogrammes de gradients orientés [19] du fait des résultats satisfaisants et rapides qu'ils permettent d'obtenir. Mais d'autres méthodes peuvent être envisagées. Le chapitre 2 est entièrement consacré aux descripteurs mis en œuvre dans ma thèse.

1.4 Apprentissage automatique

Les méthodes d'apprentissage automatique sont utilisées dans beaucoup de domaines (robotique, vision artificielle, neuroscience, biologie...) et sont très variées. C'est une autre branche de l'Intelligence Artificielle à laquelle fait couramment appel la reconnaissance des formes. Cet axe de recherche concerne le développement de méthodes ou d'algorithmes qui ont pour but, à terme, la réalisation de tâches complexes que des programmes plus simples ne pourraient pas réussir. Calquant le modèle humain, une entité informatique « apprend » à réaliser ces tâches.
Il existe différentes approches dont un certain nombre sera présenté dans le chapitre 3 (voir section 3.1). Succinctement, il s'agit de méthodes algorithmiques permettant à une machine d'agir à partir d'une phase de travail hors ligne appelée « apprentissage ». Pendant celle-ci, la machine est entraînée sur des données correspondant à un problème donné. Elle est ensuite capable de généraliser ce qu'elle a étudié pour répondre de manière adéquate à des problèmes similaires.
En reconnaissance d'objets, le but est de faire de la classification. La phase d'apprentissage

consiste à enseigner au système ce que sont les objets des différentes classes. Le système ainsi entraîné doit ensuite être capable d'associer un nouvel objet à la bonne classe.
Dans le cas de la reconnaissance d'un objet particulier, il s'agit d'une classification entre deux classes d'objets : les exemples positifs sont tout ce que peut être cet objet et les exemples négatifs sont tout ce qu'il n'est pas.
Dans cette thèse, il s'agit de mettre au point un système de détection de piétons par vision. Des méthodes d'apprentissage permettent de réaliser le cœur du système : la reconnaissance des piétons. Pour ce faire, nous avons recours à des bases d'images importantes (plusieurs milliers) contenant des exemples de piétons et de non-piétons (bâtiments, véhicules, panneaux de signalisation, arbres, etc.) afin d'être capable de distinguer ces deux classes d'objets.
Nous mettons ensuite en œuvre des techniques d'apprentissage populaires à l'heure actuelle pour faire de la reconnaissance d'objets : des algorithmes de *Boosting* et des méthodes à noyau. Le chapitre 3 présente ces méthodes et les contributions apportées à ce niveau.

1.5 Système de reconnaissance de piétons

Ma thèse se déroule dans le cadre d'un projet ANR [3] intitulé *LOVe* [4] dont le but est de lutter contre les collisions piétons/voitures. L'enjeu majeur est de développer des algorithmes capables de détecter, reconnaître et suivre en temps réel [5] les piétons évoluant dans l'environnement d'un véhicule, circulant en milieu urbain, à partir de différents capteurs embarqués sur le véhicule.
Ces dix dernières années, des améliorations ont permis d'augmenter de façon significative les performances de calcul des ordinateurs. Les applications de reconnaissance ont pu alors être envisagées en temps réel et des études ont abondé en ce sens.
Il existe de nombreuses techniques de détection de piétons pour l'aide à la conduite [33]. Chaque méthode développée est en lien direct avec le type de capteur utilisé ; certaines exploitent des données issues d'un système monovision, d'un système stéréovision, d'un télémètre laser ou encore d'un radar. Plus récemment, certains systèmes font appel à plusieurs capteurs car l'information sur l'environnement autour du véhicule est ainsi beaucoup plus riche [34]. Il s'agit de la fusion de données. Les résultats sont en général probants mais cela ne rend pas pour autant obsolète la recherche sur les applications mono-capteur. Tout d'abord, l'intérêt de développer une application d'aide à la conduite est de pouvoir l'adresser à un large public. Or installer plusieurs capteurs (caméra + télémètre + radar) sur la voiture de « monsieur-tout-le-monde » est inenvisageable financièrement et les méthodes utilisant un seul capteur gardent donc leurs

3. Agence Nationale de la Recherche
4. *LOVe* pour « LOgiciels d'Observation des VulnérablEs ».
5. Dans le contexte du projet *LOVe*, le système devra traiter les informations de façon à être capable de détecter les piétons avant que le véhicule arrive à leur niveau ; le cahier des charges du projet *LOVe* [43] indique notamment que « *Tout piéton, à une distance inférieure ou égale à 2 secondes * la vitesse du véhicule (en m/s), présent dans le champ de vision du capteur devra être détecté et signalé.* » (voir chapitre 5 pour plus de précisions).

intérêts. Toutefois, augmenter les performances des algorithmes bas-niveau [6] peut contribuer à améliorer ceux de la fusion de données ; cet axe de recherche contribue encore aujourd'hui à travailler sur les techniques mono-capteur.
Dans cette optique, les caméras sont très appréciées car leur coût est relativement faible par rapport aux autres capteurs cités ci-dessus ; en outre, un système utilisant une caméra ne nécessite qu'une calibration - bien maîtrisée à l'heure actuelle - pour exploiter les données. Enfin, l'information fournie est plus riche qu'avec certains autres capteurs (télémètre). La détection d'un piéton avec une seule caméra est ainsi envisagée même si exploiter l'information contenue dans une image est encore un challenge. Nous allons donc chercher dans cette thèse à développer des techniques de détection et de reconnaissance de piétons à partir d'une simple caméra. La réalisation des différentes fonctions présentées dans le tableau 1.1.2 doit s'intégrer dans un système de détection/reconnaissance de piétons monovision tel qu'il est montré en figure 1.3.

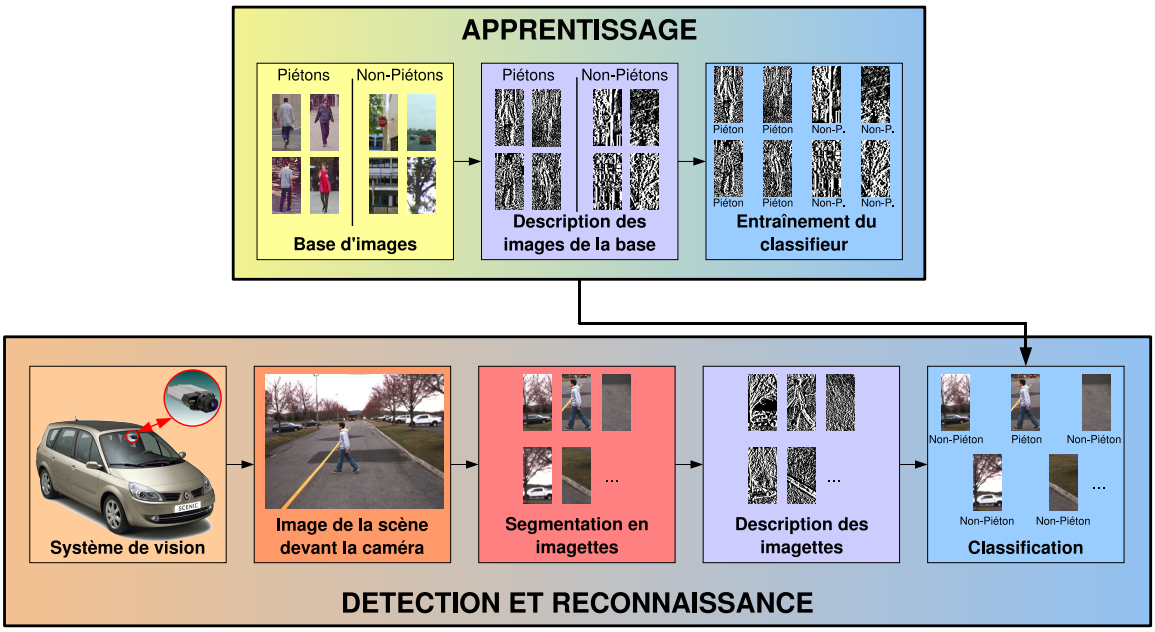


FIGURE 1.3 – Présentation du système complet désiré

Tout d'abord a lieu une étape d'apprentissage hors ligne avec la description de la base d'images et l'entraînement d'un classifieur. Pendant la phase de détection et de reconnaissance en ligne, la caméra fournit une image de la scène à l'avant du véhicule ; une segmentation est tout d'abord utilisée pour sélectionner des zones d'intérêt. Puis les imagettes extraites des zones d'intérêt sont décrites par le même descripteur utilisé pendant l'apprentissage. Le classifieur qui a été entraîné pendant l'apprentissage détermine alors la présence ou non de piétons dans ces imagettes.

6. Le bas-niveau désigne des algorithmes qui sont au plus proches du matériel et donc des capteurs.

Il existe déjà des études répondant à cette question [22, 33] ; nous présenterons dans le chapitre 5 des systèmes référents pour cette application précise (voir section 5.1).

1.6 Dans cette thèse

1.6.1 Problématique

Dans cette thèse, il s'agit d'exploiter les informations en provenance d'un système de vision se résumant à une simple caméra. Celle-ci fournit une image de la scène à l'avant du véhicule et le but de l'étude est d'être capable de reconnaître les piétons situés dans cet espace. Pour ce faire, plusieurs critères doivent être respectés :

- l'objet recherché - un piéton - est sujet à de nombreuses variations ; il faut prendre en compte non seulement les changements extrinsèques provoqués par le système de vision (conditions d'illumination, position d'observation de la caméra) mais aussi les changements intrinsèques de la classe *piétons* (dimensions des personnes, postures, vêtements, etc.) ;
- la caméra envoie une image représentant une large scène devant le véhicule ; bien que cette image soit de haute résolution, les objets recherchés à l'intérieur - les piétons - ne sont représentés que par quelques centaines de pixels ; il faut évaluer les performances des méthodes pour de faibles résolutions ;
- le but visé est une application temps réel ce qui impose des choix au niveau du temps d'exécution des méthodes utilisées : il faut que la méthode développée soit capable de détecter les piétons avant que la voiture n'arrive à leurs niveaux ; les différentes tâches réalisées - segmentation, description, classification - devront donc être peu coûteuses en temps de calcul au moment de l'exécution.

Il existe des systèmes (voir état de l'art en section 5.1) qui répondent dans une certaine mesure à cette problématique ; il faut toutefois noter que tous n'ont pas été implémentés pour des applications d'aide à la conduite, sous les contraintes précédemment citées. Le but de cette thèse est donc de tester et de comparer les méthodes existantes susceptibles de répondre au cahier des charges, de proposer une combinaison judicieuse de ces méthodes, et d'apporter de nouvelles solutions pour améliorer l'existant. Le but à atteindre est un système complet présenté en figure 1.3 avec :

- une segmentation d'images rapide et judicieuse permettant de réduire le champ de recherche de piétons éventuels ;
- un descripteur d'image rapide mais suffisamment pertinent pour permettre une classification correcte ;
- un classifieur donnant une réponse rapide quant à la classe de l'objet, tout en ayant de bonnes performances de reconnaissance de piétons et peu de fausses détections.

La rapidité se mesure en fait pour tout le système : il faut que l'exécution de la totalité des processus se fasse en « temps réel ». Ici, cette expression sous-entend que le système soit ca-

pable de détecter et de reconnaître les piétons présents dans l'environnement du véhicule avant que celui-ci ne l'atteigne, afin que le conducteur soit averti d'une situation dangereuse (voir le cahier des charges section 5.2.3).
La performance de la globalité du système développé dépend des résultats fournis en sortie de la chaîne de traitement. Cela correspond donc aux résultats du classifieur et de la bonne reconnaissance des piétons. Le système sera parfait s'il est capable de reconnaître tous les piétons tout en ne donnant aucune fausse détection [7]. En pratique, le système recherché doit avoir un bon taux de reconnaissance avec un faible taux de fausses détections.

1.6.2 Contributions

Pour réaliser ce système, nous explorons dans cette thèse un ensemble de méthodes existantes tant au niveau de la description d'images que de la classification. Il s'agit de choisir ensuite les plus pertinentes et de les combiner judicieusement, en adéquation avec les critères signalés ci-dessus. Nous apportons également nos propres contributions aux méthodes existantes pour répondre au cahier des charges :

- notre premier apport est un nouveau descripteur simple mais efficace dans ce contexte particulier, présenté en section 2.5.3. Ce nouveau descripteur binaire, basé sur des comparaisons de luminance de pixels, permet d'atteindre de bons résultats en classification, tout en étant rapide à calculer. Nous avons ensuite utilisé des méthodes d'apprentissage, soit à base de *Boosting*, soit à base de méthodes à noyaux. Celles-ci se basent sur un modèle unique de classifieur et différentes manières de résoudre le problème sont envisagées. Nous utilisons des approches courantes (SVM, RVM) et nous présentons aussi une variante (voir section 3.4.4) utilisant une estimation au sens des moindres carrés ; celle-ci atteint des résultats de classification comparables aux méthodes usuelles mais avec une formulation plus simple ;
- nous avons associé descripteurs et classifieurs afin de trouver un système de reconnaissance performant ; les performances de chacune des méthodes proposées sont testées et comparées dans le cadre d'une expérimentation sur une base d'images dans le chapitre 4. Afin de gérer au mieux nos descripteurs et classifieurs, nous proposons également de combiner les méthodes à noyau avec un algorithme AdaBoost pour diminuer la taille de l'ensemble d'apprentissage tout en améliorant les résultats du classifieur (voir sections 4.6 et 4.7) ;
- nous intégrons enfin tous les éléments développés dans un système complet de détection/reconnaissance de piétons à partir d'une caméra embarquée pour réaliser un véritable outil d'aide à la conduite (voir chapitre 5).

Tous ces travaux ont donné lieu aux publications suivantes [Leyrit 08a, Leyrit 08b, Leyrit 09b, Leyrit 09a, Leyrit 10] répertoriées page 139.

7. Une fausse détection est un objet qui n'est pas un piéton mais que le système identifie comme un piéton.

1.6.3 Plan du mémoire

Le plan de la thèse suit la construction du système développé, à savoir :

- dans le chapitre suivant, nous introduisons les descripteurs d'images envisagés pour répondre à la problématique et comment ils sont construits ;
- dans le chapitre 3, la tâche de classification est explicitée. Les classifieurs sélectionnés sont des méthodes d'apprentissage supervisées : le *Boosting* et l'AdaBoost ainsi que les méthodes à noyau (SVM, RVM, moindres carrés et moindres carrés pénalisés) ;
- dans le chapitre 4, nous présentons les stratégies mises en place pour combiner descripteurs et classifieurs pour être capable de traiter des données de grande dimension. Nous donnons des résultats théoriques sur les méthodes sélectionnées tant au niveau description que classification. Il s'agit d'entraîner ces méthodes sur des données d'apprentissage et ensuite de valider leurs performances sur des données test. Nous comparons ainsi différentes méthodes et voyons comment se positionne la nôtre ;
- le chapitre 5 décrit l'application visée par ces travaux, en rapport direct avec le projet ANR *LOVe*. Tout le système est détaillé et des résultats expérimentaux sont fournis ;
- Enfin le chapitre 6 clôture ce manuscrit par une conclusion générale. Les méthodes développées, les résultats théoriques et pratiques réalisés pendant cette thèse sont analysés et critiqués. Des perspectives qui permettraient d'améliorer et de compléter ces travaux seront également proposées.

Chapitre 2

Descripteurs d'images en reconnaissance des formes

D'un point de vue informatique, une image de luminance est simplement un tableau de nombres, dont chacun est la valeur de la luminance du pixel correspondant dans l'image. Il apparaît aisément qu'à partir de tous ces nombres, il est difficile de reconnaître non seulement une forme complexe (comme une personne) mais même des informations de plus bas niveau comme une simple forme géométrique. C'est pourquoi des descripteurs d'images sont couramment utilisés. Ceux-ci permettent de calculer pour une image donnée une information plus riche à partir d'une méthode de calcul adaptée.

2.1 Introduction

Il existe deux types de descripteurs : les descripteurs locaux et les descripteurs globaux (voir section 1.3). Les premiers décrivent une simple portion de l'image. Généralement, il s'agit du voisinage d'un point précis de l'image qui contient des informations intéressantes pour décrire la scène ou l'objet présent dans celle-ci. Ces points appelés points d'intérêt sont généralement choisis via un détecteur *ad hoc*. Une fois les points détectés, ils sont décrits par un descripteur local qui utilise le voisinage de ces points.
Les seconds - les descripteurs globaux - font, quant à eux, une description de l'ensemble de l'image. Généralement, l'image est découpée et traitée en sous-blocs. Une description de chaque partie est calculée et au final une description de l'image entière est obtenue.
Le choix du descripteur - qu'il soit local ou global - dépend de l'application visée. Les informations extraites grâce à un descripteur sont essentielles mais ne sauraient être suffisantes en elles-mêmes. Les descripteurs locaux sont des descripteurs fins qui sont en général utilisés soit pour un nombre de points restreints dans le cadre d'une application temps réel, soit pour une description très précise d'une image pour des traitements d'image plus classiques qui ne requièrent pas d'exécution temps réel. En général, les descripteurs globaux sont utilisés pour des

images de petite taille qui contiennent un seul objet et non une scène complexe. Ils peuvent être utilisés pour les applications temps réel, donnant ainsi rapidement une information globale de ce qui est présent dans l'image. Bien sûr, l'utilisation d'un descripteur n'est pas cantonnée à une application donnée, et beaucoup d'entre eux peuvent être adaptés en fonction de l'utilisation désirée.
Le but de ma thèse est de reconnaître des piétons dans des images. Elles sont fournies par une caméra embarquée dans un véhicule. Celle-ci filme toute la scène devant la voiture. Une méthode explicitée dans le chapitre 5 nous permet de segmenter l'image afin d'obtenir un ensemble d'imagettes candidates. Ces imagettes contiennent un seul objet dont il faut déterminer la classe d'appartenance. Un descripteur global est alors appliqué. Il fournit une description qui permet à un classifieur (voir chapitre 3) de déterminer si un piéton est présent ou non dans chacune de ces imagettes. Lorsque cette tâche de classification est réalisée, il faut que le descripteur utilisé pour caractériser les objets fournisse des réponses proches pour des exemples analogues. En effet, le classifieur calcule ensuite une distance entre chaque description et, en fonction de celle-ci, il établit si les exemples correspondants appartiennent à la même classe ou non.

Notations Le terme « descripteur » désigne donc l'outil qui permet de décrire une image. Un nombre N d'images est disponible. Chacune est notée I_i, avec $i \in [1, ..., N]$ le numéro de l'image. Un descripteur fournit alors pour une image I_i un vecteur de caractéristiques noté V_i de taille K. Chaque composante k de ce vecteur est donc une caractéristique de l'image I_i - ou « *feature* » en anglais - qui est notée $V_i(k)$ avec $k \in [1, ..., K]$.
Nous décrirons dans ce chapitre plusieurs descripteurs ; tout d'abord, nous introduisons les descripteurs couramment utilisés en reconnaissance d'objets : les ondelettes de Haar (voir section 2.2), les histogrammes de gradients orientés (voir section 2.3), les matrices de covariance (voir section 2.4) et les descripteurs binaires (voir section 2.5). Nous présentons ensuite celui que nous avons développé dans le cadre de cette thèse (voir section 2.5.3).

2.2 Ondelettes de Haar

2.2.1 Introduction

L'analyse par ondelettes a été introduite dans les années 1980 pour étudier des signaux. Cette représentation donne simultanément des informations temporelles et fréquentielles, facilitant ainsi l'identification des caractéristiques physiques de la source du signal. D'abord utilisé en traitement du signal, leur champ d'applications s'est étendu à bien d'autres domaines, notamment au traitement d'images. Aujourd'hui elles sont un descripteur largement répandu en reconnaissance des formes.
Une ondelette est une fonction de carré sommable sur l'espace euclidien $\mathbb{R}^n$ ($n \in \mathbb{N}^*$) , le plus souvent oscillante et de moyenne nulle. De façon générale, les ondelettes sont regroupées en familles, comprenant une ondelette initiale ψ appelée « ondelette mère » et de l'ensemble de

ses images obtenues par transformations affines dans $\mathbb{R}^n$.
La formulation générale est donnée par :

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{2.1}$$

avec t la variable à étudier et $\psi_{s,\tau}(t)$ l'ondelette de dilatation s et de translation τ dérivant de l'ondelette mère ψ.

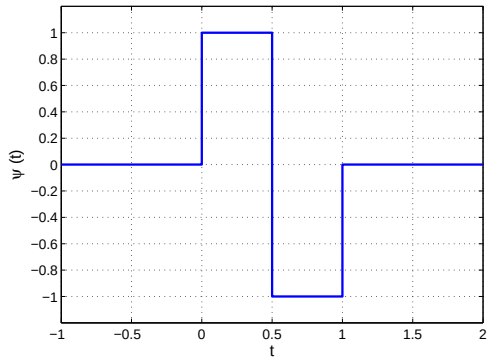


FIGURE 2.1 – L'ondelette de Haar

Il existe plusieurs types d'ondelettes utilisées en imagerie comme l'ondelette de Gabor [15, 88] ou l'ondelette de Haar [75, 95]. Celle-ci (voir figure 2.1) est définie par :

$$\psi(t) = \begin{cases} 1 & \text{si} \quad 0 \leq t < \frac{1}{2} \\ -1 & \text{si} \quad \frac{1}{2} \leq t < 1 \\ 0 & \text{sinon} \end{cases} \tag{2.2}$$

Papageorgiou et al. [73, 75] ont adapté cette ondelette afin de l'utiliser en tant que descripteur d'images pour la reconnaissance d'objets. L'idée est d'encoder les différences d'intensités entre des zones particulières d'une image Z_1 et Z_2 [1]. Prenons l'exemple d'une image de visage en niveaux de gris (voir figure 2.2) : il apparaît aisément que la bouche, le nez ou les yeux sont des zones de changements d'intensité. Les ondelettes de Haar permettent alors de capturer ces caractéristiques en les adaptant en position, en taille et en orientation ; la figure 2.3 schématise les différentes orientations usuellement exploitées en reconnaissance d'objets.
L'ondelette de Haar précédente (voir équation 2.2) est utilisée en 2D en prenant le produit tensoriel de la transformation par deux ondelettes 1D. En introduisant la « fonction de changement d'échelle » suivante :

$$\phi(t) = \begin{cases} 1 & \text{si} \quad 0 \leq t < 1 \\ 0 & \text{sinon} \end{cases} \tag{2.3}$$

1. Par convention, dans les figures suivantes (voir figures 2.2, 2.3, 2.4 et 2.5), Z_1 correspond à la partie grisée et Z_2 à la partie blanche.

L'ondelette de Haar verticale est donnée par : $\psi(x,y) = \psi(x) \otimes \phi(y)$ et l'horizontale par : $\psi(x,y) = \phi(x) \otimes \psi(y)$ [2]. L'ondelette qui encode les informations diagonales est obtenue par le produit : $\psi(x,y) = \psi(x) \otimes \psi(y)$. Elles sont représentées par la première ligne de la figure 2.3. Soit $p(x,y)$ l'intensité d'un pixel d'une image aux coordonnées (x,y). Schématiquement, une ondelette de Haar est ainsi calculée :

$$h = \sum a * \ p(x,y) \text{ avec } a = \begin{cases} 1 & \text{si} \quad p(x,y) \in Z_1 \\ -1 & \text{si} \quad p(x,y) \in Z_2 \\ 0 & \text{sinon} \end{cases} \tag{2.4}$$

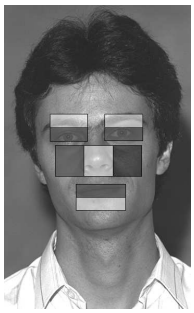


FIGURE 2.2 – Exemples de description d'un visage avec des ondelettes de Haar

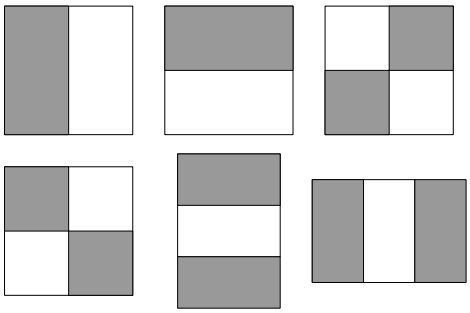


FIGURE 2.3 – Exemples d'ondelettes de Haar

Dans [75] les ondelettes sont choisies pour caractériser des endroits particuliers de l'image. Pour caractériser au mieux un objet, chaque ondelette est adaptée en position et taille. Par exemple, pour un visage, une ondelette d'orientation horizontale est très souvent utilisée pour la bouche avec une taille adéquate et une ondelette verticale pour caractériser le nez (voir figure 2.2). Mais pour améliorer le modèle, les ondelettes sont déplacées autour de la position initiale afin d'obtenir un ensemble complet pour décrire l'image.

2. $\otimes$ désigne le produit tensoriel.

2.2.2 Calcul rapide d'ondelettes de Haar à l'aide d'image intégrale

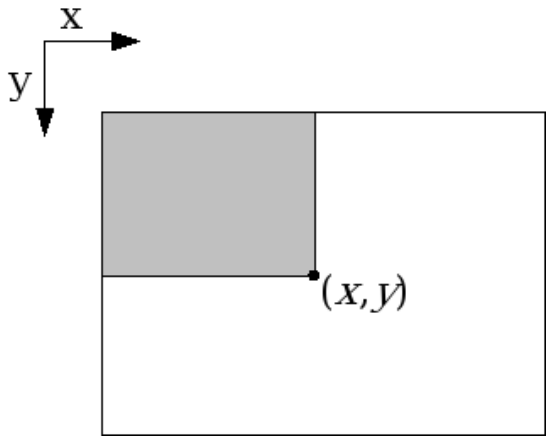


FIGURE 2.4 – Principe de l'image intégrale *La valeur du point aux coordonnées (x, y) de l'image intégrale est la somme de tous les pixels situés dans la partie grisée.*

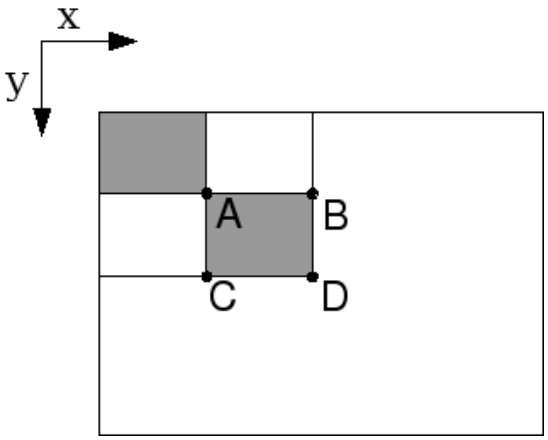


FIGURE 2.5 – Calcul d'une ondelette avec une image intégrale

Pour des objets ayant une forme complexe, il est nécessaire d'utiliser des ondelettes de multiples tailles et orientations afin d'améliorer la qualité de la description de ces objets. Le dictionnaire adéquat devient alors rapidement important et lourd à calculer. Viola et Jones [95] ont exploité les « *Summed-Area Tables* » présentées par Crow [18] pour calculer plus efficacement toutes ces ondelettes. Le principe est de passer par une représentation intermédiaire de l'image initiale appelée « image intégrale », notée $I_{\text{intégral}}$. Elle consiste à calculer pour un pixel donné de l'image, la somme des pixels situés au-dessus à gauche de celui-ci comme présenté dans la figure 2.4. Ainsi l'image intégrale contient à la position (x,y) :

$$I_{\text{intégral}}(x, y) = \sum_{x'<=x, y<=y'} I(x', y') \tag{2.5}$$

Grâce à cette représentation, une ondelette s'obtient plus rapidement. Par exemple, l'ondelette de la figure 2.5 se calcule comme la somme de 4 références de l'image intégrale :

$$h = I_{\text{intégral}}(x_A, y_A) + I_{\text{intégral}}(x_D, y_D) - (I_{\text{intégral}}(x_B, y_B) + I_{\text{intégral}}(x_C, y_C)) \tag{2.6}$$

A,B,C et D étant les quatre points de l'image.

2.3 Histogrammes de gradients orientés

2.3.1 Introduction

Les histogrammes de gradients se basent sur le calcul de gradient qui peut se faire en tout point d'une image. De façon générique, le gradient permet de calculer les variations d'une fonction par rapport aux variations de ses différents paramètres. En ce qui concerne le calcul de gradients dans les images de luminance, il s'agit de calculer la variation de l'intensité des pixels dans différentes directions. Cela revient à un calcul de gradient 1D dans les directions intéressantes (classiquement en horizontal, vertical ou diagonal). Un gradient 1D horizontal ou vertical est tout simplement un calcul de la dérivée partielle de la fonction image $I(x, y)$ (voir figure 2.6).

a b c d

FIGURE 2.6 – Calcul du module et de l'orientation du gradient
a) image initiale. b) calcul du gradient suivant x. c) calcul du gradient suivant y. d) orientation du gradient.

Les histogrammes de gradients orientés [3] permettent de calculer les occurrences des orientations du gradient dans une portion localisée de l'image. Ils ont été introduits par Dalal et Triggs [19] pour faire de la reconnaissance des formes sur des piétons dans des images fixes. Ce descripteur est aussi utilisé dans *SIFT* [4] présenté par Lowe dans [62]. Ici, l'idée est d'en exploiter la qualité mais de façon épurée afin de l'utiliser pour des applications temps réel.

3. *HOG* pour « *Histograms of Oriented Gradients* ».
4. *SIFT* pour « *Scale-Invariant Feature Transform* ».

2.3.2 Calcul des HOG

Première étape : calcul du gradient Il s'agit de calculer le gradient pour tous les points de l'image ; usuellement deux masques de dérivation sont appliqués sur l'image, un horizontal - voir équation 2.7 - et un vertical - voir équation 2.8 :

$$M_1 = [-1, 0, 1] \tag{2.7}$$

$$M_2 = [-1, 0, 1]^T \tag{2.8}$$

Pour chaque point de l'image, une approximation de la composante horizontale du gradient notée G_x et de la composante verticale notée G_y est ainsi obtenue. La plupart du temps la valeur absolue du gradient est prise car c'est le contraste entre deux régions qui importe le plus : un objet noir sur un fond blanc aura donc la même réponse qu'un objet blanc sur un fond noir.

Deuxième étape : calcul de l'orientation Une fois que le calcul du gradient a été effectué pour tous les pixels de l'image, il faut en calculer l'orientation ; celle-ci peut être définie comme étant l'angle $\theta = arctan\left(\frac{G_y}{G_x}\right)$. Classiquement $\theta \in [0, \Pi]$ (voir [87]).

Troisième étape : construction de l'histogramme L'image est divisée en plusieurs cellules. Pour chacune d'entre elles, un histogramme de gradients orientés est construit en comptabilisant les occurrences du gradient dans une barre correspondant à un intervalle d'orientation spécifique. Un exemple est montré sur la figure 2.7.
Différents types de masques ont été testés dans [19]. Il existe les R-HOG (pour *Rectangular-HOG*) des C-HOG (pour *Circular-HOG*). Pour les R-HOG, les gradients autour du pixel sont comptabilisés avec une fenêtre rectangulaire ; les C-HOG correspondent à l'utilisation d'une fenêtre circulaire autour du point considéré pour le calcul du gradient. Cette approche est toutefois peu utilisée pour des applications de reconnaissances d'objets car il requiert un temps de calcul plus conséquent.

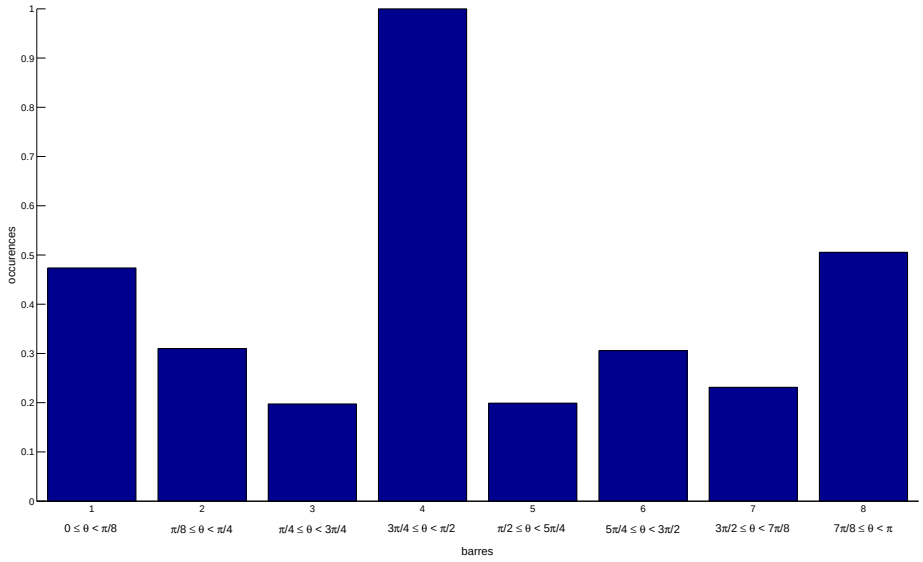


FIGURE 2.7 – Exemple d'un histogramme de gradients à 8 barres correspondant à l'image 2.6.

Diverses méthodes ont été développées. Tout d'abord, les occurrences comptabilisées peuvent être pondérées par la magnitude du gradient [6]. Ceci se justifie par le fait qu'une région uniforme se caractérise par un gradient de faible amplitude ; or c'est lorsque le gradient atteint des grandes valeurs que l'information est la plus significative car cela correspond par exemple à des coins ou des contours d'un objet.
Il faut également choisir le nombre de barres (« *bins* » en anglais) désirées par histogramme, à ajuster pour chaque application. Une barre d'histogramme correspond au nombre d'occurrences de l'orientation du gradient pour un intervalle donné. Le choix du nombre de barres influe directement sur la précision des informations : un petit nombre de barres réduit l'information disponible, mais trop de barres risque de ne pas faire ressortir une information globale intéressante.
Enfin certains préconisent la normalisation des histogrammes afin de pouvoir décrire correctement la grande variabilité des images disponibles dans une base de données. Dans [87], plusieurs termes de normalisation ont été testés. Chacun d'entre eux a été calculé pour un ensemble de cellules, appelé bloc.

2.3.3 Utilisation

Ces dernières années, de multiples versions ont été mises en œuvre [6, 76] et il paraît difficile de savoir quelle est la meilleure à utiliser pour sa propre application. Il est donc nécessaire de passer par une phase de test pour régler les nombreux paramètres des HOG (type de masque, nombre de barres de l'histogramme, etc.). Malgré cet inconvénient, ce descripteur est devenu

très populaire en reconnaissance d'objets. En effet, il fournit des bons résultats tout en étant applicable en temps réel, notamment à travers l'emploi d'images intégrales [77].

2.4 Matrices de covariance

Des approches utilisant des matrices de covariance ont récemment été développées [92, 99] avec de très bons résultats pour la vidéosurveillance. Considérons une image I de dimension $W \times H$. A partir de cette image, pour chaque pixel de coordonnées (x, y) un ensemble de caractéristiques peut être extrait comme l'intensité, le gradient... ou n'importe quelle autre information qui caractérise l'apparence de l'objet recherché. Notons d le nombre de caractéristiques calculées pour un point donné. Nous pouvons alors déterminer un ensemble $W \times H \times d$ de caractéristiques qui décrivent l'image. Regroupons dans un vecteur $\mathbf{z}_k$ les caractéristiques d'un point d'une région R de l'image (avec $R \subset I$). Cette région contient n points et un ensemble de vecteurs $\mathbf{z}_k$ avec $k \in [1, ..., n]$ est disponible. Elle est alors représentée par la matrice de covariance C_R suivante :

$$C_R = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{z}_k - \boldsymbol{\mu})(\mathbf{z}_k - \boldsymbol{\mu})^T \tag{2.9}$$

avec $\boldsymbol{\mu} = \frac{\sum_{k=1}^{n} \mathbf{z}_k}{n}$.
Les matrices de covariance, en tant que descripteur de régions, encodent une information sur la variance des caractéristiques et leurs corrélations entre elles, mais également la disposition des caractéristiques dans la région concernée si les coordonnées relatives du pixel sont présentes dans le vecteur z_k. Une méthode utilisant les images intégrales a été proposée pour les calculer plus rapidement [91]. La matrice de covariance obtenue est symétrique, et seule la partie triangulaire supérieure est conservée pour créer un nouveau vecteur de caractéristiques. La définition d'une mesure de distance entre deux matrices de covariance est aussi une problématique. Dans [92], les auteurs proposent d'utiliser une distance dans une variété riemannienne.
Différentes manières d'utiliser ces matrices de covariance sont actuellement envisagées pour de la détection de visage ou de personnes.

2.5 Descripteurs binaires

Ces dernières années, de nouveaux descripteurs d'image sont apparus pour répondre à des applications spécifiques. Des caméras bas-coût s'installent aujourd'hui dans notre environnement : surveillance de magasins, de rues, d'aéroports... mais des caméras sont également utilisées pour l'aide à la conduite. Au départ il s'agissait de caméras « passives » qui ne faisaient que filmer une scène sans autre interaction. Aujourd'hui, des programmes sont créés afin de rendre ces caméras « actives » : suivi de personnes, détection automatique d'obstacles... Dans

ce contexte, des algorithmes performants doivent être mis en place. Une des difficultés de ces processus est de réaliser les actions adéquates dans un délai imparti afin que les résultats obtenus soient utilisables. Par exemple, pour alerter un automobiliste de la présence d'un obstacle sur la route, non seulement il faut détecter correctement l'obstacle mais il faut également avertir le conducteur suffisamment tôt pour qu'il puisse agir en conséquence.
C'est dans ce contexte que les descripteurs binaires peuvent être attractifs. En effet, outre leur faible coût de calcul, ils sont aussi intéressants car une distance entre deux vecteurs binaires est aussi rapide à calculer, par rapport aux autres descripteurs. De nombreuses variantes ont été mises en place mais leur utilisation reste plus au moins confidentielle aux laboratoires qui les ont développés. C'est certes dommageable mais c'est souvent dû au fait que chaque nouveau descripteur a été mis en place pour répondre à une applications précise : limitation du matériel utilisé (processeurs), résolution des images, temps d'exécution désiré...
Certains descripteurs binaires ont été présentés pour faire de la classification avec des arbres de décisions [4]. Il s'agit de relever la présence d'une information - par exemple la présence d'un coin - et de relever la réponse binaire dans une branche de l'arbre. Chaque branche de l'arbre représente une portion de l'image et au fur et à mesure des réponses obtenues, la classe à laquelle appartient l'objet peut être déterminée. L'inconvénient de ces méthodes est qu'elles ont un coût énorme en temps de calcul. Des variantes ont été proposées [25, 59] pour stocker les informations de ces arbres dans un vecteur de caractéristiques qui peut être fourni à un classifieur plus rapide (de type AdaBoost ou SVM). Mais ces techniques supposent que l'objet à identifier ne subisse que peu de variations pour comparer les mêmes informations d'un objet à l'autre. Or ce n'est pas le cas pour de la reconnaissance de piétons, où la posture, les vêtements sont tellement changeants qu'il est difficilement envisageable de mettre en place une telle méthode.
C'est d'ailleurs ces raisons qui nous ont poussés à développer notre propre descripteur (voir paragraphe 2.5.3) en s'inspirant de ceux présentés dans les paragraphes suivants : les *Local Binary Patterns* (voir section 2.5.1) et les *points de contrôle* (voir section 2.5.2)

2.5.1 *Local Binary Patterns (LBP)*

Introduit en 1996 par Ojala et al. [71] puis complété dans [72], ce type de descripteur caractérise le voisinage d'un point de l'image en calculant la différence du niveau de gris du pixel considéré avec les niveaux de gris des pixels situés dans le voisinage (voir figure 2.8).

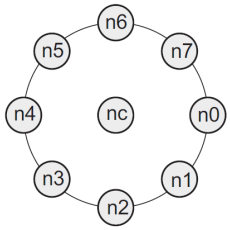


FIGURE 2.8 – *Local Binary Patterns*
Un mot binaire est créé en comparant successivement l'intensité du pixel n_c avec les pixels $n_{i=0,...,7}$ du voisinage.

Pour calculer ce descripteur, un cercle de centre n_c et de rayon R (voir figure 2.8) est considéré. N points équi-répartis sont alors sélectionnés sur ce cercle (voir figure 2.8). Ce descripteur renvoie un mot binaire dont chaque bit correspond à la comparaison du pixel central n_c avec un pixel voisin n_i ($i = 1...N$) ; si le niveau de gris du pixel voisin est supérieur ou égal, la valeur est mise à 1, et dans le cas contraire à 0, soit :

$$LBP_{R,N}(x,y) = \sum_{i=0}^{N-1} s(n_c - n_i)2^i \text{ avec } s(x) = \begin{cases} 1 & \text{si} \quad x \geq 0, \\ 0 & \text{sinon.} \end{cases} \tag{2.10}$$

Ce descripteur est largement utilisé en reconnaissance de formes [3], notamment en catégorisation d'images [41]. La catégorisation d'images est un domaine qui cherche à reconnaître un objet particulier dans une image afin de lui affecter une classe précise. En général ce type de traitement est utilisé pour classer de grandes bases de données ou faire de la fouille de données (comme la recherche d'images sur le web).
L'inconvénient de ce descripteur est le nombre de paramètres à fixer. Effectivement, les versions de ce descripteur sont nombreuses, suivant la valeur fixée N correspondant au nombre de pixels voisins et suivant le rayon R choisi. Généralement ces deux paramètres sont réglés de façon empirique : couramment N est fixé à huit et R à de petites valeurs. Toutefois, pour de grandes bases de données, le réglage optimum devient difficile à trouver. Et le passage à une application temps réel est difficile si le nombre N de pixels voisins à considérer est important.
Néanmoins, l'avantage de ce descripteur est la notion de comparaison binaire. En effet, cela permet d'allier une bonne description d'image à une facilité de calcul. De plus, le stockage mémoire des informations, en particulier pour de grandes bases de données, est d'autant plus réduit. Etant donné que le but de cette thèse est de faire de la reconnaissance de piétons, nous allons avoir recours à de grandes bases de données. Ces aspects sont donc à prendre en compte pour construire notre propre descripteur adapté à cette application.

2.5.2 Points de contrôle

Dans [1], les auteurs ont mis au point un descripteur appelé « points de contrôle ». Pour construire ce descripteur, deux listes de points sont arbitrairement constituées de façon à ce que

n'importe quel point d'une liste soit séparé d'une distance minimum K de tous les points de l'autre liste. Ensuite l'intensité des pixels est comparée. L'information retournée est binaire : si tous les points de la liste 1 ont une intensité supérieure à ceux de la liste 2, le descripteur répond 1, et 0 sinon.
Ce descripteur permet donc de s'affranchir du calcul par zones d'images qu'utilisent les ondelettes de Haar puisqu'il se place au niveau du pixel. Le but recherché est donc de créer un descripteur rapide, particulièrement bien adapté aux applications temps réel. De plus, dès qu'un point de la première liste a une intensité inférieure à celui de la deuxième liste, l'information retournée vaut 0 et il est inutile de tester les autres points de contrôle.
Il faut maintenant déterminer quels points vont être sélectionnés pour construire les deux listes. Nous avons une contrainte sur la répartition spatiale des points mais les possibilités sont encore trop importantes pour obtenir un descripteur rapide. Il est donc proposé de sélectionner ces points par un algorithme de *Boosting* (de type AdaBoost), ce qui permet de sélectionner de bonnes caractéristiques mais aussi de créer dans le même temps un classifieur.
D'abord utilisés pour faire de la reconnaissance de visages, ces « points de contrôle » ont été modifiés et étendus à des applications de reconnaissance de piétons ou de voitures [67].

2.5.3 Comparateur de niveaux de gris

Dans le projet *LOVe*, les images de luminance fournies par le système de vision proviennent de caméras standards ; elles ont une taille de 640×480 pixels et montrent la globalité de la scène devant le véhicule. Les piétons présents dans cet espace ne sont alors représentés que par quelques centaines de pixels (voir paragraphe 5.3.3). Nous avons donc développé notre propre descripteur de niveaux de gris afin de gérer au mieux ces contraintes. L'utilisation de descripteurs de grande précision n'est pas requise puisque l'information décrivant le piéton est réduite.
Nous avons alors mis en place un descripteur de niveau de gris en s'inspirant des descripteurs binaires précédents (voir section 2.5) ; la différence se situe au niveau de la stratégie de sélection des points à comparer. Les *Local Binary Patterns* sélectionnent un ensemble de points autour du pixel considéré, tandis que les *points de contrôle* ne font que des comparaisons entre des couples de points séparée par une distance minimale K à fixer. Dans notre cas, nous considérons tout d'abord des comparaisons entre des couples de points pouvant se situer dans toute l'image, même si des stratégies seront ensuite mises en œuvre pour sélectionner les comparaisons les plus pertinentes (voir la fin du paragraphe).
Ce descripteur consiste donc en une comparaison du niveau de gris de deux points appartenant à l'image. Considérons deux points p_1 et p_2. Les intensités des niveaux de gris en ces points sont notées $I(p_1)$ et $I(p_2)$; le descripteur réalise la comparaison suivante :

$$d = (I(p_1) \geqslant I(p_2)) \tag{2.11}$$

La réponse retournée d est binaire : si la condition est vérifiée alors $d = 1$ sinon $d = 0$.

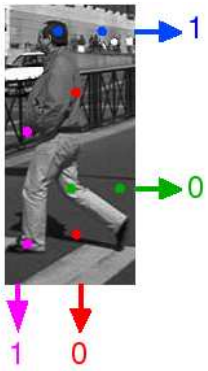


FIGURE 2.9 – Descripteur de niveaux de gris

En sélectionnant des points de comparaison sur toute l'image, l'objet qu'elle contient est ainsi décrit entièrement. En tant que tel, ce descripteur est à une seule dimension et n'est pas invariant aux changements de position ou d'échelle. Par contre, il est invariant aux changements affines de luminance. Toutefois ce descripteur est utilisé en amont d'un classifieur. Grâce à une segmentation d'image adéquate, l'objet à reconnaître remplit entièrement la portion d'image sélectionnée et il n'y a pas lieu d'utiliser un descripteur ayant des propriétés d'invariance en position. Pour décrire des images de piétons, nous choisissons d'utiliser des couples de points qui appartiennent soit à la même ligne, soit à la même colonne. Cela se justifie car les piétons à caractériser se tiennent debout devant un véhicule et dans cette situation les principaux axes de symétrie sont alors horizontaux et verticaux.
Il est tout de même indispensable de faire une sélection parmi tous les couples de points possibles. En effet, conserver la totalité ou même un grand nombre de ces comparaisons risque de nous pénaliser, tout d'abord en termes de temps de calcul (voir tableau 2.1). Le risque est alors de ne pas respecter le temps réel pour l'application finale visée (détecter et reconnaître des piétons avec une caméra embarquée dans un véhicule). Ensuite, en sélectionnant beaucoup de données, il est possible d'avoir des données redondantes. Or il a été démontré [13] que leur présence peut nuire au bon fonctionnement du classifieur qui exploite ces données.

taille de l'image		nombre de comparaisons		
hauteur	largeur	totales	horizontales	verticales
h	l	C^2_{h*l}	$h * C^2_l$	$l * C^2_h$
36	18	209628	5508	11340
640	480	$> 47.10^9$	$> 73.10^3$	$> 98.10^3$

TABLE 2.1 – Nombres de comparaisons possibles en fonction de la taille de l'image

Néanmoins, il faut en sélectionner un nombre suffisant afin que le classifieur ait assez d'informations pour classifier correctement l'objet. Et c'est là tout le dilemme : peu de comparaisons

pour ne pas perturber le classifieur mais suffisamment pour créer un bon système de reconnaissance. Pour choisir ces couples de points, nous mettrons donc en place en amont du classifieur une phase de sélection de variables pour ne conserver que des variables pertinentes pour la classification. Cette sélection nous permettra de réduire le temps de calcul du descripteur lors de l'application temps réel puisque nous limiterons les calculs aux seuls points conservés pendant cette phase. Cette étape de sélection de variables sera plus amplement décrite dans les sections 4.6 et 4.7.
Ce descripteur diffère des descripteurs binaires précédemment présentés. Tout d'abord il est relativement simple et rapide à mettre en œuvre. De plus, en sélectionnant des couples de points répartis sur toute l'image, nous nous assurons d'explorer toute l'image. Il suffit d'effectuer les comparaisons pour tous les points désirés de l'image et de concaténer ces informations binaires dans un vecteur de caractéristiques. Celui-ci sera donc la description de l'image et sera ensuite utilisé par le classifieur pour déterminer la classe de l'objet présent.

2.6 Descripteurs mis en œuvre dans la thèse

Dans le cadre de la reconnaissance, les descripteurs présentés précédemment (ondelettes de Haar et histogrammes de gradients) sont certainement les plus utilisés car ils offrent un bon compromis entre rapidité d'exécution et performances. Un descripteur utilisé pour faire de la reconnaissance doit apporter des réponses semblables entre objets intra-classes et différentes inter-classes. Le classifieur s'appuiera sur ces informations pour calculer la distance entre les objets et ainsi définir un modèle de classification. A l'heure actuelle, des solutions sont cherchées pour utiliser de nouveaux descripteurs et améliorer les résultats de reconnaissance. Les descripteurs binaires sont une piste pour accélérer les temps d'exécution tout en gardant les mêmes performances.
En ce qui concerne cette thèse, le fait de séparer le descripteur du classifieur nous permet de rester dans une logique de « modules », inhérentes au projet *LOVe*, ce qui a des bénéfices. Effectivement, le descripteur n'étant pas lié au classifieur, il est ainsi possible d'interchanger soit le descripteur, soit le classifieur pour tester et réaliser la meilleure chaîne logicielle. Plusieurs descripteurs sont essayés : ondelettes de Haar et histogrammes de gradients orientés - car ils sont significatifs de l'état de l'art actuel sur la reconnaissance de piétons pour l'aide à la conduite - et notre descripteur binaire. Dans le chapitre 4, leurs performances sont comparées (voir section 4.4) et une combinaison de ces trois descripteurs sera présentée (voir section 4.8).

Chapitre 3

Apprentissage et classification

Nous disposons d'un ensemble de données ; chacune d'entre elles est formée d'une observation (vecteur de descripteurs) et d'une étiquette de classe. Le but d'un classifieur est d'établir une relation entre ces observations et les étiquettes pendant une phase d'apprentissage. Cette étape doit permettre au classifieur de généraliser, c'est-à-dire d'attribuer une étiquette adéquate à un nouvel exemple dont seul le vecteur d'observation est connu.
De nombreuses méthodes de classification existent, chacune répondant à des problématiques bien spécifiques (voir 1.4). Dans ce chapitre, nous exposons les méthodes utilisées et mises en place dans le cadre de cette thèse. Il s'agira de méthodes de classification par apprentissage supervisé, ceci afin de réaliser l'application finale qui est de créer un outil de reconnaissance de piétons en temps réel.

3.1 Les différents types d'apprentissage

Les méthodes d'apprentissage peuvent être classées en plusieurs catégories : apprentissage supervisé, apprentissage semi-supervisé, apprentissage non-supervisé et apprentissage par renforcement.
Nous allons nous positionner dans le cas d'une tâche de classification. C'est ce dont nous avons besoin dans le cadre de ma thèse pour réaliser la reconnaissance de piétons. Le but est d'associer à un objet donné une étiquette correspondant sa classe.

Apprentissage supervisé L'apprentissage supervisé a pour but d'apprendre par l'exemple. Il faut fournir au préalable une liste d'objets avec leurs étiquettes de classe - appelée ensemble d'apprentissage - afin que le système soit capable d'expliquer et ensuite de prédire l'appartenance d'un nouvel objet à une classe connue *a priori*. Beaucoup d'algorithmes d'apprentissage supervisé sont utilisés pour faire de la reconnaissance d'objets : caractères, visages, personnes... La méthode la plus classique est certainement l'algorithme des « *K-Plus Proches Voisins* » (KPPV) [21]. Le principe est simple et consiste à calculer la distance d'un nouvel objet par

rapport à ceux dont la classe est déjà connue, par exemple avec une distance euclidienne ; le nouvel objet appartiendra à la classe dont il est le plus proche. Cette technique donne d'assez bons résultats dans des cas simples et est facile à mettre en œuvre car elle est non-paramétrique. Toutefois le temps de calcul de la prédiction est assez long car il nécessite un calcul de distance à tous les éléments de la base d'apprentissage. La figure 3.1 montre un exemple simple d'une classification par un algorithme des KPPV avec un calcul de distance euclidienne : le nouvel exemple est associé à la classe B car la majorité de ses plus proches voisins appartiennent à la classe B (au sens de la norme euclidienne).

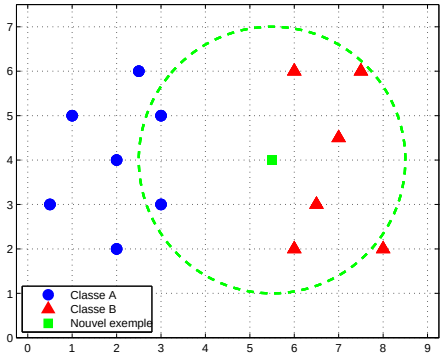


FIGURE 3.1 – Exemple simple de classification par un algorithme *K-Plus Proches Voisins* avec un calcul de distance par la norme euclidienne
Les distances entre le nouvel exemple et les différents points des deux classes sont calculées. Comme ses plus proches voisins appartiennent à la classe B, il est alors étiqueté comme appartenant à cette classe.

Il existe d'autres méthodes non-paramétriques - comme les fenêtres de Parzen [21] - qui utilisent uniquement les données d'apprentissage, mais aussi de nombreuses méthodes qui se basent sur un modèle paramétrique pour établir une règle de classification ; il s'agit par exemple de l'estimateur de Bayes, des réseaux de neurones... [21]. Le *Boosting* et les classifieurs à noyau utilisés dans cette thèse en sont également ; leur fonctionnement est expliqué dans la suite de ce chapitre (voir sections 3.3 et 3.4).

Apprentissage semi-supervisé Ce terme regroupe des méthodes qui se situent entre l'apprentissage non-supervisé et l'apprentissage supervisé. Ce type de méthodes est utilisé quand un grand nombre de données est disponible mais sans qu'elles soient toutes étiquetées. L'initialisation de la méthode est faite à partir d'un petit jeu de données correctement étiquetées. Puis l'algorithme doit lui-même étiqueter les exemples suivants et construire son propre modèle.
Les algorithmes d'apprentissage non-supervisé et semi-supervisé sont beaucoup utilisés pour la recherche d'informations sur internet notamment. Ils permettent de traiter ainsi une grande

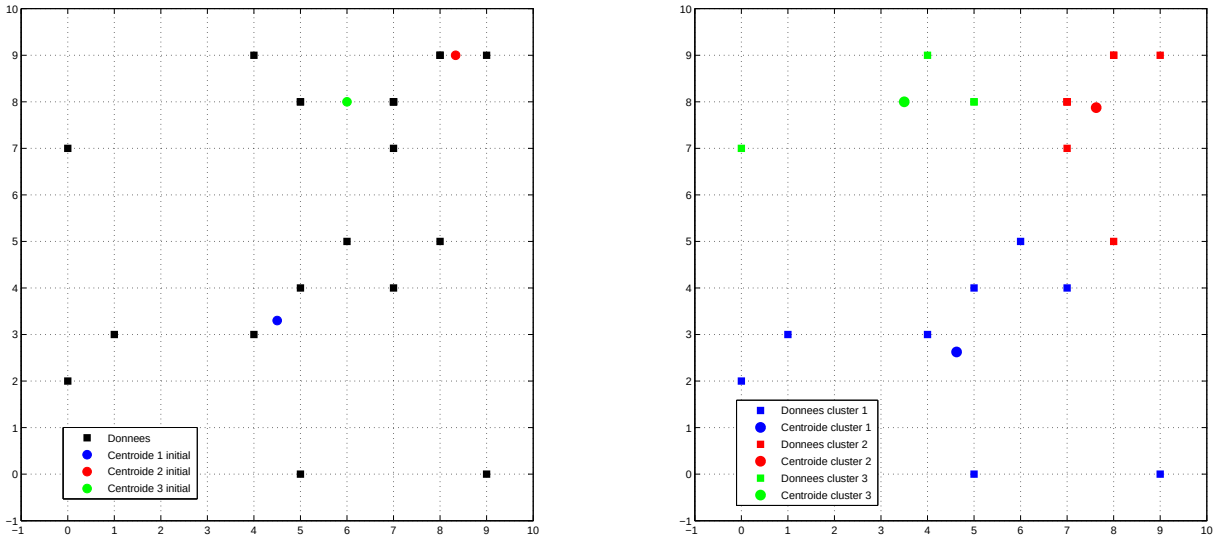


FIGURE 3.2 – Exemple simple de classification non-supervisé par un algorithme des *K-means* *a- Données de départ et initialisation de* 3 *centroïdes b- Résultat final*.

quantité de données.

Apprentissage non-supervisé Pour l'apprentissage non-supervisé, l'ensemble d'apprentissage est seulement composé d'exemples, sans aucune étiquette de classe. C'est à l'algorithme de trouver des dépendances, des structures entre les différents exemples. Le « *clustering* » ou partitionnement de données regroupe un ensemble de méthodes d'apprentissage non-supervisé, comme l'algorithme des *K-means* [21] ou l'*Isodata* [46]. Les classes (ou « *clusters* » en anglais) sont créées par l'algorithme qui regroupe dans une même classe des objets ayant des caractéristiques communes entre elles et différentes avec les objets n'appartenant pas aux mêmes classes. Prenons un exemple de classification par les *K-means*. L'algorithme est initialisé aléatoirement avec un certain nombre de clusters pour lesquels un point moyen, appelé centroïde, est évalué. A chaque itération, la distance entre chaque exemple aux différents centroïdes est calculée ; chaque exemple est alors associé au cluster dont la distance au centroïde est la plus proche. Puis les centroïdes sont réévalués. L'algorithme se termine lorsqu'il n'y a plus aucun changement. Un exemple simple de fonctionnement est montré sur la figure 3.2.

Apprentissage par renforcement Ce type de méthode est un apprentissage interactif. A chaque décision que l'algorithme prend, il reçoit en retour des réponses de l'environnent appelées signaux de renforcement. C'est un processus adaptatif qui améliore la solution en fonction des réponses qu'il reçoit. L'algorithme de « *Q-learning* » [97] réalise un apprentissage par renforcement. Il produit une matrice Q dont chaque élément $Q(s, a)$ est une mesure de l'inté-

rêt d'effectuer l'action a lorsque le système se trouve dans l'état s. Par ailleurs, des résultats théoriques garantissent, dans des cas précis, la convergence de l'algorithme vers des valeurs optimales de Q. Ces algorithmes par renforcement permettent d'établir des processus plus complexes, comme le guidage de robots. Ce dernier a un objectif final à atteindre et en fonction des réponses de ses différents capteurs, il va pouvoir affiner ses actions.

3.2 L'apprentissage supervisé pour deux classes d'objets

3.2.1 La base d'apprentissage

Pendant l'apprentissage, le but recherché est d'entraîner une machine à distinguer des objets appartenant à différentes classes. Dans le cas de la classification entre deux classes, il s'agit de différencier un type d'objet particulier par rapport à tout ce qui n'est pas cet objet ; les exemples positifs correspondent à la classe d'objets recherchés, tandis que les exemples négatifs correspondent à tout ce que n'est pas cet objet.
Pour réaliser un apprentissage, une base de données, appelée base d'apprentissage, est disponible ; elle contient un grand nombre d'exemples positifs et négatifs (plusieurs milliers). Le choix de la base d'apprentissage n'est pas trivial. Il faut tout d'abord choisir des images en rapport avec l'application visée. Par exemple, pour détecter des piétons dans un environnement urbain, les exemples positifs doivent également être sélectionnés dans le même contexte et les exemples négatifs doivent représenter ce que le système est susceptible de trouver dans celui-ci (véhicules, mobilier urbain...). En effet, si le classifieur est entraîné sur des images de personnes prises en intérieur, avec des exemples négatifs ne correspondant pas au mobilier urbain, le classifieur ne sera pas préparé au cas de figure extérieur, et ses performances seront moindres par rapport à un apprentissage sur des données prises en extérieur.
Un autre point important est le nombre d'images prises dans la base d'apprentissage. Pour les applications où les deux classes d'objets regroupent des objets de grande diversité (par exemple, pour la classe *piétons*, la taille, les vêtements... changent d'une personne à l'autre), il faut souvent prendre des milliers d'exemples.

3.2.2 Le classifieur

Chaque exemple de cette base d'apprentissage est caractérisé par un descripteur comme ceux présentés dans le chapitre 2. Puis le classifieur est entraîné sur ces données. Son rôle est de déterminer les caractéristiques communes aux exemples d'une même classe afin de pouvoir ultérieurement reconnaître à quelle classe appartient un nouvel exemple inconnu.
Le nombre de caractéristiques représentant un objet est aussi un point important pour assurer un apprentissage valide. Effectivement, si un objet n'est pas suffisamment décrit, le risque d'erreur de classification est important. A l'inverse, un classifieur peut aussi avoir dû mal à gérer un trop

grand nombre d'informations. Ce phénomène s'appelle le « surapprentissage » [1]. Ce cas survient lorsque le classifieur se focalise sur des exemples « difficiles » qui sont restés mal classés jusque là. Le classifieur, cherchant à coller au mieux aux données d'apprentissage, va perdre sa capacité à généraliser.

Parmi toutes les méthodes existantes, citons l'apprentissage par réseau de neurones qui est inspiré du fonctionnement des neurones biologiques. Le premier algorithme de ce type est le perceptron de Rosenblatt [79]. Celui-ci ne comporte qu'un neurone qui reçoit en entrée toutes les caractéristiques d'un objet. Il effectue une combinaison linéaire de ces dernières grâce à une pondération par un vecteur poids. Le résultat obtenu est comparé à un seuil et l'exemple est associé à une classe ou l'autre en fonction de cette comparaison. Les SVM sont largement inspirés de ce principe (voir section 3.4.2). Les réseaux de neurones, apparus avant les SVM, découlent plus directement de cet algorithme du perceptron [80]. Il s'agit d'utiliser non pas un mais plusieurs neurones, répartis sur plusieurs couches. Les neurones d'entrée reçoivent les données brutes, les neurones cachés constituent le réseau, et les neurones de sortie reçoivent les réponses de tout le réseau pour fournir la réponse finale. Lorsqu'un neurone est activé - c'est-à-dire qu'il reçoit des données - il calcule une combinaison linéaire avec son propre vecteur poids et sa réponse est envoyée à la couche suivante. La réponse finale est donnée par le neurone de sortie qui obtient la valeur la plus grande. Il existe de nombreuses variantes de cette version de base. La difficulté majeure de ces réseaux de neurones est de choisir l'architecture du réseau : nombre de couches, de neurones par couches, etc. Certaines heuristiques simples peuvent aider à construire le réseau mais il n'existe pas à ce jour une méthode pour déterminer le réseau optimal, si ce n'est une recherche exhaustive.

Un autre type de classification usuelle se base sur la théorie de Bayes comme le classifieur naïf de Bayes. Toutes les caractéristiques des exemples d'apprentissage sont supposées indépendantes. Les probabilités *a priori* de chaque classe sont déterminées sachant ces observations. Puis, pour un exemple inconnu, la probabilité *a posteriori* des classes est calculée. La règle du maximum *a posteriori* permet alors d'associer ce nouvel exemple à la classe la plus probable. En pratique les données ne sont pas toujours indépendantes, mais ce classifieur donne souvent des résultats très satisfaisants.

Dans ce chapitre, nous présentons des méthodes de classification plus récentes comme le *Boosting* (voir section 3.3) et les machines à noyau (voir section 3.4) qui se basent sur la notion de séparatrice non-linéaire dans l'espace des descripteurs.

Etablir une classification entre deux classes revient à trouver géométriquement une séparation entre la classe des positifs et celle des négatifs. Si l'espace de description des données est en 1D, il s'agit tout simplement d'un seuil, en 2D d'une droite, et pour des espaces de plus grande dimension, c'est un hyperplan.

1. *overfitting* en anglais.

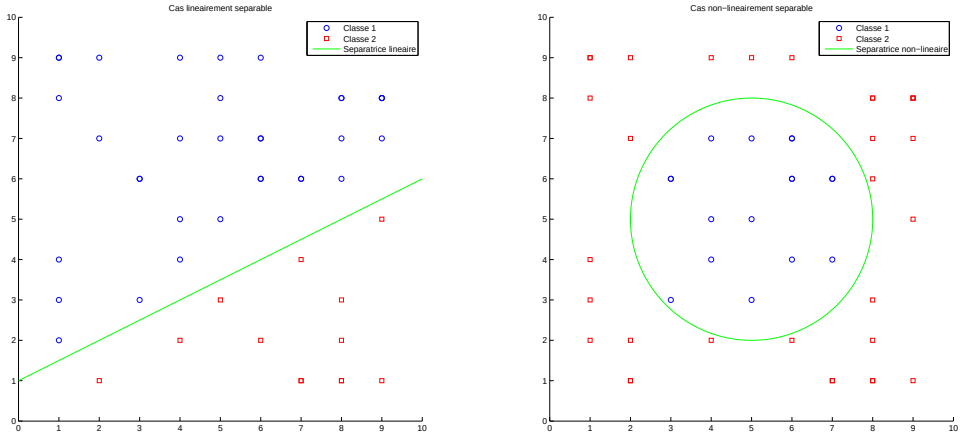


FIGURE 3.3 – Séparatrices linéaire et non-linéaire
A gauche, cas linéairement séparable : la séparatrice est une droite ; à droite, cas non-linéairement séparable : la séparatrice est un cercle.

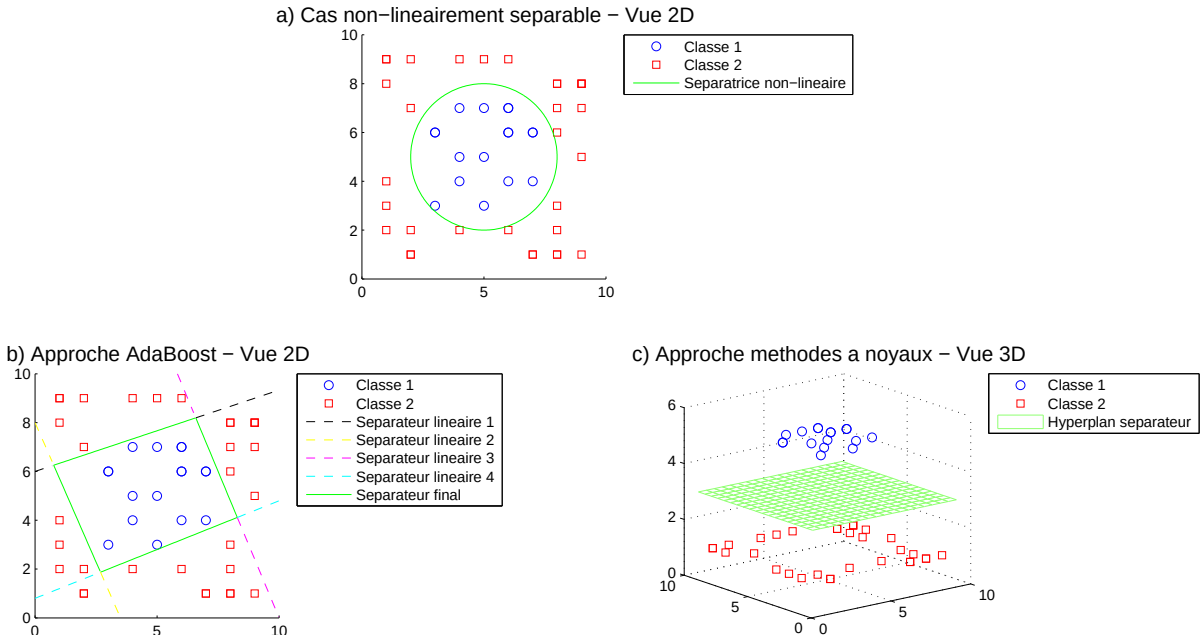


FIGURE 3.4 – Approche AdaBoost et méthodes à noyau pour un cas non-linéaire
a) cas de données non-linéairement séparables ;
b) l'AdaBoost propose une combinaison de séparatrices linéaires ;
c) les méthodes à noyau comme les SVM déterminent un hyperplan séparateur en transposant les données dans un espace de plus grande dimension où le problème devient linéairement séparable.

Mais pourquoi chercher une séparatrice linéaire ? Parce que c'est plus simple à déterminer. Cependant les données d'apprentissage ne sont pas toujours linéairement séparables. Les méthodes que nous proposons permettent de pallier à ce problème. Les méthodes de *Boosting* résolvent le problème par combinaison de plusieurs séparatrices linéaires. Quant aux méthodes à noyau, elles transposent les données dans un espace de plus grande dimension où il existe un hyperplan séparateur. C'est ce qui est appelé les « *kernel tricks* ».
Lorsqu'un apprentissage est réalisé, il faut l'évaluer afin de voir s'il est capable ou non de généraliser et de reconnaître le plus grand nombre d'images inconnues, positives ou négatives. Pour cela, une base de validation ou de test est usuellement utilisée ; elle contient des images totalement différentes et indépendantes de celles contenues dans la base d'apprentissage. Le classifieur est lancé sur ces images et sa réponse est analysée pour toutes les images. Il existe de nombreuses façons d'évaluer les performances d'un classifieur, sur lesquelles nous reviendrons au chapitre suivant (voir section 4.2.2).

3.3 Méthodes de *Boosting*

Les méthodes de *Boosting* sont des méthodes très populaires car elles sont généralement simples et faciles à mettre en œuvre. L'idée principale est qu'une combinaison de plusieurs classifieurs dits faibles peut fournir un classifieur dit fort. Un classifieur faible est un classifieur qui fournit des résultats peu performants, au minimum légèrement meilleurs que le hasard. L'objectif, en regroupant plusieurs classifieurs faibles, est de créer un classifieur fort, c'est-à-dire un classifieur qui classifie correctement tout l'ensemble d'apprentissage. Le classifieur ainsi obtenu a d'autant plus de chances de mieux généraliser et de classifier correctement un nouvel exemple.

3.3.1 Le *Boosting*

Le principe a été mis en place par Schapire [81]. Dans cet article, il a montré qu'un algorithme d'apprentissage qui produit un classifieur faible peut être amélioré s'il est entraîné sur trois échantillons choisis parmi l'ensemble d'apprentissage initial. La combinaison des trois règles permet d'obtenir une règle de décision forte.
Soit $\mathcal{S}$ l'ensemble d'apprentissage comprenant N exemples. Nous nous plaçons dans le cas d'un problème de classification en deux classes. L'algorithme fournit à chaque apprentissage une hypothèse h, c'est-à-dire un scalaire indiquant à quelle classe appartient l'objet.

1. l'algorithme d'apprentissage est entraîné sur un échantillon S_1 comprenant n_1 exemples sélectionnés dans S_1 (avec $n_1 < N$) ; un premier classifieur faible C_1 est obtenu ; il fournit une première règle de décision h_1 ;
2. un échantillon de n_2 exemples dans l'ensemble $(\mathcal{S} - \mathcal{S}_1)$ est sélectionné de telle sorte que parmi ces échantillons la moitié soit correctement classée par C_1, l'autre non. Un deuxième classifieur faible C_2 est alors obtenu ; il produit une règle de classification h_2 ;

3. enfin un classifieur faible C_3 est créé en l'entraînant sur un échantillon S_3 comprenant n_3 exemples sélectionnés dans $(\mathcal{S} - \mathcal{S}_1 - \mathcal{S}_2)$ pour lesquels C_1 et C_2 ne fournissent pas la même réponse ;
4. le classifieur final C a pour règle de décision H telle que :
   $H =$ vote majoritaire(h_1, h_2, h_3).

Un exemple simple de cette méthode est fourni figures 3.5, 3.6 et 3.7 (exemple tiré du livre d'A. Cornuéjols et L. Miclet [17]). Bien sûr, les trois échantillons $\mathcal{S}_1$, $\mathcal{S}_2$ et $\mathcal{S}_3$ tendent à recouvrir entièrement $\mathcal{S}$ afin de tirer le maximum d'informations de tout l'ensemble d'apprentissage. En pratique, il arrive souvent que le premier classifieur utilisé classe correctement un maximum d'exemples. Le découpage en sous-échantillons se fait généralement de façon empirique, en fonction du classifieur et des données présentes. Le processus présenté ci-dessus peut être réalisé de façon récursive pour chaque classifieur faible et ainsi traiter de 9 voire 27 sous-ensembles. Toutefois, avec ce type d'apprentissage, il est envisageable de remplacer la fonction de test par une distribution de probabilités sur tout l'ensemble d'apprentissage afin d'obtenir une meilleure généralisation. La méthode qui suit reprend précisément cette idée.

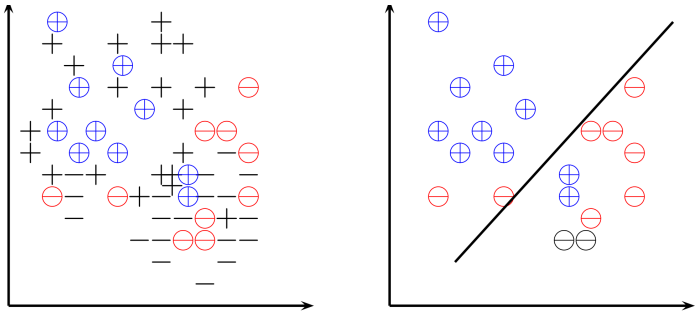


FIGURE 3.5 – Exemple de *Boosting* sur un exemple simple - première étape [17]
A gauche, l'ensemble d'apprentissage $\mathcal{S}$ comprend tous les exemples positifs (+) et négatifs (-). Le sous-ensemble $\mathcal{S}_1$ est constitué d'un échantillon de $\mathcal{S}$ (points cerclés en bleu pour les exemples positifs et en rouge pour les négatifs).
A droite, l'ensemble $\mathcal{S}$ et la droite C_1 apprise sur celui-ci.

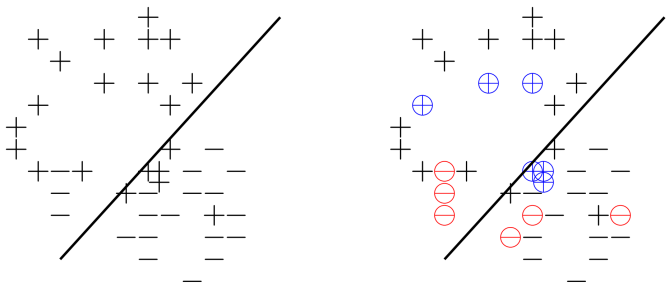


FIGURE 3.6 – Exemple de *Boosting* sur un exemple simple - deuxième étape [17]
A gauche, l'ensemble $\mathcal{S} - \mathcal{S}_1$ et la droite C_1 apprise sur $\mathcal{S}_1$.
A droite, un ensemble $\mathcal{S}_2$ (points cerclés) inclus dans $\mathcal{S} - \mathcal{S}_1$, comprenant les exemples les plus informatifs pour C_1.

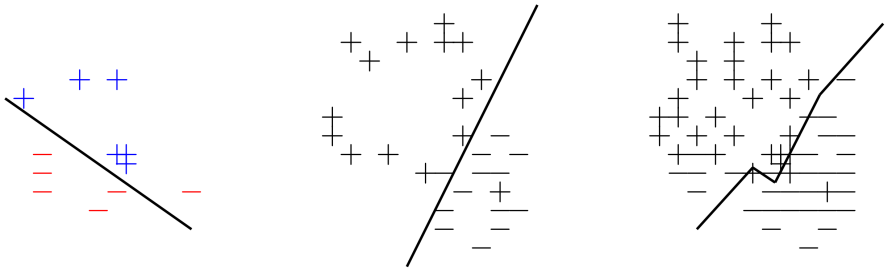


FIGURE 3.7 – Exemple de *Boosting* sur un exemple simple - dernière étape [17]
A gauche, l'ensemble $\mathcal{S}_2$ et la droite C_2 apprise sur celui-ci.
Au centre, l'ensemble $\mathcal{S}_3 = \mathcal{S} - \mathcal{S}_1 - \mathcal{S}_2$ et la droite C_3 apprise sur $\mathcal{S}_3$.
A droite, l'ensemble d'apprentissage $\mathcal{S}$ et la combinaison des 3 droites, qui permet de classifier correctement tous les exemples.

3.3.2 *Adaptative Boosting* (AdaBoost)

Introduit en 1996 par Freund et Schapire [27], l'AdaBoost se base sur le principe fondamental du *Boosting* et énonce que l'avis de plusieurs experts est meilleur que celui d'un seul. Le but est donc là encore d'associer plusieurs classifieurs faibles afin de créer un classifieur fort. La nouveauté introduite par l'AdaBoost est de proposer une distribution de probabilités *a priori* sur tout l'ensemble d'apprentissage en fonction de la réponse de l'algorithme à l'itération précédente. De cette façon, il est facile d'intégrer un grand nombre de classifieurs faibles dans la création du classifieur final.
Notons $\mathcal{S}$ l'ensemble d'apprentissage composé de N exemples tel que : $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1,...N}$, $\mathbf{x}_i$

étant le vecteur de caractéristiques d'un exemple et y_i étant un scalaire indiquant la classe de l'objet. Dans le cas présent d'une classification en deux classes, $y_i = 1$ si l'objet appartient à la classe d'objet recherché (exemple positif), et $y_i = -1$ sinon (exemple négatif). Notons $\mathbf{p}_t$ un vecteur poids dont chaque composante est le poids associé à chaque exemple ; initialement $p_t = p_1$ et tous les éléments ont la même valeur. Un échantillon, noté S_1 et comprenant n_1 exemples (avec $n_1 < N$), est ensuite sélectionné parmi l'ensemble d'apprentissage. Un classifieur C_1 est entraîné sur cet échantillon, qui fournit une hypothèse h_1 correspondant directement à la classe de l'objet. Cette règle de décision est ensuite appliquée pour tous les exemples compris dans S afin de calculer l'erreur ϵ_1 du classifieur C_1 sur l'ensemble $\mathcal{S}$. Le poids correspondant à chaque exemple est ensuite mis à jour en fonction du résultat de la classification. Si un exemple est bien classé, son poids diminue et s'il est mal classé son poids augmente. Pour cela, un coefficient α_1 est calculé tel que :

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) \tag{3.1}$$

Le poids des différents exemples vaut :

$$\begin{aligned} p_2(\mathbf{x}_i) &= \frac{p_1(\mathbf{x}_i)}{Z_1}\mathrm{e}^{-\alpha_1} \quad \text{si} \quad h_1(\mathbf{x}_i) = y_i \\ p_2(\mathbf{x}_i) &= \frac{p_1(\mathbf{x}_i)}{Z_1}\mathrm{e}^{+\alpha_1} \quad \text{si} \quad h_1(\mathbf{x}_i) \neq y_i \end{aligned} \tag{3.2}$$

Z_1 est un terme de normalisation tel que $\sum_{i=1}^{m} p_1(\mathbf{x_i}) = 1$.
Après cette étape de mise à jour des poids, un nouvel échantillon d'exemples est sélectionné. Les exemples choisis en priorité sont ceux dont les poids correspondants sont les plus élevés. Puis un nouveau classifieur est entraîné sur cet échantillon. De cette façon, ce nouveau classifieur créé se focalisera sur les exemples qui sont jusque là mal classés. Les poids sont alors remis à jour et le processus est réitéré t fois (avec $t < T$) tant que l'erreur globale sur tout l'ensemble d'apprentissage ϵ_T n'est pas nulle. Un ensemble de T classifieurs faibles est alors obtenu, qui constitue un classifieur fort C. A la fin de l'algorithme, chaque classifieur faible voit sa règle de décision pondérée par une valeur α_t calculée au cours de l'apprentissage ; pour la classification d'un nouvel exemple, le classifieur fort fournit ainsi l'hypothèse suivante :

$$H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \tag{3.3}$$

Pour un objet ayant pour vecteur de caractéristiques $\mathbf{x}$, si $H(\mathbf{x}) \geq 0$, l'objet est associé à la classe des positifs, et si $H(\mathbf{x}) < 0$ il est associé à la classe des négatifs. L'algorithme ainsi créé est résumé dans l'algorithme 1. Un exemple de classification est présenté figures 3.8, 3.9 et 3.10 (exemple tiré de [48]).
En pratique, un seuil est fixé et lorsque l'erreur globale sera en deçà de celui-ci, l'algorithme arrête d'ajouter de nouveaux classifieurs faibles. En effet, lorsqu'il ne reste que quelques exemples

mal classés, les nouveaux classifieurs ajoutés vont exclusivement se focaliser sur ces exemples et n'apporteront pas une réelle amélioration sur la règle de décision finale. Le but étant de créer un classifieur fort capable de généraliser correctement pour de nouveaux exemples inconnus, ajouter des classifieurs faibles seulement pour quelques exemples n'est pas utile et risque même de détériorer les résultats du classifieur fort final (problème de surapprentissage).

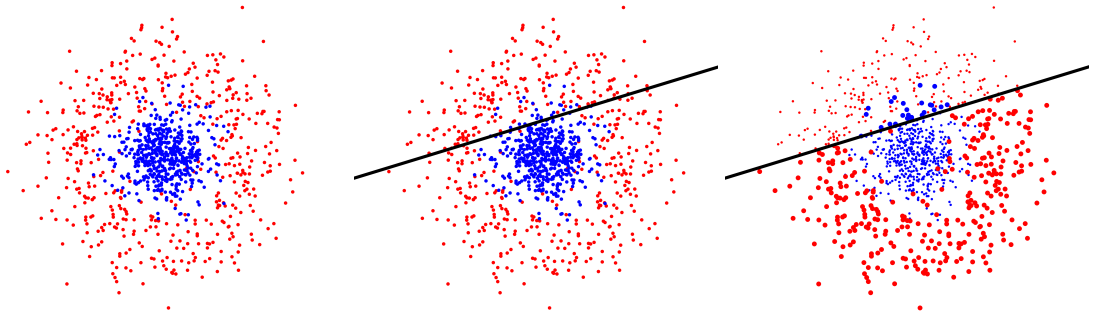


FIGURE 3.8 – Exemple d'apprentissage par un algorithme AdaBoost - ajout d'un premier classifieur faible [48]

A droite, l'ensemble de départ ; au milieu, ajout d'un premier classifieur faible ; à gauche, le poids des exemples mal classés augmentent.

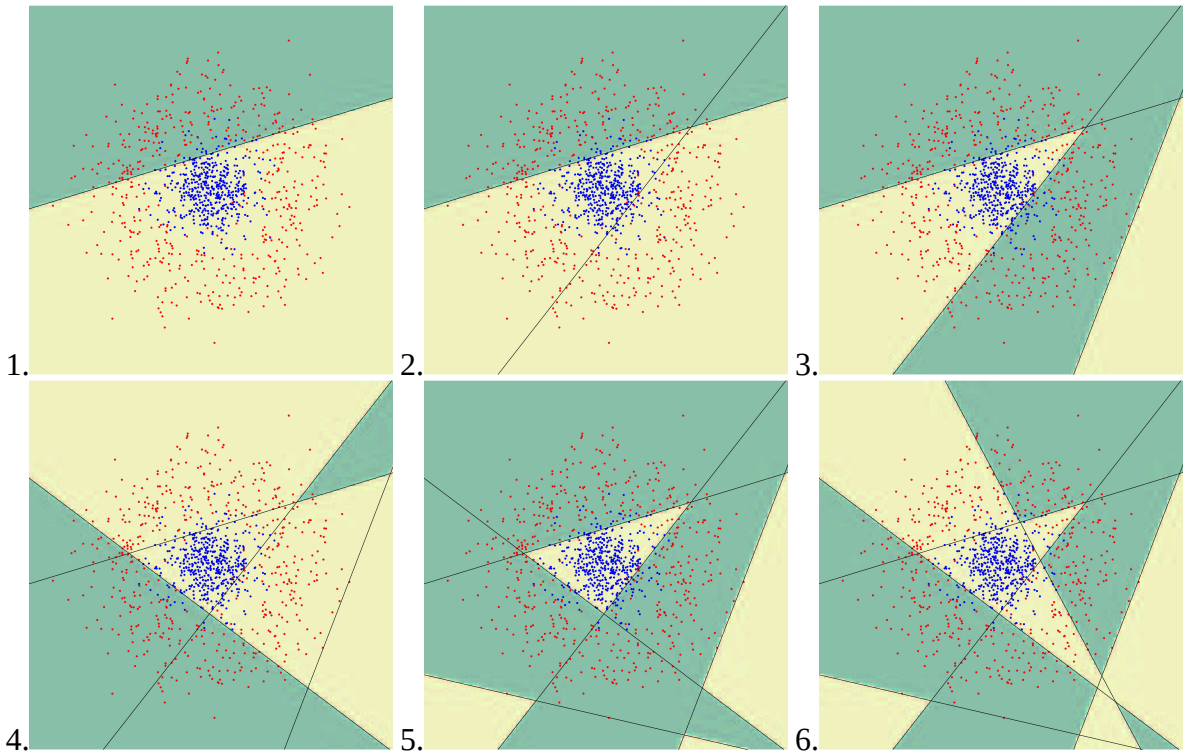


FIGURE 3.9 – Exemple d'apprentissage par un algorithme AdaBoost - combinaison de plusieurs classifieurs faibles [48]
A chaque itération, un nouveau classifieur faible est ajouté, et son avis est ajouté à la règle de décision de façon à séparer au mieux les deux classes d'objets.

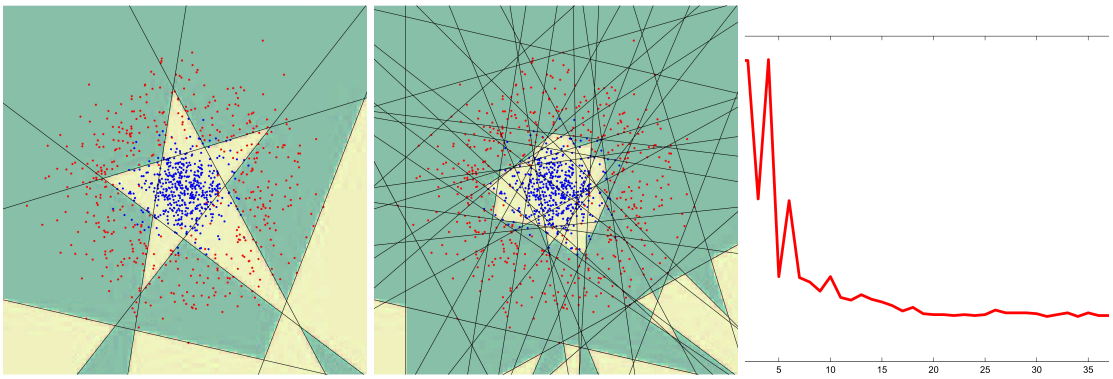


FIGURE 3.10 – Exemple d'apprentissage par un algorithme AdaBoost - fin [48]
Au fur et à mesure que les classifieurs faibles sont ajoutés, l'erreur de classification sur l'ensemble d'apprentissage converge vers une valeur (ici environ 5%*). Au-delà de cette valeur, les nouveaux classifieurs doivent se concentrer à classer les derniers exemples « difficiles ».*

Algorithme 1 AdaBoost

ENTRÉES:

Soit un ensemble d'apprentissage tel que
$\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1,...,N}$,
avec $\mathbf{x(i)} \in \mathbb{R}^M$,
et $y_i \in \{+1, -1\}$, label de classe.
pour $i = \{1, ..., N\}$ **faire**
$p_1(\mathbf{x_i}) \leftarrow \frac{1}{N}$
$i = i + 1$
fin pour $i = N$
pour $t = \{1, ..., T\}$ **faire**
Tirer un échantillon d'apprentissage S_t dans S selon les probabilités p_t.
Entraîner un classifieur C_t sur ce sous-échantillon.
Soit ϵ_t l'erreur apparente de C_t sur S.
Calculer $\alpha_t = \frac{1}{2} ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
pour $i = \{1, ..., N\}$ **faire**
- si $h_t(\mathbf{x}_i) = y_i$ (bien classé par h_t) :

$$p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{-\alpha_t}$$

- si $h_t(\mathbf{x}_i) \neq y_i$ (mal classé par h_t) :

$$p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{+\alpha_t}$$

- $i = i + 1$
fin pour $i = N$
$t = t + 1$
fin pour $t = T$
NB : Z_t est une valeur de normalisation telle que $\sum_{i=1}^{m} p_t(\mathbf{x_i}) = 1$.

SORTIES: Un classifieur fort C constitué d'un ensemble de classifieurs faibles $C_1, ..., C_T$ tel que $\epsilon_T \approx 0$ fournissant l'hypothèse finale : $H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ pour un nouvel exemple $\mathbf{x}$. La règle de décision pour attribuer la classe y de cet exemple est : $y = sign\,(H(\mathbf{x}) \geq 0)$, avec la convention $sign(0) = 1$.

3.3.3 Discussion

Nous verrons dans le chapitre 4 l'application d'un algorithme d'AdaBoost sur un exemple de classification concret. Mais il est d'ores et déjà possible de discuter des avantages et des inconvénients d'une telle méthode de classification.
Tout d'abord, notons la facilité de mise en œuvre d'un tel algorithme. En effet, peu de paramètres sont à régler : la taille de l'ensemble d'apprentissage et de chaque sous-échantillon, et le critère d'arrêt de l'algorithme. Classiquement l'algorithme est interrompu lorsque l'erreur sur l'ensemble d'apprentissage passe en dessous d'un seuil jugé satisfaisant par l'utilisateur ou lorsque le nombre de classifieurs faibles sélectionnés atteint un nombre fixé au préalable. Ensuite, l'algorithme AdaBoost ne nécessite pas une connaissance *a priori* du classifieur faible et peut donc s'utiliser pour tous types de classifieurs faibles. De plus, l'AdaBoost permet de détecter au cours de l'apprentissage les exemples erronés (« *outliers* ») puisqu'ils obtiendront rapidement un poids très grand.
Le problème majeur de l'AdaBoost apparaît ainsi. Cet algorithme est sensible aux données aberrantes et donc au bruit sur l'ensemble d'apprentissage. Des solutions ont été apportées pour résoudre ce problème comme le « *Gentle AdaBoost* » [40] ou le « *BrownBoost* » [26].
Enfin notons que des méthodes ont également été proposées pour adapter cet algorithme aux classifications multi-classes [102].

3.4 Méthodes à noyau

En 1998, la parution du livre de Vapnik [93] intitulé « *Statistical Learning Theory* » a permis de faire émerger un nouveau type de classifieurs couramment appelés « machines à noyau ». Nous nous plaçons là encore dans le cas de la classification entre deux classes d'objets. Nous avons un modèle unique de classifieur pour lequel nous cherchons à déterminer les poids $\{w_i\}_{i=1,\dots,N}$:

$$h(\mathbf{x}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) \tag{3.4}$$

avec $\mathbf{x}_i$ le vecteur de caractéristiques d'un objet et $h(\mathbf{x})$ l'hypothèse fournie par le classifieur quant à la classe de l'objet. K est une fonction noyau.
L'idée principale de ces méthodes est de transposer l'espace de représentation des données dans une dimension supérieure lorsque dans l'espace d'origine les données ne sont pas linéairement séparables ; le but est de trouver dans ce nouvel espace une séparatrice linéaire. Cette transposition est assurée par la fonction noyau K qui ne requiert pas le connaissance explicite de la transformation à appliquer pour changer d'espace.
Pour déterminer les poids w_i, plusieurs méthodes peuvent être envisagées, nous en présentons trois :

– les SVM pour *Support Vector Machine* [2] ont une approche géométrique du problème (voir section 3.4.2) ;
– les RVM pour *Relevance Vector Machines* abordent celui-ci sous une formulation probabiliste (voir section 3.4.3) ;
– enfin, la dernière approche que nous proposons estime les paramètres w_i avec un critère au sens des moindres carrés pénalisés (voir paragraphe 3.4.4).

3.4.1 Notations

Soit le cas d'une classification entre deux classes d'objets. Un ensemble d'apprentissage $\mathcal{S}$ est composé de N exemples :

$$\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1,\dots N}$$

avec $\mathbf{x}_i \in \mathbb{R}^M$, le vecteur de caractéristiques d'un exemple comprenant M composantes.
et $y_i \in \mathbb{R}$, un scalaire indiquant la classe de l'objet : classiquement $y_i = 1$ si l'objet appartient à la classe d'objet recherché (exemple positif), et $y_i = -1$ sinon (exemple négatif).

3.4.2 *Support Vector Machine* (SVM)

3.4.2.1 Introduction

Les SVM [11, 16, 93] sont la méthode la plus connue parmi les machines à noyau.
Elles utilisent aussi une fonction noyau pour transposer les données d'entrée dans un espace de dimension supérieure et ainsi déterminer une séparation linéaire entre les deux classes d'objets. Le principe des SVM est ensuite de maximiser la marge, qui est la distance séparant les exemples les plus proches de la frontière entre les deux classes d'objets. Maximiser cette marge permet d'être plus robuste aux éventuelles variations des exemples de chaque classe.

3.4.2.2 Formulation

Classifieur linéaire Notons X l'espace de représentation des données $\mathbf{x}_i$ ($X \subseteq \mathbb{R}^M$). Un classifieur linéaire est généralement réalisé par une fonction $h(\mathbf{x})$ affine définie par :

$$h(\mathbf{x}) = \sum_{j=1}^{M} w_j x_j + b = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \tag{3.5}$$

où $\mathbf{w}$ est appelé vecteur poids, composé de M éléments, qui permet de pondérer les composantes du vecteur d'observation $\mathbf{x}$ et b représente le biais. $\langle \cdot \rangle$ correspond au produit scalaire.

2. traduit en français par « Machines à Vecteurs Support » ou par « Séparateur à Vaste Marge » (la raison de cette appellation est expliquée à la section 3.4.2).

Pour savoir à quelle classe appartient l'objet, la règle de décision est donnée par le signe de $h(\mathbf{x})$: $y = sign(h(\mathbf{x}))$ avec la convention $sign(0) = 1$. Si $sign(h(\mathbf{x})) \geq 0$, l'exemple caractérisé par $\mathbf{x}$ est assigné à la classe positive, sinon cet exemple est assigné à la classe négative.
La frontière de décision est un hyperplan qui divise l'espace en deux sous-ensembles qui correspondent aux deux différentes classes d'objets (voir figure 3.11).
Introduisons maintenant la notion de séparation linéaire des données. Les données d'un ensemble d'apprentissage sont linéairement séparables si et seulement si :

$$\exists (w, b) \in (\mathbb{R}^M, \mathbb{R}) \text{ tel que } y_i \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 0 \; \forall i = 1, ..., N \text{ et } y_i \in -1, 1 \qquad (3.6)$$

Cette définition indique qu'il doit exister un hyperplan séparateur laissant d'un côté toutes les données positives ($y_i \geq 0$) et de l'autre, toutes les données négatives ($y_i < 0$).

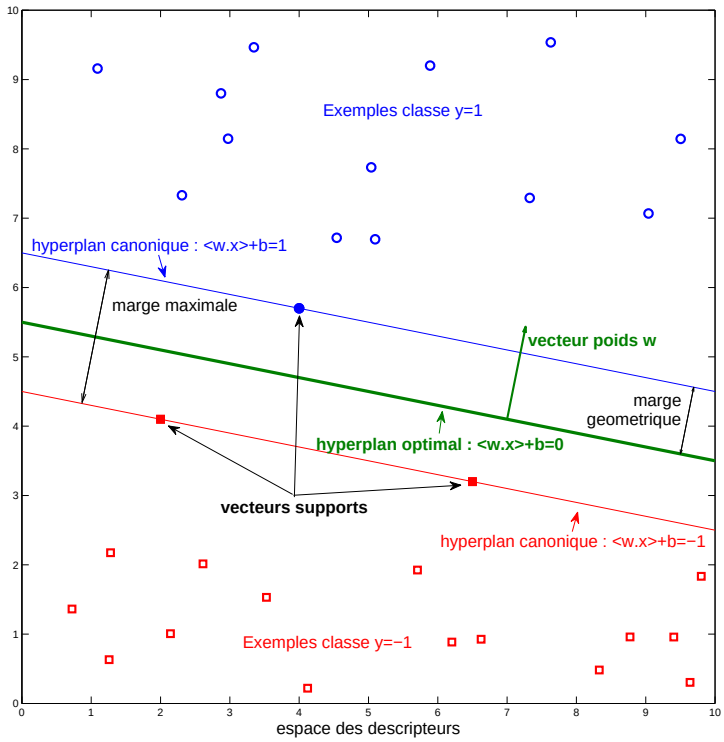


FIGURE 3.11 – Marge maximale, marge géométrique, hyperplans canoniques, hyperplan optimal et vecteurs support
Deux hyperplans canoniques passent par les exemples critiques de chaque classe, appelés « vecteurs support ». Les SVM essayent de trouver un hyperplan optimal qui passe entre ces deux hyperplans canoniques de façon à maximiser la marge géométrique.

Marge fonctionnelle et marge géométrique Nous voyons dès à présent qu'il est possible de trouver plusieurs hyperplans séparateurs. Nous allons alors chercher à choisir parmi tous les hyperplans existants le plus sûr, soit celui qui passe « au milieu » des points. En effet, si un exemple proche de la frontière de séparation doit subir une légère variation, cela n'affectera pas sa classification si sa distance à l'hyperplan est suffisamment grande.
Plus formellement, il s'agit de chercher un hyperplan dont la distance minimale aux exemples d'apprentissage les plus proches est maximale. Cette distance est appelée « marge ». Une des clés des SVM est de chercher à maximiser cette marge [3] ; les exemples les plus proches ont donc un rôle primordial puisqu'ils vont permettent de choisir le meilleur hyperplan : ils sont appelés « vecteurs support » [4]. Il existe deux marges : la marge fonctionnelle et la marge géométrique.
La marge fonctionnelle d'un exemple $\mathbf{x}_i$ à l'hyperplan se définit comme étant la quantité :

$$M_f(\mathbf{x}_i, y_i) = y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \tag{3.7}$$

La marge géométrique, quant à elle, se définit comme étant la distance euclidienne (prise perpendiculairement) entre l'exemple $\mathbf{x}_i$ et l'hyperplan caractérisé par $\mathbf{w}$ et b (voir figure 3.11), soit :

$$M_{g(\mathbf{w},b))}(\mathbf{x}_i, y_i) = y_i \left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i \right\rangle + \frac{b}{\|\mathbf{w}\|} \right) \tag{3.8}$$

La marge géométrique de tout l'ensemble d'apprentissage peut ainsi être définie par rapport à l'hyperplan caractérisé par $\mathbf{w}$ et b :

$$M_{g(\mathbf{w},b)} = \underset{i=1,\dots N}{min} M_{g(\mathbf{w},b)}(\mathbf{x}_i, y_i) \tag{3.9}$$

La marge géométrique sur tout l'ensemble d'apprentissage est donc la distance qui sépare l'exemple le plus proche de l'hyperplan à ce même hyperplan.

Cas linéairement séparable Les SVM cherchent donc un séparateur linéaire qui permet de maximiser la marge géométrique de tout l'ensemble d'apprentissage (voir équation 3.9). Le classifieur linéaire permet de classer les exemples de part leurs distances à l'hyperplan. Cette information est fournie par le signe de $h(\mathbf{x})$: $y = sign(h(\mathbf{x}))$ (avec la convention $sign(0) = 1$). Or, nous avons également vu que la distance d'un exemple à l'hyperplan est définie par la marge fonctionnelle (voir équation 3.7). Nous allons donc considérer la sortie du classifieur $h(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$ qui nous donnera ainsi la distance de chaque exemple à l'hyperplan.
Dans le cas de la classification entre deux classes, les exemples sont classés soit en exemples positifs, soit en exemples négatifs. L'hyperplan séparateur est la médiatrice du plus petit segment reliant les enveloppes convexes des deux classes. Si les données sont linéairement séparables, il est possible de définir deux hyperplans canoniques dont la marge fonctionnelle vaut 1 et sur

3. C'est pourquoi l'autre appellation des SVM est « Séparateur à Vaste Marge ».

4. La dénomination « *Support Vector Machine* » (traduit littéralement par « Machine à Vecteurs Support ») découle de cette notion.

lesquels reposent les exemples les plus proches de l'hyperplan séparateur. Leurs équations sont données par :

$$\begin{array}{rcll} \langle \mathbf{w} \cdot \mathbf{x} \rangle + b & = & 1 & \text{pour les exemples positifs,} \\ \langle \mathbf{w} \cdot \mathbf{x} \rangle + b & = & -1 & \text{pour les exemples négatifs.} \end{array} \tag{3.10}$$

En supposant que les données sont linéairement séparables, nous allons essayer de trouver le meilleur hyperplan séparateur - défini par son vecteur poids $\mathbf{w}$ et par son biais b - c'est-à-dire celui qui passe au lieu des deux hyperplans canoniques. La marge géométrique du classifieur est la distance d'un hyperplan canonique à l'autre et vaut :

$$\begin{aligned} M_{g(\mathbf{w},b)} &= \frac{1}{2}\left(y^+ \left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^+ \right\rangle + \frac{b}{\|\mathbf{w}\|} \right) + y^- \left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^- \right\rangle + \frac{b}{\|\mathbf{w}\|} \right) \right) \\ &= \frac{1}{2\|\mathbf{w}\|} \left((\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b) - (\langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b) \right) \\ &= \frac{1}{\|\mathbf{w}\|} \end{aligned} \tag{3.11}$$

$\mathbf{x}^+$ et y^+ correspondent à un exemple positif tandis que $\mathbf{x}^-$ et y^- correspondent à un exemple négatif.

Nous cherchons donc l'hyperplan optimal de façon à maximiser la marge, ce qui revient à minimiser $\|\mathbf{w}\|$. Le problème d'optimisation suivant [5] peut alors s'écrire :

$$\begin{cases} \text{Minimiser} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{sous les contraintes} & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geqslant 0 \quad \forall i = 1, ..., N \end{cases} \tag{3.12}$$

Cette formulation est appelée le problème *primal* qui nécessite le réglage de $m+1$ paramètres. L'ensemble d'apprentissage $\mathcal{S}$ est disponible pour le résoudre et les inconnues sont $\mathbf{w}$ et b. C'est un problème quadratique qui peut être résolu pour des petites valeurs de m mais inenvisageable dès lors que m devient très grand. Le but est alors d'écrire le problème proposé sous une autre formulation qui permettra de le résoudre plus aisément. Il s'agit de la formulation *duale*. Un problème *dual* fournit la même solution que le problème *primal* via une autre formulation [6]. Les variables du problème dual sont appelées variables duales. Celles-ci n'interviennent d'ailleurs pas dans l'expression du problème primal.

Dans le cas des SVM , c'est une fonction, appelée lagrangien, qui est utilisée. Le lagrangien est la somme de la fonction à minimiser et d'une combinaison linéaire des contraintes, soit :

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) \tag{3.13}$$

5. Minimiser $\|\mathbf{w}\|$ revient à minimiser $\frac{1}{2}\|\mathbf{w}\|^2$; en effet le carré permet de s'affranchir de la racine incluse dans la norme et le facteur $\frac{1}{2}$ permet de simplifier les notations pour la suite.

6. La formulation du problème *primal* en problème *dual* ne peut s'effectuer que si la fonction à minimiser et les contraintes sont strictement convexes.

où $\alpha_i > 0$ sont les multiplicateurs de Lagrange ou variables duales.
Pour résoudre le problème, il faut minimiser la formulation duale par rapport aux variables primales ($\mathbf{w}$ et b) et la maximiser par rapport aux variables duales (α_i). D'après le théorème de Kuhn-Tucker, cette solution est donnée au point-selle du lagrangien, où la dérivée du lagrangien par rapport aux variables primales doit s'annuler, soit :

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) &= 0 \\ \text{et} \quad \frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) &= 0 \end{aligned} \tag{3.14}$$

$\mathbf{w}$ est obtenu par :

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{3.15}$$

et

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.16}$$

En substituant (3.15) et (3.16) dans (3.13), il résulte :

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{N} \alpha_i (y_i(\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + b) - 1) \\ &= \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \sum_{i=1}^{N} \alpha_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \end{aligned} \tag{3.17}$$

Le problème d'optimisation peut alors s'écrire ainsi :

$$\begin{cases} \arg\min\limits_{\boldsymbol{\alpha}} & \left\{ \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \right\} \\ \text{tel que} & \alpha_i \geq 0 \text{ pour } i = 1, ..., N \\ \text{et} & \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{3.18}$$

Une solution $\boldsymbol{\alpha}^*$ est alors obtenue ; elle permet de déterminer le vecteur poids solution $\mathbf{w}^*$ d'après l'équation 3.15. L'équation de l'hyperplan séparateur est alors donnée par :

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b^* \tag{3.19}$$

Les conditions complémentaires de Karush-Kuhn-Tucker (appelé *conditions KKT complémentaires*) permettent de préciser plusieurs choses.
Tout d'abord, il est montré que seuls les points situés sur un hyperplan canonique (voir 3.10) jouent un rôle. Ce sont en effet les seuls exemples pour lesquels les multiplicateurs de Lagrange sont non-nuls. Ils sont appelés les « *vecteurs support* ». Ainsi $\mathbf{w}^*$ peur être déterminé avec seulement un sous-échantillon de S constitué des vecteurs support. Ainsi l'équation de l'hyperplan solution (3.19) n'est exprimée qu'en fonction des vecteurs support, ce qui peut se vérifier de façon intuitive en se référant à la figure 3.4.
Enfin la formulation précédente (3.18) ne fait pas intervenir le biais b. Néanmoins celui-ci peut être déterminé avec l'équation suivante :

$$\alpha_i \left[y_i \left(\langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle + b \right) - 1 \right] = 0 \text{ avec } i \in 1, ..., N \tag{3.20}$$

Grâce à cette formulation, la solution ne dépend plus de la dimension des données m mais du nombre d'échantillons N contenus dans l'ensemble d'apprentissage $\mathcal{S}$. Pour être plus précis, elle ne dépend même que du nombre de vecteurs support et le plus souvent les méthodes d'optimisation standard peuvent suffire pour trouver la solution. En outre, l'hyperplan solution ne requiert au final que le calcul des produits scalaires $\langle \mathbf{x}, \mathbf{x}_i \rangle$.

Cas non-linéaire Dans le cas de données non linéairement séparables, des variables ressort (en anglais « stack variables ») permettent de rechercher un hyperplan en essayant de faire le moins d'erreurs possible. Pour se faire, des contraintes sont relâchées grâce à des variables ressort $\xi_i \geqslant 0$ telles que :

$$y_i \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i \tag{3.21}$$

Le problème primal consiste alors à minimiser :

$$\begin{aligned} \arg\min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \textstyle\sum_{i=1}^{N} \xi_i \\ \text{tel que} \quad & y_i \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i \end{aligned} \tag{3.22}$$

C étant une constante strictement positive. Le problème dual s'écrit alors :

$$\begin{cases} \arg\max\limits_{\boldsymbol{\alpha}} & \left\{ \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i \cdot \mathbf{x}_j \right\rangle \right\} \\ \text{tel que} & \alpha_i \geq C \text{ pour } i = 1, ..., N \\ \text{et} & \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{3.23}$$

Le coefficient C est souvent appelé constante de « *trade-off* » (compromis) puisqu'il règle le nombre d'erreurs acceptables par rapport au fait de maximiser la marge. Il est choisi par l'utilisateur. En pratique, si les données d'apprentissage sont très bruitées, il faut choisir une petite valeur pour C afin de donner plus d'importance à la marge. Par contre, pour obtenir des résultats correspondants plus à l'ensemble d'apprentissage, il faut utiliser une constante C de grande valeur.
Suivant la valeur des multiplicateurs de Lagrange α_i, les exemples d'apprentissage $\mathbf{x}_i$ correspondants peuvent se classer en trois catégories :

- $\alpha_i = 0$: l'exemple d'apprentissage correspondant est bien classé mais n'est pas sur un hyperplan canonique ; ce n'est donc pas un vecteur support et il ne sera pas conservé pour construire le modèle final ;
- $0 < \alpha_i < C$: l'exemple correspondant est bien classé et se trouve sur un hyperplan canonique ; il s'agit donc d'un vecteur support qui servira à construire le classifieur final ;
- $\alpha_i = C$: l'exemple est mal classé mais étant donné que α_i est non nul, il sera tout de même considéré comme vecteur support ; il s'agit d'un outlier.

Les vecteurs support conservés pour construire le classifieur final seront donc les exemples de l'ensemble d'apprentissage pour lesquels les multiplicateurs de Lagrange sont non-nuls.

Introduction d'une fonction noyau Les SVM apportent une solution adéquate pour un système linéaire mais la solution n'est pas optimale pour des systèmes non-linéaires.
Une solution consiste à choisir le bon espace de description des données afin de se placer dans une situation linéairement séparable. Ceci consiste ni plus ni moins qu'à avoir une idée bien précise de la solution. Ceci dit, plus l'espace de description est élevé, plus nous avons de chances de trouver un hyperplan séparateur linéaire. Il faut donc chercher à transformer l'espace d'entrée en un espace de dimension supérieure et ensuite utiliser les SVM pour trouver un hyperplan séparateur dans ce nouvel espace. Soit Φ une transformation non-linéaire qui transforme l'espace d'entrée χ en un espace de redescription $\Phi(\chi)$; dans notre cas, pour un exemple de l'ensemble d'apprentissage $\mathbf{x}$, nous obtenons :

$$\mathbf{x} = x_1, ..., x_M \mapsto \Phi(\mathbf{x}) = \Phi(x_1), ..., \Phi(x_M) \tag{3.24}$$

Le problème d'optimisation devient alors :

$$\begin{cases} \arg\max\limits_{\boldsymbol{\alpha}} & \left\{ \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \right\rangle \right\} \\ \text{tel que} & \alpha_i \geq 0 \text{ pour } i = 1, ..., N \\ \text{et} & \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{3.25}$$

L'équation de l'hyperplan séparateur devient alors :

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + b^* \tag{3.26}$$

où α_i^* et b^* sont solutions de 3.25.
Cette écriture est intéressante si le calcul du produit scalaire $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$ est omis. En effet, il devient rapidement impossible à calculer si l'espace de description des exemples est élevé. Heureusement il existe des fonctions bilinéaires symétriques positives, notées $K(x, y)$, appelées fonctions noyau, qui facilitent le calcul de ce produit scalaire. En effet, ces fonctions, qui vérifient la condition de Mercer [7], correspondent à un produit scalaire $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$ dans un espace de grande dimension.
Le problème peut ainsi se simplifier :

$$\begin{cases} \arg\max\limits_{\boldsymbol{\alpha}} & \left\{ \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i . \mathbf{x}_j) \right\} \\ \text{tel que} & \alpha_i \geq 0 \text{ pour } i = 1, ..., N \\ \text{et} & \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{3.27}$$

avec pour solution l'hyperplan d'équation :

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^* \tag{3.28}$$

7. Le théorème de Mercer donne les conditions pour qu'une fonction K soit équivalente à un produit scalaire (voir annexe A)

où α_i^* et b^* sont solution de 3.27.
Introduire une fonction noyau K permet de transposer la dimension d'entrée m en une dimension de redescription supérieure à m où les données sont linéairement séparables. Trouver un séparateur linéaire dans ce nouvel espace devient possible alors que c'était irréalisable dans l'espace d'entrée. Le classifieur final peut s'écrire sous une forme plus compacte :

$$h(\mathbf{x}) = \sum_{i=1}^{N} w_i^* K(\mathbf{x}, \mathbf{x}_i) + b^* \tag{3.29}$$

La figure suivante (voir figure 3.12) donne un exemple de classification par les SVM sur un jeu de données d'apprentissage. La frontière de décision s'appuie sur des vecteurs support (en bleu) ; la séparation est choisie de façon à maximiser la marge.

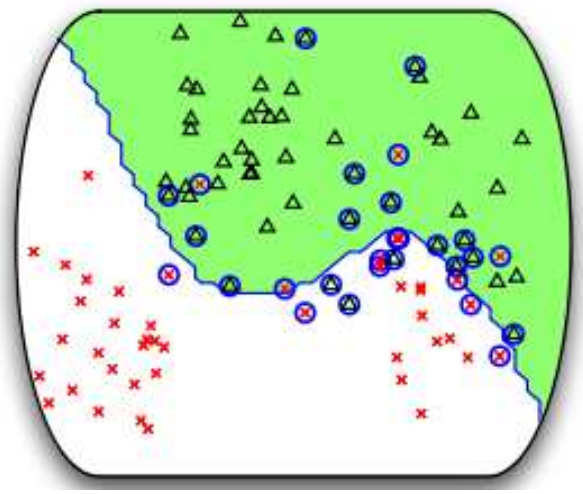


FIGURE 3.12 – Exemple de classification par les SVM
Les croix rouges et les triangles noirs représentent les points d'apprentissage de deux classes différentes. Les points cerclés en bleu représentent les vecteurs support sélectionnés par les SVM. La surface verte représente la région de l'espace des paramètres à laquelle appartiennent les points de la classe des triangles noirs et la surface blanche représente la région à laquelle appartiennent les points de la classe des croix rouges.

Choix d'une fonction noyau Il existe plusieurs sortes de noyaux. Une description de ceux-ci est fournie dans le chapitre « *Kernel-Induced Features Spaces* » de [16]. Trouver le noyau adapté à une application précise est un choix souvent difficile à faire. Le choix se fait souvent de façon empirique, à force d'essais. C'est certainement l'aspect le moins attractif des SVM mais c'est le seul élément à paramétrer, ce qui est finalement satisfaisant au regard d'autres méthodes d'apprentissage.
En ce qui concerne l'application visée dans mes travaux, nous avons utilisé des fonctions noyau

à base radiale (kernel RBF) car il n'y a qu'un seul paramètre à ajuster et elles peuvent facilement s'adapter à une grande base de données.
La fonction noyau RBF est donnée par :

$$K(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{3.30}$$

avec σ^2 l'écart-type. Ce type de noyau est très populaire car il transpose les données dans un espace de dimension infinie. Le seul paramètre à régler est σ qui correspond à la largeur de la gaussienne. Si σ a une valeur trop grande, tous les exemples vont se ressembler, tandis que si sa valeur est trop petite, aucun exemple ne ressemblera à un autre.

3.4.2.3 Discussion

Les SVM sont une technique bien fondée mathématiquement et un certain nombre de propriétés est maîtrisé. Le plus grand intérêt de cette méthode est d'apporter une solution automatique au difficile compromis à faire entre la fidélité aux données d'apprentissage et la création d'un bon modèle discriminant. L'approfondissement de la formulation théorique donne et donnera certainement encore lieu à des améliorations des SVM et il faut s'attendre à ce que de nouvelles méthodes voient le jour.
D'ailleurs, la plus grande et intéressante conséquence des SVM est d'avoir permis de mettre en avant ces fonctions noyau. En effet, celles-ci peuvent profiter à d'autres méthodes utilisant des nouvelles mesures de distance que celles qui étaient initialement envisagées pour la méthode SVM .
La méthode qui suit, les *Relevance Vector Machine*, a recours à ces fonctions noyau dans une approche probabiliste.

3.4.3 *Relevance Vector Machine* (RVM)

3.4.3.1 Introduction

La méthode des RVM pour « *Relevance Vector Machine* » a été développée par Tipping [89, 90]. C'est une méthode qui permet aussi de traiter des problèmes de régression. Elle utilise le modèle classique linéaire des machines à noyau des SVM (voir équation 3.5) mais fait appel à une formulation bayésienne pour déterminer les paramètres et sélectionner les exemples pertinents qui permettront de réaliser le modèle discriminant final.

3.4.3.2 Formulation

Le même ensemble d'apprentissage $\mathcal{S}$ que précédemment est utilisé ; il est composé de N exemples tel que $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1,...N}$ avec $\mathbf{x}_i \in \mathbb{R}^M$, le vecteur de caractéristiques d'un exemple comprenant M composantes et $y_i \in \mathbb{R}$, un scalaire indiquant la classe de l'objet.

Pour une meilleure compréhension de la méthode, plaçons-nous tout d'abord dans le cadre de la régression, où y_i est une fonction continue ; le passage au cas de la classification, où y_i prend deux valeurs discrètes, sera explicité à la fin de cette section.
Le principe des RVM est de modéliser toutes les quantités du système par des densités de probabilité. Cela permet de représenter le bruit sur les données d'apprentissage et d'éviter ainsi des phénomènes de surapprentissage. Pour chaque donnée d'apprentissage, considérons t_i la réalisation bruitée de la vérité y_i :

$$t_i = y_i + \epsilon_i \tag{3.31}$$

avec $(\epsilon_i)_{i\in\mathbb{I}}$ des bruits indépendants Gaussien de moyenne nulle et de variance σ^2.
Les SVM fournissent une prédiction donnée dans l'équation 3.29. Le vecteur poids est noté $\mathbf{w} = \{w_1, ..., w_N\}$. Si le biais b est intégré dans $\mathbf{w}$, le problème à résoudre est le suivant :

$$y = \sum_{i=1}^{N} w_i^* K(\mathbf{x}, \mathbf{x}_i) \tag{3.32}$$

avec $K(,)$ la fonction noyau.
Un modèle bayésien classique est introduit tel que la fonction de vraisemblance sur l'ensemble d'apprentissage s'écrive :

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-\frac{N}{2}} exp\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \Phi\mathbf{w}\|^2\} \tag{3.33}$$

avec $\mathbf{w} = w_1, ..., w_N$.
Φ est la matrice telle que $\Phi_{nm} = K(\mathbf{x}_n, \mathbf{x}_{m-1})$ et $\Phi_{n1} = 1$. L'estimation du maximum de vraisemblance mène généralement à des phénomènes de surapprentissage. Mais ce problème peut être résolu en utilisant des fonctions plus « régulières » définies par des poids gaussiens de l'ARD [8] :

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \tag{3.34}$$

avec $\boldsymbol{\alpha}$ un vecteur de $N+1$ hyperparamètres. L'introduction d'un hyperparamètre pour chaque poids est la clé des RVM ; c'est ce qui permet aussi d'obtenir un modèle épars. En appliquant la règle de Bayes, la probabilité *a posteriori* sur les poids est obtenue :

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-\frac{N+1}{2}} |\Sigma|^{-\frac{1}{2}} exp\{-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{w} - \boldsymbol{\mu})\} \tag{3.35}$$

avec :

$$\Sigma = (\Phi^T \mathrm{B} \Phi + \mathrm{A})^{-1} \tag{3.36}$$

8. L'ARD pour « *Automatic Relevance Determination* » est une approche basée sur l'interférence bayésienne dans laquelle ce sont des hyperparamètres qui permettent de contrôler l'amplitude des intervalles dans lesquels évoluent les données d'entrée [64].

$$\boldsymbol{\mu} = \Sigma \Phi^T \mathtt{B} \mathbf{t} \tag{3.37}$$

$$\mathtt{A} = diag(\alpha_0, \alpha_1, ..., \alpha_N) \tag{3.38}$$

$$\mathtt{B} = \sigma^{-2} \mathtt{I}_N \tag{3.39}$$

σ^2 est également traité comme hyperparamètre et peut être estimé à partir des données d'apprentissage.
En substituant les poids, nous obtenons la vraisemblance marginale :

$$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-\frac{N}{2}} |\mathtt{B}^{-1} + \Phi \mathtt{A}^{-1} \Phi^T|^{\frac{1}{2}} exp\{-\frac{1}{2} \mathbf{t}^T (\mathtt{B}^{-1} + \Phi \mathtt{A}^{-1} \Phi^T)^{-1} \mathbf{t}\} \tag{3.40}$$

Dans l'idéal, il faudrait également définir des hyperpoids sur $\boldsymbol{\alpha}$ et σ^2 et substituer aussi ces hyperparamètres ; toutefois cette méthode est difficilement applicable sous cette formulation. C'est pourquoi Tipping propose d'utiliser une procédure plus pragmatique citée dans [63]. Celle-ci consiste à optimiser directement la vraisemblance marginale (voir équation 3.40), par rapport à $\boldsymbol{\alpha}$ et σ^2, ce qui revient à trouver le maximum de $p(\boldsymbol{\alpha}, \sigma^2|t)$ en assurant des poids uniformes. Ensuite, en utilisant ces valeurs maximales, des prédictions sont faites à partir de 3.35.
Les valeurs de $\boldsymbol{\alpha}$ et σ^2 qui maximisent 3.40 ne peuvent pas être obtenues directement. Il faut passer par une formule itérative pour estimer la valeur de σ :

$$\alpha_i^{\text{nouveau}} = \frac{1 - \alpha_i \Sigma_{ii}}{\mu_i^2} \tag{3.41}$$

La valeur de σ^2 est donnée par :

$$(\sigma^2)^{\text{new}} = \frac{\|\mathbf{t} - \Phi \boldsymbol{\mu}\|^2}{N - \sum_i 1 - \alpha_i \Sigma_{ii}} \tag{3.42}$$

En pratique, pendant le processus d'optimisation, une grande partie des α_i tend vers l'infini et, en se référant à 3.35, la probabilité $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)$ tend vers 0 ; et c'est là toute l'astuce des RVM car les poids correspondants sont nuls ou quasi-nuls, ce qui implique que les fonctions noyau correspondantes peuvent être enlevées du modèle. Les vecteurs restants sont donc les vecteurs pertinents - « *relevant* » en anglais.
Ce principe est appliqué pour faire de la régression dans le cas où y a un continuum de valeurs possibles. Pour la classification entre deux classes, où $y \in \{-1, 1\}$ ($y_i = 1$ pour un exemple positif, et $y_i = 0$ sinon), l'objectif est de prédire la probabilité *a posteriori* de la classe d'un nouvel exemple sachant son observation $\mathbf{x}$. Le modèle linéaire précédant est alors généralisé en appliquant la fonction logique sigmoïde suivante à $y(\mathbf{x})$:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{3.43}$$

La vraisemblance peut s'écrire :

$$p(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{N} \sigma\{y(\mathbf{x}_i)\}^{t_i} [1 - \sigma\{y(\mathbf{x}_i)\}]^{1-t_i} \tag{3.44}$$

Il est toutefois impossible de substituer les poids pour obtenir de façon analytique la vraisemblance marginale, et une procédure itérative proposée par MacKay [63] est utilisée (voir algorithme 2).

Algorithme 2 Classification par les RVM - mise à jour des poids

répéter

- pour une valeur fixée de $\boldsymbol{\alpha}$, il faut rechercher les meilleurs poids $\mathbf{w}_{MP}$ possibles (localisation du maximum *a posteriori*). C'est équivalent à une optimisation classique d'un modèle logistique régularisé. Un algorithme des moindres carrés pondérés récursifs est utilisé pour trouver le maximum.
- le Hessien est calculé par rapport à $\mathbf{w}_{MP}$. :

$$\nabla\nabla \; log \; p(\mathbf{t}, \mathbf{w}|\boldsymbol{\alpha})|_{\mathbf{w}_{MP}} = -(\Sigma^T \mathtt{B}\Phi + \mathtt{A})$$

avec $\mathtt{B}$ et une matrice diagonale telle que $\mathtt{B} = diag\{B_1, B_2, ..., B_N\}$ et

$$\mathtt{B}_n = \sigma y(\mathbf{x}_n)[1 - \sigma y(\mathbf{x}_n)]$$

ce qui nous permet de déterminer la matrice de covariance Σ ; les hyperparamètres $\boldsymbol{\alpha}$ peuvent alors être mis à jour.

jusqu'à ce qu'un critère approprié de convergence soit satisfait.

La figure 3.13 présente un exemple de classification par les RVM sur les mêmes données d'apprentissage que pour l'exemple fourni figure 3.13. La frontière de décision s'appuie sur des vecteurs « *relevant* » (en bleu).

3.4.3.3 Discussion

Les RVM se basent sur le même modèle que les SVM . Ces deux méthodes construisent un modèle linéaire épars - ou parcimonieux - dans le sens où seul un certain nombre de vecteurs d'apprentissage est conservé : les vecteurs support des SVM et les vecteurs « *relevant* » des RVM.

Même si l'apprentissage est hors-ligne, il faut toutefois une puissance de calcul plus conséquente pour les RVM qui requièrent d'inverser au moins à la première itération une matrice de taille $N \times N$, N étant le nombre de données comprises dans l'ensemble d'apprentissage.

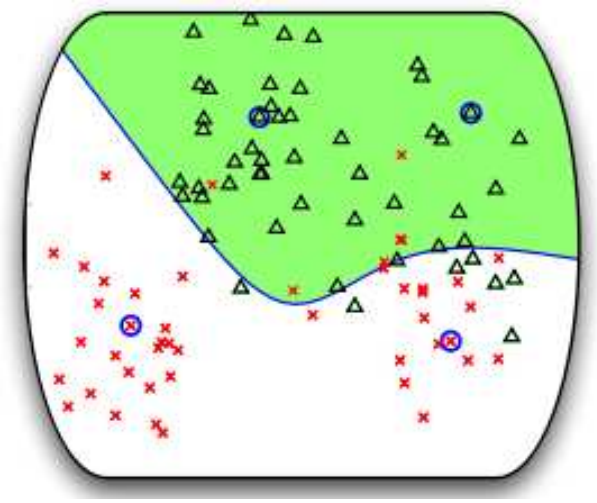


FIGURE 3.13 – Exemple de classification par les RVM
Les croix rouges et les triangles noirs représentent les points d'apprentissage de deux classes différentes. Les points cerclés en bleu représentent les vecteurs « relevant » des RVM. La surface verte représente la région de l'espace des paramètres à laquelle appartiennent les points de la classe des triangles noirs et la surface blanche représente la région à laquelle appartiennent les points de la classe des croix rouges.

3.4.4 Estimation au sens des moindres carrés

Nous avons vu précédemment que les SVM ont permis l'introduction des fonctions noyau, et que celles-ci ouvraient la voie à de nombreuses méthodes. Les RVM en ont fait une approche probabiliste. Il est également possible de résoudre le problème initial avec une approximation au sens des moindres carrés.

3.4.4.1 Méthode basique

Le but est de déterminer la classe y d'un objet inconnu ayant un vecteur d'observation $\mathbf{x}$; le classifieur recherché permet de formuler le problème sous la forme suivante :

$$y = sign\left(\sum_{t=1}^{T} w_t * K(\mathbf{x}, \mathbf{x}_t)\right) \tag{3.45}$$

$\mathbf{x}_{t\in\{1,\dots T\}}$ sont les vecteurs d'observation d'un échantillon d'apprentissage et w_t sont les poids associés à déterminer ; K une fonction noyau (voir paragraphe 3.4.2.2).
Les fonctions de base, notées $\phi_t(\mathbf{x})$, s'écrivent telles que :

$$\phi_t(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_t) \tag{3.46}$$

l'équation précédente (3.45) peut s'écrire sous une forme plus compacte :

$$y = sign(\langle \mathbf{w} \cdot \boldsymbol{\phi} \rangle) \tag{3.47}$$

avec $\mathbf{w}$ le vecteur poids tel que $\mathbf{w} = \{w_1, w_2, ..., w_T\}$ et $\boldsymbol{\phi}$ le vecteur des fonctions de bases telle que $\boldsymbol{\phi} = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_T(\mathbf{x}))$.
Pour estimer le modèle et donc déterminer le vecteur poids $\mathbf{w}$, l'ensemble d'apprentissage $\mathcal{S} = \{x_i, y_i\}_{i=1,...N}$ est utilisé ; il est composé de N exemples ayant chacun pour observation un vecteur $\mathbf{x}_i$ constitué de M attributs et y_i une étiquette de classe. Le système à résoudre est donc de la forme :

$$\mathbf{y} = \mathbf{w}\Phi \tag{3.48}$$

avec $\mathbf{y} = [y_0, ..., y_N]$ et :

$$\Phi = \begin{bmatrix} \phi_1(\mathbf{x}_1, \mathbf{x}_1) & \phi_2(\mathbf{x}_1, \mathbf{x}_2) & ... & \phi_t(\mathbf{x}_1, \mathbf{x}_t) & ... & \phi_T(\mathbf{x}_1, \mathbf{x}_T) \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ \phi_1(\mathbf{x}_i, \mathbf{x}_1) & \phi_2(\mathbf{x}_i, \mathbf{x}_2) & ... & \phi_t(\mathbf{x}_i, \mathbf{x}_t) & ... & \phi_T(\mathbf{x}_i, \mathbf{x}_T) \\ ... & ... & ... & ... & ... & \\ ... & ... & ... & ... & ... & ... \\ \phi_1(\mathbf{x}_N, \mathbf{x}_1) & \phi_2(\mathbf{x}_N, \mathbf{x}_2) & ... & \phi_t(\mathbf{x}_N, \mathbf{x}_t) & ... & \phi_T(\mathbf{x}_N, \mathbf{x}_T) \end{bmatrix} \tag{3.49}$$

Nous utilisons la norme euclidienne pour mesurer les erreurs dans l'espace des prédictions et l'estimation du problème est de la forme :

$$\mathbf{w} \stackrel{def}{=} \arg\min_{\mathbf{w}} \|\mathbf{w}\Phi - \mathbf{y}\|^2 \tag{3.50}$$

En utilisant le critère au sens des moindres carrés précédemment défini dans 3.50, l'estimation du vecteur $\mathbf{w}$ est donnée par [9] :

$$\mathbf{w}_{LS} = \mathbf{y}\Phi^+ \tag{3.51}$$

Généralement, les vecteurs utilisés dans les fonctions de base sont un sous-échantillon de ceux disponibles dans l'ensemble d'apprentissage $\mathcal{S}$ tels que dans le modèle final (équation 3.45) $T \leq N$, ce sont des moindres carrés pénalisés. C'est le cas pour les méthodes SVM - avec les vecteurs support - et RVM - avec les vecteurs « *relevant* ». Mais il est également possible de choisir d'utiliser tout l'ensemble d'apprentissage et $T = N$. La matrice Φ est symétrique et, si elle est définie positive, le système peut se résoudre plus facilement par une décomposition de Cholesky. Dans ce cas, il faut être vigilant au phénomène de surapprentissage (voir figure 3.14). La figure 3.15 présente un exemple de classification par une méthode de moindres carrés pénalisés (il s'agit des mêmes données d'apprentissage que pour les exemples des figures 3.12 et 3.13).

9. Φ^+ est la pseudo-inverse de Φ.

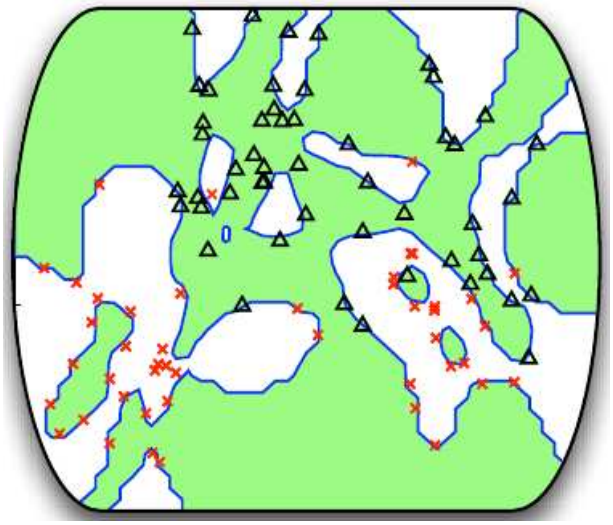


FIGURE 3.14 – Exemple de classification par des moindres carrés : problème de surapprentissage
Les croix rouges et les triangles noirs représentent les points d'apprentissage de deux classes différentes. La surface verte représente la région de l'espace des paramètres à laquelle appartiennent les points de la classe des triangles noirs et la surface blanche représente la région à laquelle appartiennent les points de la classe des croix rouges. Le classifieur a fait du surapprentissage en se focalisant sur certains exemples.

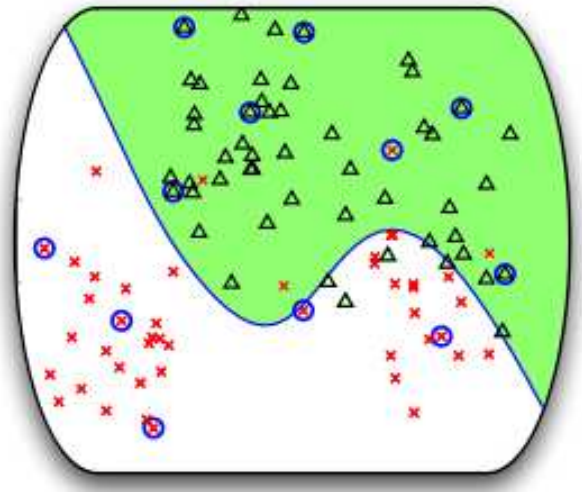


FIGURE 3.15 – Exemple de classification par des moindres carrés pénalisés
Les croix rouges et les triangles noirs représentent les points d'apprentissage de deux classes différentes. Les points cerclés en bleu représentent les vecteurs conservés par le modèle. La surface verte représente la région de l'espace des paramètres à laquelle appartiennent les points de la classe des triangles noirs et la surface blanche représente la région à laquelle appartiennent les points de la classe des croix rouges.

3.4.4.2 Sélection de vecteurs support par « *Ridge regression* »

Conserver tous les vecteurs de l'ensemble d'apprentissage permet d'obtenir rapidement un classifieur sans être obligé de passer par une phase itérative de sélection de vecteurs support pendant l'apprentissage. Toutefois garder tous ces vecteurs peut poser problème lors de l'utilisation du classifieur pendant une phase de reconnaissance en temps réel puisqu'il faut calculer la distance du nouvel exemple à tous les exemples de la base d'apprentissage.
Une approche présentée par Agarwal [2] propose une version alternative de la régression de type RVM appelée « Support Vector Regression » en utilisant une régression de Ridge. Elle permet d'ajouter plus de régularité via une contrainte sur la matrice Φ (voir équation 3.49). Cette contrainte peut se réaliser en ajoutant un terme d'amortissement ou de régularisation sur le vecteur poids, noté $R(\mathbf{w})$, qui va permettre de pénaliser les grandes valeurs du vecteur poids $\mathbf{w}$:

$$\mathbf{w} \stackrel{def}{=} \arg\min_{\mathbf{w}} \|\mathbf{w}\Phi - \mathbf{y}\|^2 + R(\mathbf{w}) \tag{3.52}$$

Lorsque le système est résolu par une simple méthode des moindres carrés (comme précédemment dans 3.4.4.1), $R(\mathbf{w}) = 0$; mais ici $R(\mathbf{w}) = \lambda\|\mathbf{w}\|^2$ avec λ le terme de régularisation. Il s'agit alors une méthode de moindres carrées amortis qui cherchent à minimiser le terme suivant :

$$\|\mathbf{w}\tilde{\Phi} - \tilde{\mathbf{y}}\|^2 \stackrel{def}{=} \|\mathbf{w}\Phi - \mathbf{y}\| + \lambda\|\Phi\|^2 \tag{3.53}$$

avec $\tilde{\Phi} = [\Phi \ \boldsymbol{\lambda}]$ ($\boldsymbol{\lambda} = [\lambda \ \lambda \ ... \ \lambda]$ un vecteur de taille $(1 \times N)$ et $\tilde{\mathbf{y}} = [\mathbf{y} \ \ 0]$.
La solution peut s'obtenir en résolvant le système linéaire suivant :

$$\mathbf{w}\tilde{\Phi} = \tilde{\mathbf{y}} \tag{3.54}$$

Les vecteurs dont les poids associés sont nuls sont supprimés du système ; et le classifieur final est obtenu en recalculant la matrice Φ avec les vecteurs support restants (voir equation 3.50). Le terme λ doit être réglé avec attention ; sa valeur doit être suffisamment grande pour contrôler mauvais conditionnement et surapprentissage mais une valeur trop forte conduirait à un amortissement trop grand : les poids seraient forcés vers 0 et le système sous-estimerait systématiquement la solution. Le réglage se fait donc souvent par validation croisée.

Chapitre 4

Analyse comparative des performances

Nous avons vu dans les deux chapitres précédents quels étaient les descripteurs (chapitre 2) et les classifieurs (chapitre 3) couramment mis en œuvre pour faire de la reconnaissance d'objets. Nous avons également proposé un descripteur avec des comparaisons de niveaux de gris (voir section 2.5.3). Ce chapitre présente maintenant comment nous avons employé et combiné tous ces éléments pour réaliser une tâche de reconnaissance. Chaque méthode développée est testée et évaluée.

4.1 Introduction

Les chapitres précédents ont abordé les deux points clés associés à une tâche de reconnaissance basée apprentissage : l'extraction de primitives (chapitre 2) et les techniques d'apprentissage et de classification (chapitre 3). Le but de ce chapitre est de montrer comment combiner un descripteur et un classifieur, et même s'il est possible d'en combiner plusieurs entre eux.
La façon la plus simple d'utiliser ces méthodes est de faire appel à un descripteur d'images et d'utiliser ensuite un classifieur. Pour cela, une base d'images conséquente est utilisée ; elle comprend des exemples de l'objet à reconnaître et des contre-exemples. La phase d'apprentissage se fait en deux étapes :

- dans un premier temps, il convient de décrire toutes ces images par le descripteur choisi ; un vecteur d'observation est ainsi obtenu pour chacun des exemples et des contre-exemples de la base d'apprentissage ;
- dans un second temps, un classifieur est entraîné sur toutes ces observations ; son rôle est de déterminer les caractéristiques communes entre exemples d'une même classe pour établir une séparatrice entre les deux classes d'objets.

Le but d'une reconnaissance est de déterminer à quelle classe appartient un nouvel objet. Les mêmes éléments que pendant l'apprentissage sont utilisés, à savoir :

- le nouvel objet est décrit au préalable par le même descripteur utilisé pendant l'apprentissage ;

– puis le classifieur entraîné pendant l'apprentissage donne une réponse quant à la classe de ce nouvel objet.

De multiples systèmes existent déjà. Certains ont été portés pour réaliser des applications temps réel - comme dans cette thèse - d'autres non. Un aperçu des systèmes faisant de la détection de piétons sera présenté dans le chapitre suivant (voir section 5.1). Les réalisations les plus communes font appel à un seul descripteur suivi d'un seul classifieur [20, 75, 96] mais de nouvelles méthodes explorent des combinaisons associant plusieurs descripteurs et/ou classifieurs [32, 31, 82].

Dans un premier temps, nous proposons d'utiliser une version classique avec un descripteur et un classifieur (voir sections 4.3, 4.4 et 4.5). En ce qui concerne les descripteurs, nous mettons en place ceux présentés dans le chapitre 2, à savoir : ondelettes de Haar, histogrammes de gradients orientés et comparaisons de niveaux de gris. Pour les classifieurs, nous aurons recours à un algorithme AdaBoost et des méthodes à noyau : SVM, RVM et moindres carrés.

Dans un second temps, nous proposons l'association de l'algorithme AdaBoost avec une méthode à noyau (voir sections 4.6 et 4.7) ; celle-ci permet de réduire la dimensionnalité des données d'entrée dans le classifieur à noyau et d'améliorer les performances de ce dernier. Cette méthode prend tout son sens puisque la base d'apprentissage, déjà conséquente, peut être décrite par des descripteurs fournissant des vecteurs d'observation de grande taille. Enfin, une association des trois descripteurs cités ci-dessus est proposée (voir section 4.8), afin de voir dans quelle mesure il est possible d'en tirer un avantage. D'un point de vue théorique, cette association devrait permettre un gain des performances. Par contre, d'un point de vue pratique, elle nécessite de calculer trois descripteurs, ce qui n'est pas réalisable en temps réel.

4.2 Protocole expérimental

4.2.1 Base d'apprentissage

Nous présentons dans la suite de ce chapitre de nombreux tests afin de comparer et valider les différentes méthodes qui sont proposées dans ce mémoire (les descripteurs, les classifieurs et les différentes associations). Pour réaliser ces tests, nous utilisons la base d'images proposée dans [68]. Ils ont proposé de subdiviser cette base en cinq parties ; chacune contient 4500 exemples positifs et 5000 négatifs. Il s'agit d'images de luminance, de taille 36×18 pixels.

Dans les images d'exemples positifs, les piétons se tiennent debout et sont entièrement visibles ; ils ont été pris dans différentes postures, et dans des conditions d'illumination et de fond variables. Pour chaque piéton, une vingtaine d'images sont disponibles : en effet, l'image initiale a été aléatoirement reflétée et décalée de quelques pixels dans les directions horizontale et verticale afin de prendre en compte toutes les variations de position du piéton dans l'image. Les images d'exemples négatifs représentent l'environnement urbain : bâtiments, arbres, voitures, panneaux de signalisation... Cette base (dont quelques exemples sont montrés dans la figure 4.1) constitue les données utilisées pour entraîner et valider la méthode proposée.

Pour réaliser un apprentissage et évaluer ses performances en reconnaissance, deux sous-ensembles d'images différentes de piétons et de non-piétons sont sélectionnés. Les auteurs préconisent d'utiliser les trois premiers ensembles pour les besoins de l'apprentissage et d'effectuer la validation sur les deux ensembles restants afin de garantir une indépendance des résultats. Nous suivons ces recommandations et menons chaque apprentissage sur un ensemble d'images provenant des trois premiers ensembles et validons les résultats sur des images issues des deux derniers. Le nombre d'images sélectionnées pour chacun des ensembles sera précisé pour chaque expérimentation réalisée.

FIGURE 4.1 – Exemples de piétons (première ligne) et de non-piétons (deuxième ligne)
Taille d'une imagette : 36×18 *pixels ; images extraites de la base [68]*

4.2.2 Critères d'évaluation

Le but d'une reconnaissance est tout d'abord de savoir retrouver les exemples positifs, désignés sous le terme de « vrais positifs » ; ce qui dans notre application correspond à des piétons que le système reconnaît bien comme des piétons.
Mais il faut également chercher à limiter le nombre de fausses alarmes, c'est-à-dire les faux positifs ; ce sont des objets que le système prend pour des piétons mais qui n'en sont pas. Ces termes sont regroupés dans la « matrice de confusion » suivante (voir tableau 4.1) :

		Réalité	
		Positifs	Négatifs
Prédictions	Positifs	Vrais positifs (VP)	Faux positifs (FP)
	Négatifs	Faux Négatifs (FN)	Vrais Négatifs (VN)

TABLE 4.1 – Matrice de confusion

Les classifieurs ne donnent jamais un résultat final parfait (par exemple, 1 pour un exemple positif et -1 pour un exemple négatif) mais une valeur réelle. Pour associer un objet à une classe, le signe de cette valeur réelle est usuellement pris (avec la convention $sign(0) = 1$), ce qui revient à fixer un seuil de classification à zéro. Selon les besoins de chaque application, ce seuil peut être ajusté. Augmenter le seuil permet d'éviter des détections ratées alors que baisser le seuil permet de limiter le taux de fausses alarmes (voir figure 4.2).

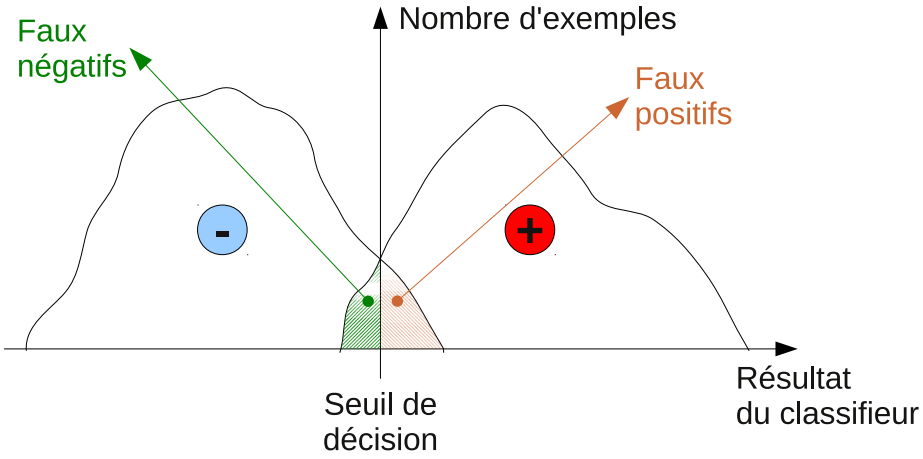


FIGURE 4.2 – Histogramme de la répartition des piétons en fonction du résultat du classifieur
Il faut ajuster le seuil de décision de façon à classer correctement le plus d'exemples possibles, tout en évitant au mieux faux positifs et faux négatifs.

Des courbes sont couramment utilisées pour évaluer les performances d'un apprentissage. Les plus courantes sont les courbes ROC [1] ou les courbes de précision/rappel.
Les courbes ROC représentent le taux de bonnes détections en fonction du taux de fausses alarmes, soit les vrais positifs en fonction des faux positifs. Pour tracer cette courbe, il faut faire varier le seuil de décision et, pour chaque valeur, calculer le taux de bonnes détections et de fausses alarmes. Sur la figure 4.3 à droite, un classifieur parfait est représenté par la courbe rouge : 100% de bonnes détections pour 0% de fausses alarmes. Une courbe correspondant à la diagonale (droite bleue sur la figure) est un classifieur qui n'est pas meilleur que le hasard. Les performances d'un classifieur peuvent ainsi être évaluées : plus il s'approchera du cas idéal, meilleur il sera.
La précision est la quantité telle que :

$$Precision = \frac{VP}{VP + FP} \tag{4.1}$$

1. *Receiver Operating Curves.*

et le rappel vaut :

$$Rappel = \frac{VP}{VP + FN} \tag{4.2}$$

Lorsque le seuil de décision est réévalué pour une application donnée, cela revient à se déplacer sur la courbe ROC et choisir un point de fonctionnement en fonction des critères désirés (par exemple, un point de fonctionnement pour 10% de fausses alarmes).
La précision représente la capacité du classifieur à associer les exemples à la bonne classe tandis que le rappel souligne la capacité du classifieur à retrouver les exemples positifs. Les courbes de précision/rappel représentent la précision en fonction du rappel pour différents seuils de classification. Le cas idéal à atteindre est d'avoir une précision et un rappel de 100%, ce qui signifie que le classifieur est capable de retrouver tous les exemples positifs sans aucune fausse détection (ce qui correspond à la courbe rouge sur la figure 4.3 à gauche).

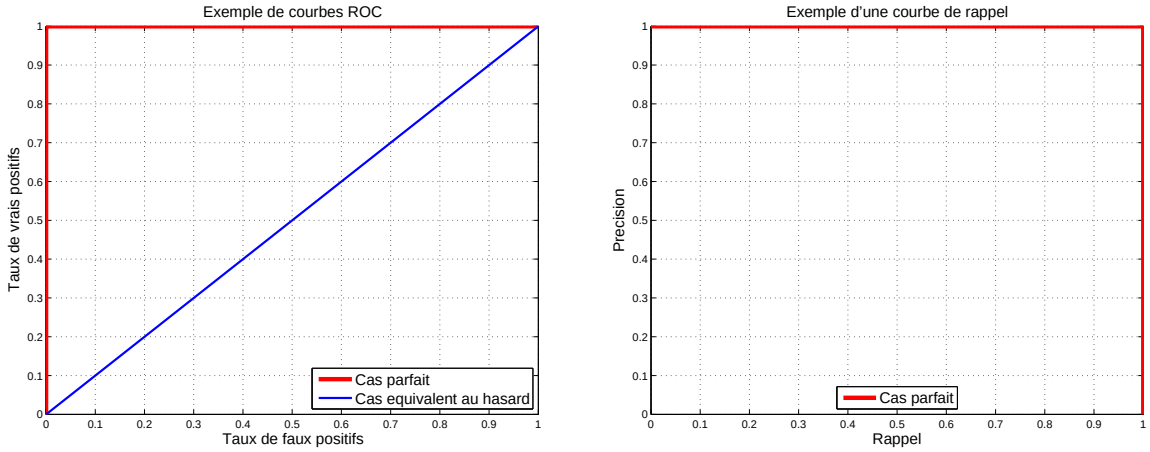


FIGURE 4.3 – Exemples de courbes ROC et de précision/rappel
A droite se trouvent deux courbes ROC et à gauche une courbe de précision/rappel. Les courbes rouges correspondent à un cas idéal. La courbe ROC bleue correspond à un classifieur qui n'est pas meilleur que le hasard.

4.3 Influence de la taille de la base d'apprentissage

Le choix de l'ensemble d'apprentissage n'est pas anodin. En effet, il faut que les images sur lesquelles le classifieur est entraîné pendant l'apprentissage soient proches des cas réels auxquels il sera confronté, tout en décrivant la variété des classes des positifs et des négatifs. C'est pourquoi nous avons choisi la base d'images précédemment présentée (voir section 4.2) car elle a été conçue à partir de séquences vidéo enregistrées par une caméra embarquée en milieu urbain, ce qui est précisément le contexte de l'application visée par cette thèse.
Une fois la base d'images choisie, il faut sélectionner le nombre d'exemples à utiliser. *A priori*,

plus le classifieur « voit » d'exemples, meilleure sera sa capacité à généraliser. Nous avons essayé de démontrer ce phénomène par les courbes suivantes (voir figures 4.4, 4.5, 4.6 et 4.7). Nous avons successivement réalisé plusieurs apprentissages dans les mêmes conditions (même descripteur et même classifieur) mais avec un nombre d'exemples d'apprentissage différent (à chaque fois, nous utilisons autant d'exemples positifs que négatifs). Puis nous avons testé les performances sur la même base de test comprenant 1000 images, comprenant à parts égales des exemples positifs et négatifs.
Les résultats tendent à démontrer ce phénomène. Sur les figures 4.4, 4.5 et 4.6, globalement, les résultats s'améliorent avec le nombre d'exemples compris dans la base d'apprentissage. A 20% de fausses alarmes, en augmentant de 500 à 4000 exemples d'apprentissages, le taux de bonnes détections passe de $57,4\%$ à $69,4\%$ pour un classifieur AdaBoost avec un descripteur par ondelettes de Haar (voir figure 4.4). De même, pour un rappel de 80%, la précision s'améliore de $71,56\%$ à $78,59\%$. Le même constat se fait pour le descripteur par histogrammes de gradients orientés : le taux de bonnes détections passe de 68% à 77% et la précision de $71,56\%$ à $78,59\%$ (voir figure 4.5).
Les résultats s'améliorent également pour les classifieurs à noyau [2]. Pour 20% de fausses alarmes, le taux de bonnes détections passe de $74,2\%$ à $83,2\%$ et pour un rappel de 80%, la précision passe de $85,07\%$ à $87,89\%$ en augmentant l'ensemble d'apprentissage, pour un classifieur SVM avec un descripteur par ondelettes de Haar (voir figure 4.6).
Toutefois, certaines fois, un apprentissage à partir de moins d'exemples d'apprentissage est aussi performant qu'avec plus d'exemples. Prenons par exemple la figure 4.7 : pour un descripteur par histogrammes de gradients orientés, le taux de bonnes détections n'augmente que faiblement (de $87,4\%$ à $90,6\%$) et la précision reste la même. Cela s'explique par le fait que le classifieur a réussi à trouver une bonne séparatrice entre exemples négatifs et positifs, même avec peu d'exemples.
Par contre, lorsqu'un apprentissage sur une base d'images plus conséquente fournit des résultats moins performants, il peut s'agir d'un phénomène de surapprentissage. Ce cas survient quand le classifieur essaye de classer correctement toute la base d'apprentissage et notamment des exemples « difficiles » qui sont encore mal classés. Le classifieur se focalise alors sur ces exemples en ajoutant soit des classifieurs faibles pour l'AdaBoost soit des vecteurs support supplémentaires pour les méthodes à noyau. Ce phénomène détériore ainsi les performances de généralisation du classifieur. L'origine de ces contre-performances n'est toutfois pas clairement définie, et il est également possible qu'un mauvais réglage des classifieurs détériore les résultats (choix du noyau, de la largeur du noyau, ...). Par exemple, sur la figure 4.4, à partir d'une base d'apprentissage de 2000 exemples, le taux de bonnes détections obtenu est de $62,6\%$ pour 20% de fausses alarmes alors que pour apprentissage à partir 1000 exemples, le taux de bonnes détections est de $68,6\%$; la précision se détériore en passant de $70,12\%$ pour un apprentissage sur 1000 exemples à $68,14\%$ pour 2000 exemples (pour un rappel de 80%). De même, sur la

2. Des résultats supplémentaires pour les RVM et les moindres carrés sont présentés en annexe B.1. Les conclusions de cette section sont les mêmes pour ces deux autres classifieurs.

figure 4.7, pour 1000 exemples d'apprentissage, le taux de bonnes détections est de $88,6\%$ alors que pour 2000 exemples, ce taux est de $87,8\%$. Mais la précision reste sensiblement la même. Ces variations ne sont toutefois pas assez importantes pour mettre en doute le constat général observé au premier abord, à savoir que les performances des classifieurs s'améliorent lorsque le nombre d'exemples d'apprentissage augmente.

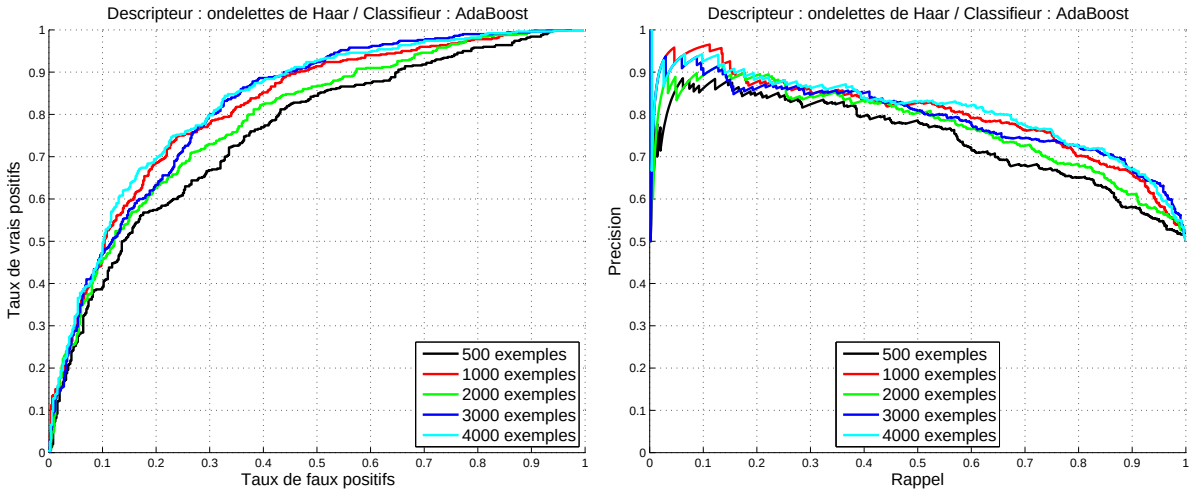


FIGURE 4.4 – Influence de la taille de l'ensemble d'apprentissage : descripteur par ondelettes de Haar et apprentissage par AdaBoost
Les performances du classifieur s'améliorent lorsque le nombre d'exemples d'apprentissage augmente. Toutefois, le classifieur entraîné sur 2000 *exemples a de moins bonnes performances que celui entrainé sur* 1000 *exemples.*

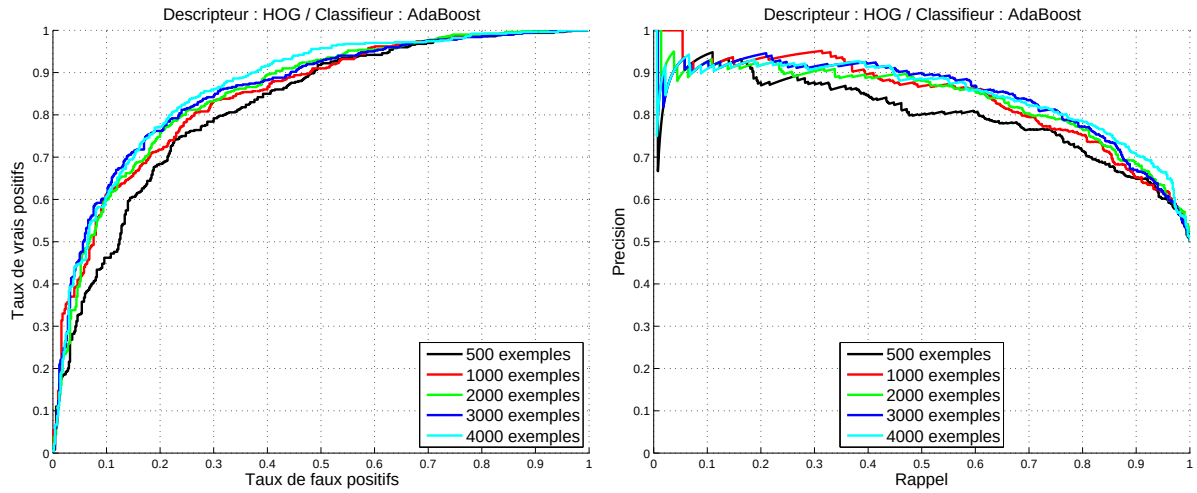


FIGURE 4.5 – Influence de la taille de l'ensemble d'apprentissage : descripteur par histogrammes de gradients orientés et apprentissage par AdaBoost
Globalement, les performances du classifieur s'améliorent lorsque la taille de l'ensemble d'apprentissage augmente.

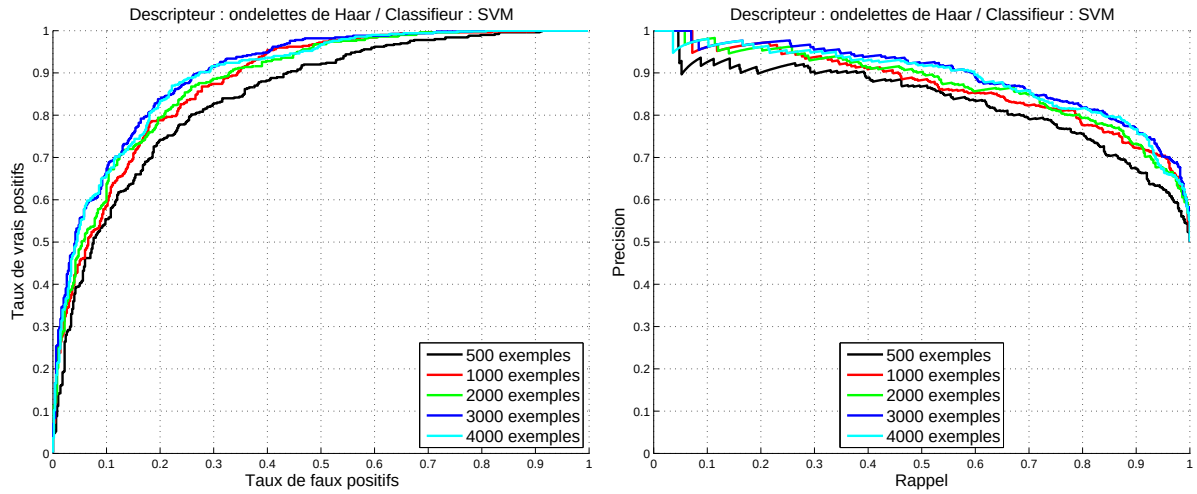


FIGURE 4.6 – Influence de la taille de l'ensemble d'apprentissage : descripteur par ondelettes de Haar et apprentissage par SVM
Même constat que précédemment : les résultats s'améliorent avec la taille de l'ensemble d'apprentissage, même si le gain de performance s'amenuise petit à petit.

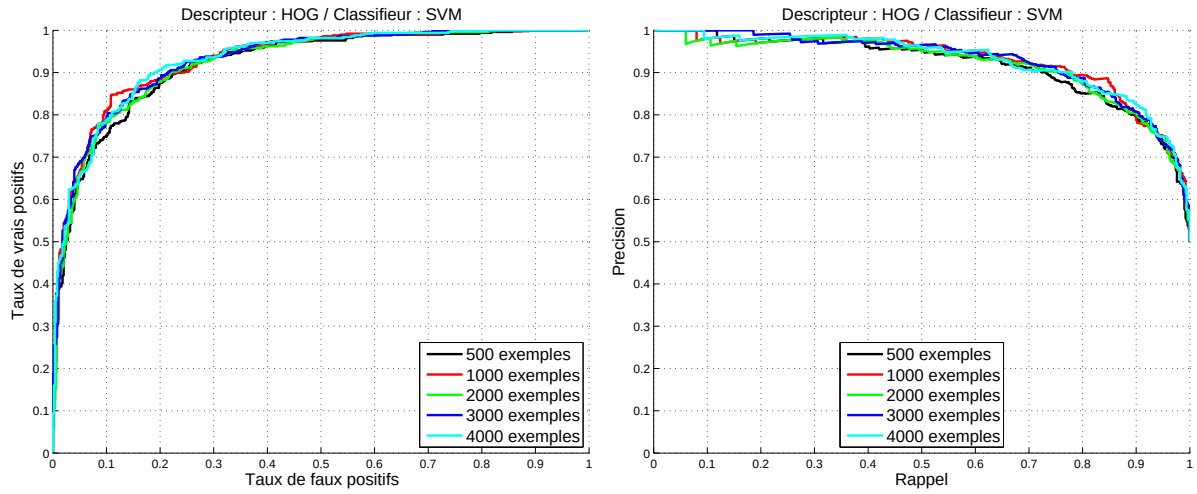


FIGURE 4.7 – Influence de la taille de l'ensemble d'apprentissage : descripteur par histogrammes de gradients orientés et apprentissage par SVM
Les performances sont quasiment équivalentes, même si les classifieurs entraînés sur des bases d'apprentissage plus grandes sont légèrement meilleurs.

4.4 Comparaison des descripteurs

Nous allons ici comparer les performances des trois descripteurs implémentés : histogrammes de gradients, ondelettes de Haar et descripteur par comparaisons de niveaux de gris. Ils ont été associés avec un classifieur de type machine à noyau avec moindres carrés. La base d'apprentissage [68] contient 5 bases, 3 pour mener des apprentissage et 2 pour réaliser des tests (voir paragraphe 4.2.1). Ce classifieur a été entraîné sur deux des bases d'apprentissage disponibles (soit 3 apprentissages différents), puis testé sur les deux bases de tests (soit 2 tests pour chaque apprentissage), ce qui nous donne 6 courbes pour chaque descripteur.
Pour les ondelettes de Haar, nous avons gardé les mêmes paramètres que dans [68]. Deux tailles d'ondelettes sont conservées (4×4 et 8×8) et calculées pour les trois orientations vues précédemment, et avec un recouvrement d'un quart de la taille de l'ondelette. Ce descripteur fournit donc pour chaque imagette un vecteur de caractéristiques de taille 1755.
Pour les histogrammes de gradient, nous avons utilisé des cellules de taille 3×3 sans recouvrement et des histogrammes à 8 barres, sans pondération, ni normalisation. Nous obtenons donc un vecteur de caractéristiques de taille 576 pour une imagette donnée.
Enfin, en ce qui concerne le descripteur binaire, les meilleurs couples de points sont choisis au préalable avec une sélection de variables par AdaBoost (voir section 4.6) qui nous a permis d'obtenir un vecteur de caractéristiques de taille 2468 pour chaque image.
La figure 4.8 présente la courbe moyenne de chacun des 6 tests réalisés pour chaque descripteur. Le premier constat est que ces trois apprentissages ont des performances proches. Les histo-

grammes de gradients orientés semblent toutefois avoir de meilleurs résultats. Le descripteur binaire, malgré son apparente simplicité, apporte des résultats quasi similaires. Les ondelettes de Haar sont légèrement en-dessous mais conservent de très bonnes performances.

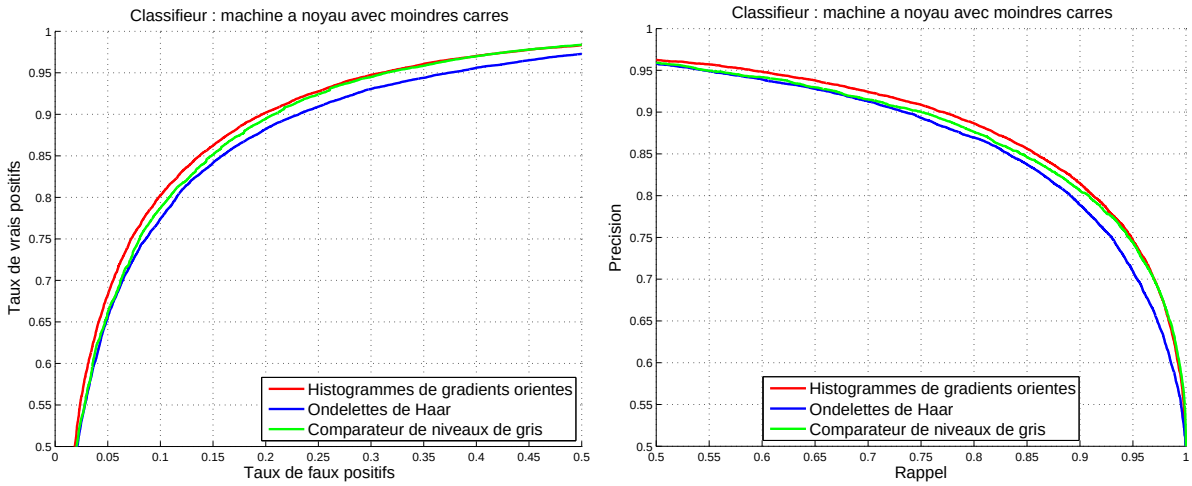


FIGURE 4.8 – Comparaison des descripteurs : classifieur moindres carrés entraîné sur 2 bases
Les histogrammes de gradients donnent de meilleurs résultats, même si le descripteur binaire s'approche des mêmes performances.

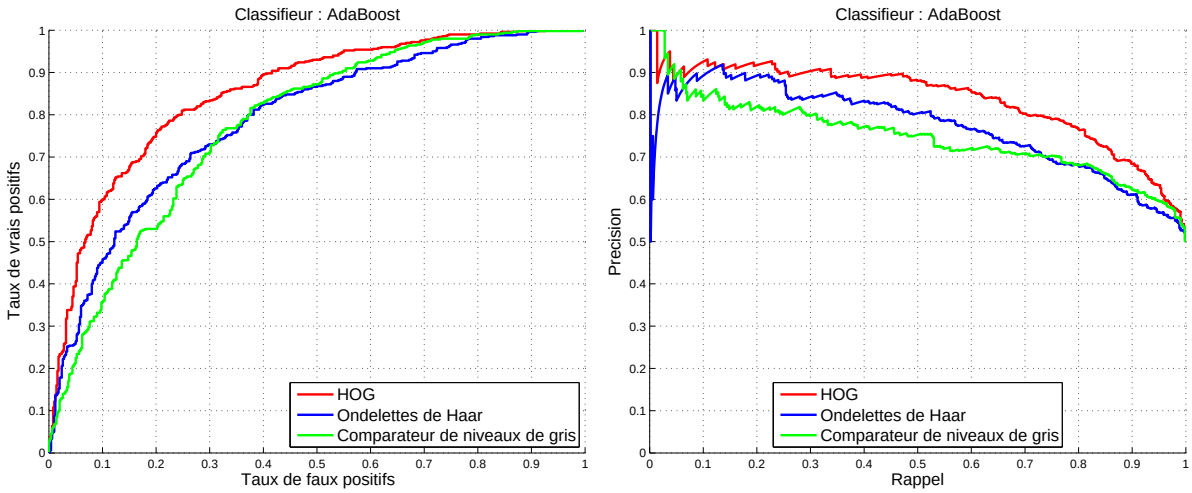


FIGURE 4.9 – Comparaison des descripteurs : classifieur AdaBoost entraîné sur 2000 exemples
Les histogrammes de gradients offrent de meilleurs résultats même avec une base d'apprentissage restreinte. Par contre, les ondelettes de Haar offrent des performances moindres, talonnées par celles du descripteur de niveaux de gris.

Toutefois, ces performances du descripteur binaire sont plus difficiles à atteindre lorsque l'ensemble d'apprentissage comprend moins d'exemples comme sur la figure 4.9 avec un apprentissage par AdaBoost. Le descripteur par histogrammes de gradients orientés permet d'obtenir de bons résultats même avec peu d'exemples ; les ondelettes de Haar sont moins performantes ; et le descripteur de comparaisons de niveaux de gris ne permet pas d'obtenir des résultats aussi performants que précédemment.

4.5 Performances des classifieurs

Différents classifieurs ont été mis en œuvre dans cette thèse (voir chapitre 3). Nous comparons ici leurs performances. Pour chacun d'entre eux, nous les entraînons sur la même base d'apprentissage (1000 images), comprenant autant d'exemples positifs que négatifs, et décrite par le même descripteur. La base de validation est la même pour chaque test réalisé ; elle contient également 1000 images (autant de piétons que de non-piétons) [3].
En ce qui concerne les méthodes à noyau, nous choisissons une fonction RBF (voir equation 3.30). Nous initialisons la valeur de la largeur de la gaussienne σ en utilisant une optimisation non-linéaire en maximisant un critère C empirique basé sur la somme des variances calculées pour chaque ligne de la matrice Φ :

$$\sigma \stackrel{def}{=} \arg\max_{\sigma} \left[C(\sigma)\right] \tag{4.3}$$

avec :

$$C(\sigma) = \frac{1}{T} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\phi_t(\mathbf{x}_i) - \overline{\phi_t}(\mathbf{x}_i)\right)^2 \tag{4.4}$$

et

$$\overline{\phi_t}(\mathbf{x}_i) = \frac{1}{T} \sum_{t=1}^{T} \phi_t(\mathbf{x}_i) \tag{4.5}$$

Pour ajuster sa valeur, nous utilisons les images des trois premiers ensembles d'apprentissage par validation croisée, comme proposé dans [68].
Le premier constat qui s'impose est que l'algorithme AdaBoost est globalement moins performant que les méthodes à noyau proposées. Pour une description par ondelettes de Haar, l'AdaBoost obtient $68,4\%$ de bonnes détections pour 20% de fausses alarmes (avec 947 classifieurs faibles) alors que les machines à noyau atteignent plus de $78,8\%$ et pour un rappel de 80%, l'AdaBoost a une précision de $70,12\%$ contre plus de $77,63\%$ pour les machines à noyau (voir figure 4.10). Les SVM ont sélectionné 824 vecteurs support et les RVM 290 vecteurs « *relevant* ».
De même, pour une description par histogrammes de gradients orientés, à 20% de fausses

3. Des résultats supplémentaires sont proposés en annexe B.2 pour des bases d'apprentissage contenant 2000 exemples.

alarmes, l'AdaBoost atteint $76,2\%$ de bonnes détections (avec 312 classifieurs faibles) alors que les machines à noyau atteignent des taux supérieurs à $87,4\%$; de même pour un taux de rappel de 80%, la précision n'est que de $75,14\%$ pour l'AdaBoost contre plus de $89,49\%$ pour les machines à noyau (voir figure 4.11). Les SVM ont sélectionné 901 vecteurs support et les RVM 1000 (soit tous les vecteurs de la base pour cet apprentissage).
Par contre, il est difficile de départager les trois classifieurs à noyau. Les écarts sont faibles pour être vraiment significatifs. En effet, pour divers points de fonctionnement, chacun des trois classifieurs est tour à tour le plus performant. Les deux tableaux suivants récapitulent les performances pour quelques points de fonctionnement pour le descripteur par ondelettes de Haar (voir tableau 4.2) et pour le descripteur par histogrammes de gradients orientés (4.3).
Le choix du meilleur classifieur dépend donc de l'application visée, du point de fonctionnement souhaité (moins de fausses alarmes ou tous les piétons détectés...) mais également des ressources disponibles pour pouvoir implémenter une méthode.

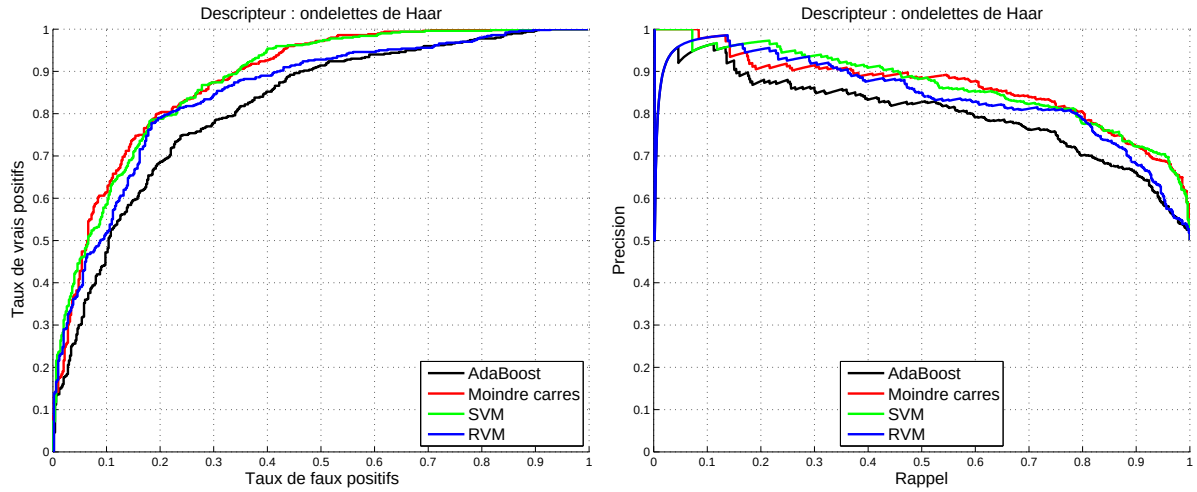


FIGURE 4.10 – Comparaisons des classifieurs : descripteur par ondelettes de Haar et apprentissage sur 1000 exemples
L'AdaBoost est moins performant que les classifieurs à noyau qui offrent quant à eux des performances équivalentes.

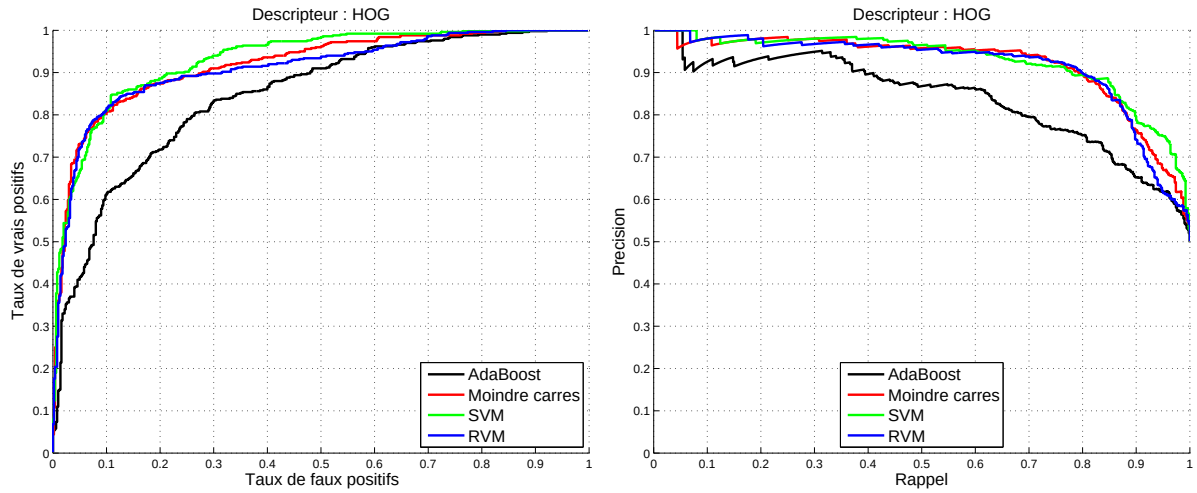


FIGURE 4.11 – Comparaisons des classifieurs : descripteur par histogrammes de gradients orientés et apprentissage sur 1000 exemples
Même constat que sur la figure 4.10 : là encore, l'AdaBoost est moins performant que les trois autres classifieurs à noyau ; et il est toujours aussi difficile de départager les classifieurs à noyau.

	Bonnes détections pour 10% de fausses alarmes	Précision pour un rappel de 90%	Bonnes détections pour 20% de fausses alarmes	Précision pour un rappel de 80%	Bonnes détections pour 30% de fausses alarmes	Précision pour un rappel de 70%
AdaBoost	47,20	66,18	68,60	70,12	77,80	76,20
SVM	58,60	**72,35**	78,80	77,71	**87,40**	82,31
RVM	51,60	68,08	79,00	78,14	84,60	81,16
M. carrés	**61,40**	72,30	**80,20**	**80,48**	**87,40**	**83,89**

TABLE 4.2 – Comparaisons des classifieurs avec un descripteur par ondelettes de Haar
La base d'apprentissage comprenait 1000 images. Les résultats sont exprimés en %. Les chiffres en gras correspondent aux meilleures performances pour le critère correspondant. L'AdaBoost est moins performant que les trois méthodes à noyau proposées, qui sont, quant à elles, difficiles à départager tant les résultats sont proches.

	Bonnes détections pour 10% de fausses alarmes	Précision pour un rappel de 90%	Bonnes détections pour 20% de fausses alarmes	Précision pour un rappel de 80%	Bonnes détections pour 30% de fausses alarmes	Précision pour un rappel de 70%
AdaBoost	60,00	65,17	76,40	75,14	88,40	79,50
SVM	81,00	**79,75**	**88,60**	89,49	**94,00**	92,08
RVM	**81,60**	74,14	87,60	**89,89**	89,80	93,83
M. carrés	**80,80**	76,40	87,40	89,69	91	**94,07**

TABLE 4.3 – Comparaisons des classifieurs avec un descripteur par histogrammes de gradients orientés
La base d'apprentissage comprenait 1000 *images. Les résultats sont exprimés en* $\%$*. Les chiffres en gras correspondent aux meilleures performances pour le critère correspondant. L'AdaBoost donne là encore de moins bons résultats que les trois méthodes à noyau qui ont des résultats quasiment similaires.*

4.6 Sélection de variables

Lors de la création de tous ces systèmes, des problèmes de dimensionnalité apparaissent. En effet, utiliser de grandes bases d'apprentissage comme proposé dans la section 3.2 peut poser des problèmes de stockage mémoire, qui sont encore plus flagrants lorsque le descripteur utilisé donne un grand vecteur d'observation. Or toutes ces descriptions ne sont pas forcément utiles à la tâche de classification et il devient donc intéressant de supprimer ces données inutiles ; pour cela, nous avons recours à une méthode de sélection de variables, que nous présentons dans cette section.

4.6.1 Intérêt

Pour construire un modèle d'objets, les méthodes présentées précédemment dans le chapitre 3 font appel à un ensemble d'apprentissage composé de deux classes d'objets : les exemples positifs (tout ce qu'est l'objet recherché) et d'exemples négatifs (tout ce que n'est pas l'objet recherché). Le plus souvent, cet ensemble d'apprentissage est conséquent afin de prendre en compte la grande variabilité des deux classes d'objets. De plus, chaque exemple (négatif ou positif) est souvent décrit par un vecteur de grande taille, dont souvent seul un sous-échantillon est pertinent pour reconnaître la classe de l'objet. Dans la cadre de cette thèse, où il faut reconnaître des piétons, apprendre un modèle générique est un challenge particulièrement difficile à cause de la grande variabilité de l'apparence des piétons (la taille, les habits, la couleur de peau...) ; c'est pourquoi nous avons besoin - en fonction de la taille du vecteur d'observation fourni par le descripteur - d'une sélection de variables qui permettra de réduire le nombre de caractéristiques

avant d'entraîner le classifieur.
Le but recherché est double. Il s'agit tout d'abord de réduire le temps de calcul du modèle en diminuant la dimensionnalité des données d'entrée. Même s'il est possible d'objecter que, pour un traitement hors ligne, le temps de calcul n'est pas le principal problème.
Le deuxième enjeu est d'améliorer les résultats du classifieur. En effet, des études [9, 13, 60] ont montré l'efficacité de la sélection de caractéristiques pour améliorer les performances du classifieur : la présence de données inutiles peut perturber le classifieur et des espaces mémoire sont employés inutilement.

4.6.2 État de l'art

Plusieurs approches ont été proposées pour réduire ce nombre de caractéristiques [12, 37]. Elles sont de deux types : les méthodes par filtrage (*filters*) et les méthodes d'enveloppage (*wrappers*).

Les méthodes par filtrage utilisent seulement l'ensemble d'apprentissage. Elles traitent toutes les données avant le début de l'apprentissage et conservent uniquement les caractéristiques utiles. Le plus connu est l'algorithme *Relief*, introduit par Kira et Rendell [50] et amélioré par Konenko [52], qui calcule un critère de pertinence pour chaque caractéristique de l'ensemble d'apprentissage. Une autre approche présentée par Hall [38] utilise un score de corrélation pour réduire l'ensemble d'apprentissage. Une extension de cette méthode a été développée dans [100] pour les grands ensembles d'apprentissage. Mais certains développent leurs propres critères de sélection adaptés à une application précise [25, 42, 45, 61, 83].

Les méthodes d'enveloppage font la sélection de caractéristiques au fur et à mesure de l'apprentissage. En outre, elles utilisent le processus lui-même pour sélectionner les caractéristique pertinentes [51]. Des solutions ont été apportées pour les SVM. Weston [98] explore l'espace des paramètres par une descente de gradient et fixe un seuil d'arrêt sur l'erreur d'apprentissage. Rakotomamonjy [78] propose un critère de sélection basé sur l'influence de la variable sur la règle de décision d'un classifieur SVM. Généralement ces méthodes qui prennent en compte l'ensemble d'apprentissage et le classifieur en même temps donnent de bons résultats mais induisent de forts temps de calcul.

Dans une optique de gain de temps, le challenge est donc de trouver une sélection de caractéristiques qui travaille indépendamment du processus d'apprentissage. Mais ne pas prendre en compte le classifieur est le principal inconvénient des méthodes par filtrage : le risque est de sélectionner des caractéristiques qui ne seraient finalement pas utiles. Pour garantir la pertinence des caractéristiques conservées pour le classifieur, le meilleur outil est ce classifieur lui-même.

4.6.3 AdaBoost pour la sélection de variables

Les algorithmes de type AdaBoost peuvent également être utilisés pour la sélection de caractéristiques. Dans [94], une cascade AdaBoost est utilisée avec une description d'images par ondelettes de Haar. A chaque itération, une taille d'ondelette est choisie et les images sont subdivisées en plusieurs sous-fenêtres. A chaque étape, le classifieur rejette les sous-fenêtres non-informatives et le processus continue. Cette cascade - composée de 38 couches - permet ainsi de limiter le nombre de caractéristiques par ondelettes pour le dernier classifieur AdaBoost de la cascade.
Cette méthode en cascade a été reprise dans [55] pour constituer une cascade de trois couches. Les deux premiers étages de la cascade sont des algorithmes AdaBoost qui rejettent les caractéristiques - par ondelettes de Haar - non-informatives de l'image. La dernière étape consiste à entraîner un classifieur SVM à partir des caractéristiques résultantes des étapes précédentes de la cascade. Cette méthode permet de réduire le nombre de caractéristiques avant d'arriver au classifieur à noyau. Dans [85], le même principe est appliqué mais avec un descripteur par histogrammes de Gabor.
Nous proposons ici d'utiliser l'algorithme AdaBoost pour sélectionner les caractéristiques pertinentes à partir d'un ensemble initial de grande taille. Le but est de diminuer la taille de l'ensemble de l'apprentissage avant d'entraîner un classifieur sur ces données. La chaîne ainsi formée est constituée d'un algorithme AdaBoost qui sélectionne les variables pertinentes puis d'un classifieur à noyau qui apprend sur le nouvel ensemble.
Au départ, chaque caractéristique est vue comme un classifieur faible et l'AdaBoost sélectionne les plus pertinentes (voir algorithme 3). Un ensemble $\mathcal{S}$ comprenant N exemples d'apprentissage de taille $N \times M$ est utilisé. Les caractéristiques choisies par AdaBoost sont celles qui seront conservées pour créer un nouvel ensemble d'apprentissage $\mathcal{S}'$ comprenant toujours N exemples mais de taille $N \times M'$ avec $M' < M$. Ce nouvel ensemble d'apprentissage $\mathcal{S}'$ de dimension réduite est fourni en entrée à un classifieur à noyau qui réalisera l'apprentissage. Pendant la reconnaissance, une nouvelle image sera simplement décrite par les variables choisies par AdaBoost puis, à partir de ces caractéristiques, le classifieur à noyau associera cette image à la classe d'objets adéquate.

4.6.4 Résultats

Nous avons entraîné plusieurs classifieurs après une sélection préalable de variables par AdaBoost. Nous présentons ici les résultats pour les RVM. Chaque classifieur a été entraîné sur la même base d'images comprenant 2000 exemples (moitié piétons, moitié non-piétons) puis testé sur une base de 1000 nouvelles images (même répartition). Les observations effectuées sont valables pour les autres classifieurs (voir annexe B.3).

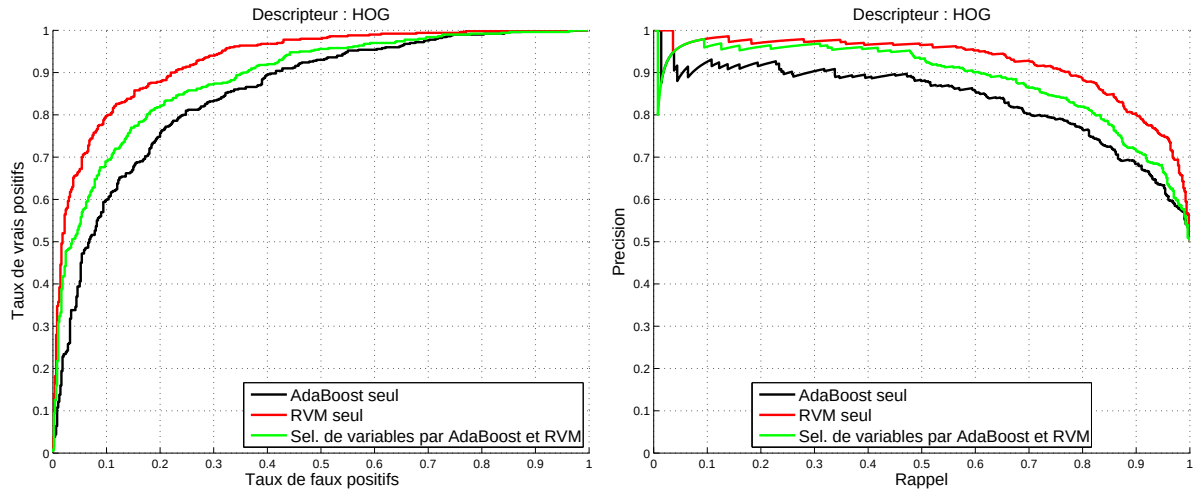


FIGURE 4.12 – AdaBoost pour la sélection de variables : descripteur par histogrammes de gradients orientés et apprentissages sur 2000 exemples
*L'association AdaBoost (pour la sélection de variables) et RVM permet d'améliorer les résultats par rapport à un AdaBoost seul. Les performances de cette association ne parviennent toutefois pas à égaler celles du classifieur RVM seul mais elles sont tout de même satisfaisantes au regard du nombre restreint de variables utilisées (*152 *sur* 576*).*

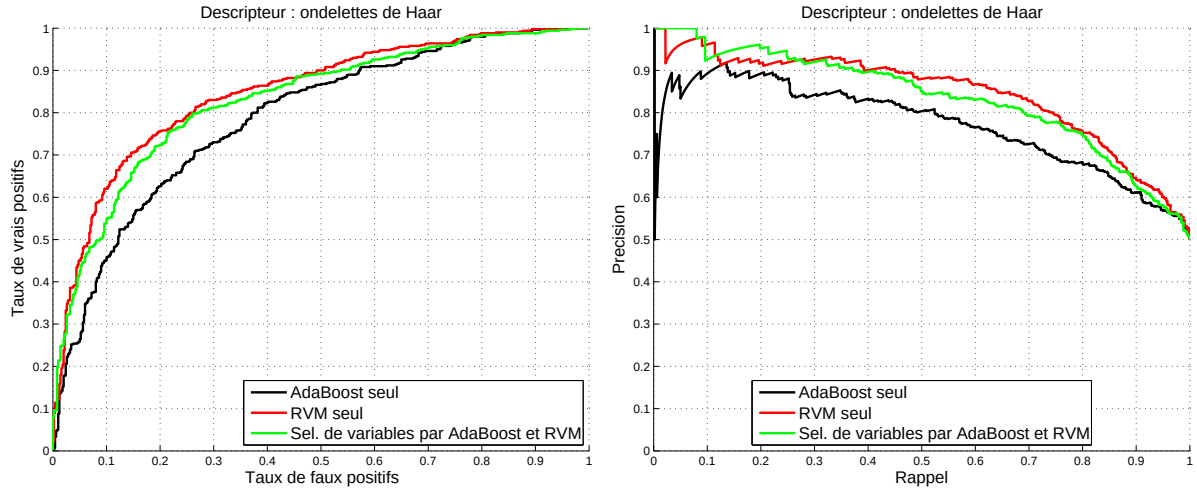


FIGURE 4.13 – AdaBoost pour la sélection de variables : descripteur par ondelettes de Haar et apprentissage sur 2000 exemples
Les constatations sont les mêmes que précédemment. La sélection de variables par AdaBoost retient 863 *variables sur* 2442 *initiales. Les performances sont accrues par rapport à l'AdaBoost seul, mais diminuées par rapport à l'apprentissage des RVM sur l'ensemble d'apprentissage initial.*

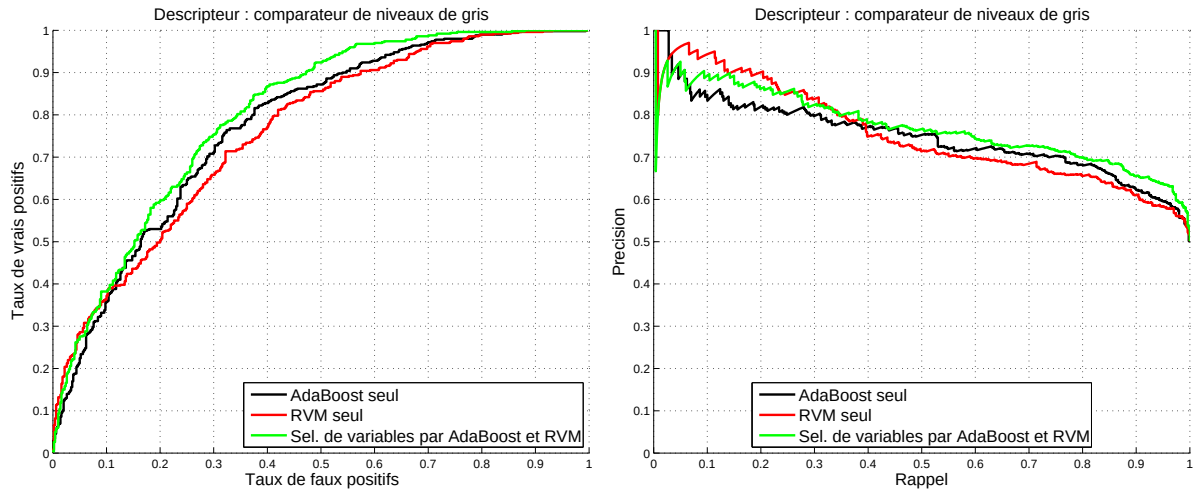


FIGURE 4.14 – AdaBoost pour la sélection de variables : descripteur par comparaisons de niveaux de gris et apprentissages sur 2000 exemples
La sélection préalable de variables permet de réduire de façon conséquente la taille de l'ensemble d'apprentissage en ne conservant que 495 *comparaisons sur les* 16848 *disponibles. Cette réduction permet non seulement d'améliorer les résultats par rapport à l'AdaBoost seul, mais aussi par rapport à l'apprentissage des RVM sur l'ensemble d'apprentissage initial.*

Les résultats sont intéressants. La première constatation est que quelque soit le descripteur ou le classifieur employé, les résultats s'améliorent par rapport à l'AdaBoost seul (voir figure 4.12, 4.13 et 4.14). Par contre, lors de l'apprentissage avec les RVM, avec le descripteur par histogrammes de gradients orientés ou par ondelettes de Haar, cette sélection de variables ne permet pas de conserver les performances obtenues avec l'ensemble d'apprentissage initial (voir tableau 4.4). Mais au regard du nombre de caractéristiques conservées, les résultats sont satisfaisants. Par contre, cette sélection de variables par AdaBoost s'est avérée très bénéfique pour le descripteur par comparaison de niveaux de gris ; les résultats s'améliorent aussi par rapport à l'apprentissage sur l'ensemble initial. Ceci nous permet de conclure que dans le cas de ce descripteur, certaines comparaisons étaient inutiles et perturbaient le travail du classifieur.

	Bonnes détections pour 20% de fausses alarmes	Précision pour un rappel de 80%
Descripteur	Ondelettes de Haar	
Sélection	863 caractéristiques sur 2442	
AdaBoost	62,60	68,26
Sel. variables par AdaBoost + RVM	72,40	74,63
RVM	75,60	75,76
Descripteur	Histogrammes de gradients orientés	
Sélection	152 caractéristiques sur 576	
AdaBoost	75,80	76,78
Sél. variables par AdaBoost + RVM	82,20	81,93
RVM	87,80	88,69
Descripteur	Comparaisons de niveaux de gris	
Sélection	495 caractéristiques sur 16848	
AdaBoost	53,00	68,26
Sel. variables par AdaBoost + RVM	59,80	65,52
RVM	50,40	69,88

TABLE 4.4 – Sélection de variables par AdaBoost : comparatif

Les résultats sont exprimés en %. La sélection de variables par AdaBoost avec un apprentissage avec une machine à noyau améliore les résultats par rapport à l'AdaBoost et se rapproche beaucoup des performances du classifieur à noyau sans sélection de variables.

Algorithme 3 AdaBoost pour la sélection de variables ou de classifieurs faibles

ENTRÉES:

Soit un ensemble d'apprentissage tel que
$\mathcal{S} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$,
avec $i \in \{1, ..., N\}$ et $\mathbf{x(i)} \in \mathbb{R}^M$,
et $y_i \in \{+1, -1\}$, label de classe.
pour $i = \{1, ..., N\}$ **faire**
$p_0(\mathbf{x_i}) \leftarrow \frac{1}{M}$
$i = i + 1$
fin pour $i = N$
pour $t = \{1, ..., T\}$ **faire**
Tirer un échantillon d'apprentissage S_t dans S selon les probabilités p_t.
Calculer l'erreur de classification de chaque descripteur sur le sous-échantillon.
Sélectionner h_t qui minimise cette erreur.
Soit ϵ_t l'erreur apparente de h_t sur S.
Calculer $\alpha_t = \frac{1}{2} ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$.
pour $i = \{1, ..., N\}$ **faire**
- si $h_t(\mathbf{x}_i) = y_i$ (bien classé par h_t) :

$$p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{-\alpha_t}$$

- si $h_t(\mathbf{x}_i) \neq y_i$ (mal classé par h_t) :

$$p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{+\alpha_t}$$

- $i = i + 1$
fin pour $i = N$
$r = r + 1$
fin pour $t = T$
NB : Z_t est une valeur de normalisation telle que $\sum_{i=1}^{n} p_t(\mathbf{x_i}) = 1$.

SORTIES: Un ensemble d'apprentissage $\mathcal{S}'$ tel que :
- pour la sélection de variables : $\mathcal{S}' = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$
avec $i \in \{1, ..., N\}$ et $(\mathbf{x}_i) \in \mathbb{R}^T$
- pour la sélection de classifieurs faibles : $\mathcal{S}' = \{(\mathbf{h(x_1)}, y_1), ..., (\mathbf{h(x_N)}, y_N)\}$
avec $i \in \{1, ..., N\}$ et $\mathbf{h(x_i)} \in \{-1, 1\}^T$

4.7 Combinaison d'AdaBoost avec une méthode à noyau

4.7.1 Méthode

Nous avons vu dans la section précédente que l'AdaBoost peut être employé pour faire de la sélection de variables, mais il peut également sélectionner des classifieurs faibles.
Dans le cas de la sélection de variables, la taille de l'ensemble d'apprentissage du départ est diminuée en réduisant le nombre de caractéristiques décrivant chaque exemple, puis un classifieur est entraîné sur l'ensemble ainsi réduit.
Nous proposons une méthode originale qui consiste à fournir directement la sortie binaire des classifieurs faibles de l'AdaBoost au classifieur à noyau comme nouvelles données d'apprentissage. Soit $\mathcal{S}$ l'ensemble d'apprentissage tel que :

$$\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1,...,N} \text{ avec } \mathbf{x}_i \in \mathbb{R}^M \tag{4.6}$$

L'AdaBoost sélectionne T classifieurs faibles ayant chacun une règle de décision h_t binaire ($t = 1, ..., T$) (voir section 3.3.2). Le nouvel ensemble d'apprentissage $\mathcal{S}'$ est constitué des sorties de l'AdaBoost soit :

$$\mathcal{S}' = \{\mathbf{h}(\mathbf{x_i}), y_i\}_{i=1,...,N} \tag{4.7}$$

avec :

$$\mathbf{h}(\mathbf{x_i}) = \left\{h_1(\mathbf{x}_i^1), ..., h_T(\mathbf{x}_i^T)\right\} \text{ et } \mathbf{h}(\mathbf{x_i}) \in \{-1, 1\}^T \tag{4.8}$$

(voir algorithme 3). Le nouveau modèle est :

$$H(\mathbf{x}) = \sum_{i=1}^{N} w_i K(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}_i)) \tag{4.9}$$

Non seulement l'ensemble d'apprentissage $\mathcal{S}'$ a une dimension réduite, mais les données de l'ensemble d'apprentissage sont binarisées. Un tel changement permet de diminuer la dimensionnalité de l'ensemble d'apprentissage mais aussi de simplifier la tâche de classification comme présentée dans la figure 4.15 : l'introduction des machines à noyau permet de créer des séparateurs non-linéaires dans l'espace des classifieurs faibles. Enfin, le fait de prendre des données binaires permet de faciliter les calculs et de diminuer le nombre de stockages mémoire utilisés.

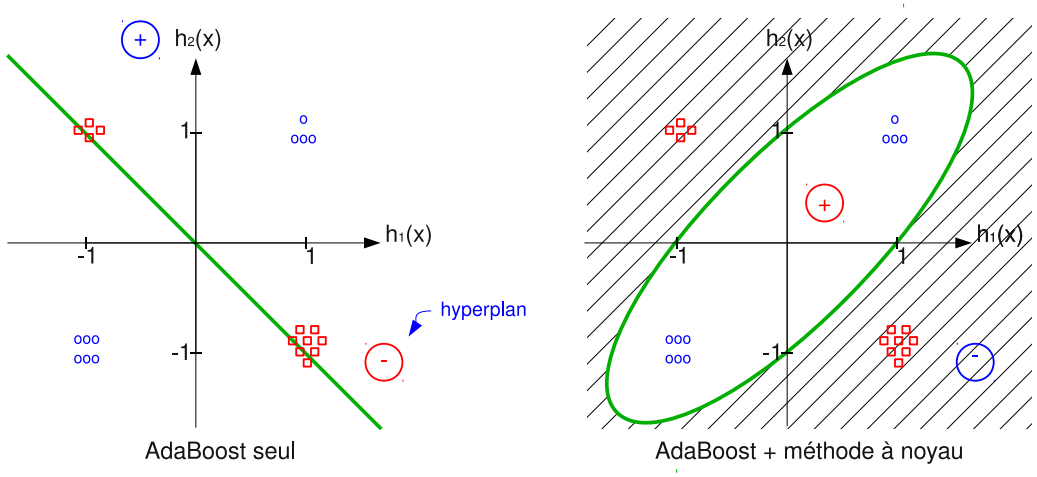


FIGURE 4.15 – Association AdaBoost et méthode à noyau pour la sélection de classifieurs faibles
Les exemples positifs sont désignés par des carrés rouges, et les exemples négatifs par des ronds bleus. $h_1(x)$ *et* $h_2(x)$ *sont des classifieurs faibles sélectionnés par AdaBoost qui retournent la valeur* 1 *pour les exemples classifiés comme positifs et* -1 *pour les exemples classifiés comme négatifs. Dans l'espace des classifieurs faibles, l'AdaBoost fournit un séparateur linéaire et certains exemples sont mal-classés. L'introduction des machines à noyau permet de créer des séparateurs non-linéaires dans l'espace des classifieurs faibles et de classer correctement tous les exemples.*

4.7.2 Résultats

Pour le descripteur par comparaisons de niveaux de gris, les résultats sont identiques à de la sélection de variables par AdaBoost puisqu'il s'agit déjà d'un descripteur binaire. Il est possible de se référer aux figures 4.14 et B.11 en annexe B.3 où la sélection, efficace, améliore les performances des classifieurs.
Nous avons aussi testé cette méthode de sélection de classifieurs faibles dans le cas des ondelettes de Haar qui fournissent des valeurs réelles avec les classifieurs à noyau précédents (voir figures 4.16, 4.18 et 4.17). La base d'apprentissage est constituée de 1000 images, moitié exemples positifs, moitié exemples négatifs. La base de test est aussi composée de 1000 images, réparties à l'identique. L'association AdaBoost pour la sélection de classifieurs faibles et machine à noyau est meilleure que celle avec de l'AdaBoost pour la sélection de variables. Elle permet de diminuer de façon significative la dimensionnalité des vecteurs d'entrée dans le classifieur à noyau tout en améliorant les résultats par rapport aux méthodes « AdaBoost seul » et sélection de variables par « AdaBoost et machine à noyau ». Les résultats par rapport à l'utilisation d'une machine à noyau sur l'ensemble d'apprentissage initial sont équivalents en termes de bonnes détections par rapport aux fausses alarmes mais offrent des performances

accrues en termes de précision/rappel (voir tableau 4.5). Cette méthode est donc une approche très intéressante pour réduire la taille des vecteurs d'entrée dans un classifieur à noyau. De plus, le passage à des données binaires est un atout pour implémenter un système de reconnaissance sur du matériel industriel ayant peu de capacités de mémoire et de calcul (processeurs de faibles puissances).
Nous ne présentons pas ici de résultats pour le descripteur à histogrammes de gradient, car les tests réalisés n'ont pas été concluants. En effet, le vecteur de caractéristiques étant déjà de faible dimension pour les images considérées (de taille 576), le passage à des données binaires par une sélection de classifieurs faibles a trop diminué l'information pour réussir à générer un apprentissage. Des tests complémentaires pour des images de plus grande dimension seraient certainement nécessaires pour évaluer les performances de cette méthode avec un descripteur à histogrammes de gradient de taille plus conséquente.

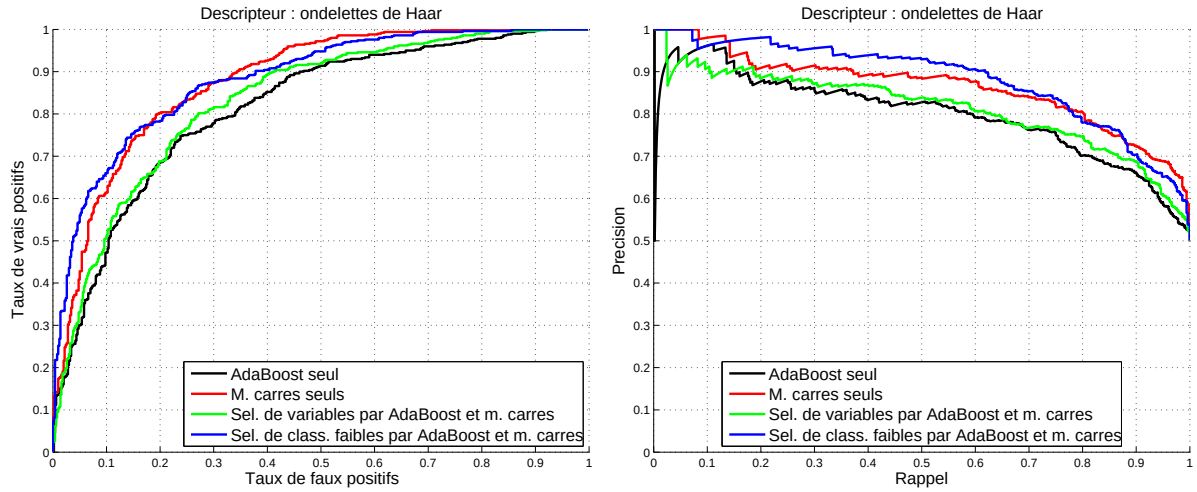


FIGURE 4.16 – Sélection de classifieurs faibles : descripteur par ondelettes de Haar et apprentissage par moindres carrés sur 1000 exemples
La méthode proposée atteint des performances équivalentes voire supérieures - notamment pour la précision - à celle de l'apprentissage par moindres carrés sans réduction de la taille de l'ensemble d'apprentissage.

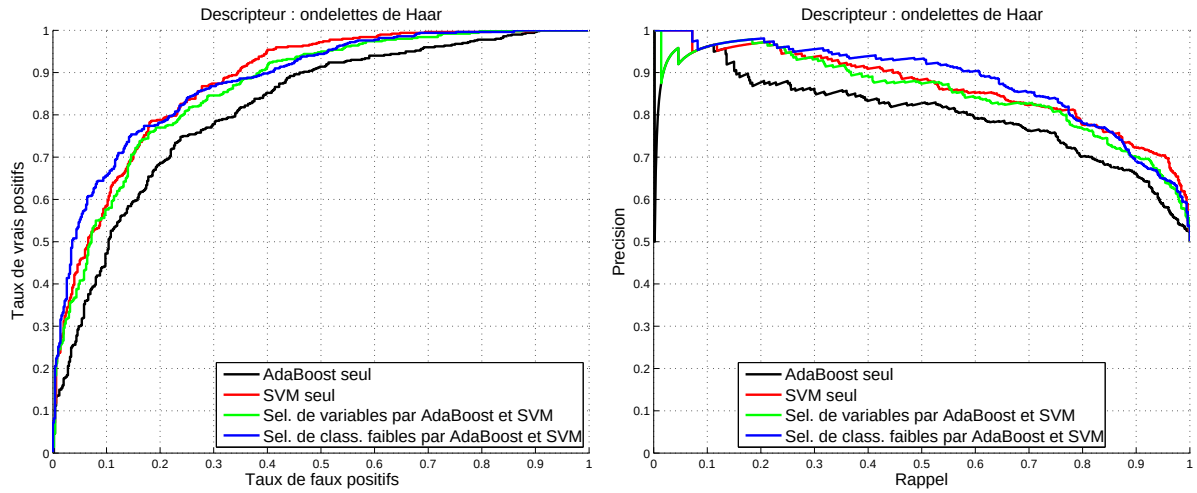


FIGURE 4.17 – Sélection de classifieurs faibles : descripteur par ondelettes de Haar et apprentissage par SVM sur 1000 exemples
La sélection de classifieurs faibles et SVM a d'aussi bons résultats que les SVM sans sélection préalable pour le taux de bonnes détections et offre des performances accrues en termes de précision/rappel.

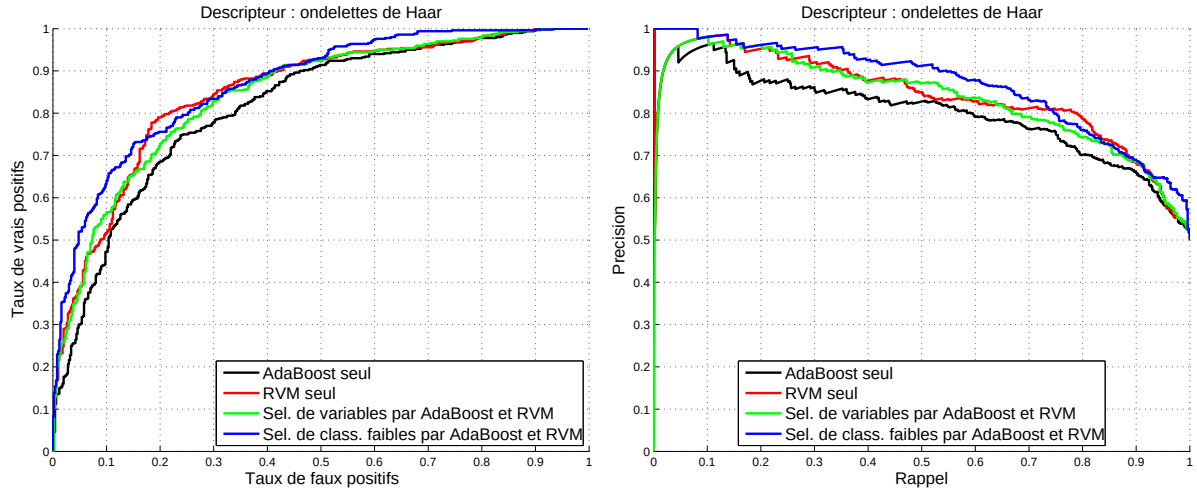


FIGURE 4.18 – Sélection de classifieurs faibles : descripteur par ondelettes de Haar et apprentissage par RVM sur 1000 exemples
La courbe ROC reste sensiblement la même entre la méthode proposée et les RVM ; par contre la courbe précision/rappel de cette méthode est meilleure que celle des RVM.

	Bonnes détections pour 20% de fausses alarmes	Précision pour un rappel de 80%	Bonnes détections pour 10% de fausses alarmes	Précision pour un rappel de 90%
AdaBoost seul	68,6	70,12	47,2	66,18
Moindres carrés	**80,2**	**80,48**	61,4	**72,3**
Avec sél. variables	68,8	74,97	51,2	68,7
Avec sél. class. faibles	78,4	77,93	**66**	70,38
SVM	**78,8**	77,63	58,6	**72,35**
Avec sél. variables	77	76,78	57,6	70,09
Avec sél. class. faibles	78,2	**78,12**	**65,6**	69,02
RVM	**79**	**78,74**	51,8	68,08
Avec sél. variables	72,8	74,3	56,6	68,49
Avec sél. class. faibles	75,6	76	**64**	**69,02**

TABLE 4.5 – Sélection de classifieurs faibles par AdaBoost : comparatif
Les résultats sont exprimés en %. Les chiffres en gras correspondent aux meilleures performances pour le critère correspondant. L'association de sélection de classifieurs faibles par AdaBoost et machine à noyau atteint des performances quasi égales à celles du classifieur sans aucune sélection.

4.8 Association de descripteurs

Nous avons vu comment utiliser combiner l'Adaboost à un classifieur à noyau pour améliorer les résultats tout en diminuant le nombre de caractéristiques. Il est également envisageable d'associer plusieurs descripteurs pour capturer le plus d'informations pertinentes. Cette section est une première étude utilisant des méthodes simples d'association pour tester les éventuels bénéfices d'une telle combinaison.
Dans [32] et [69], deux types de descripteurs sont utilisés et concaténés afin d'améliorer les résultats de la classification. Au lieu de concaténer les différents descripteurs pour chaque image, ce qui donnerait un vecteur de caractéristiques de très grande dimension pour chaque objet et difficile à utiliser pour un classifieur, nous proposons de fusionner les résultats des différents classifieurs (chacun utilisant un descripteur différent).
La première solution envisagée est une simple moyenne algébrique. La règle de décision est fournie par :

$$y_{combinaison1} = \frac{1}{C}\sum_{i=1}^{C} y_i \tag{4.10}$$

C représente le nombre de classifieurs fusionnés.
Une deuxième méthode d'association consiste à choisir parmi les différentes réponses des clas-

sifieurs, celle dont la valeur absolue est maximale, soit :

$$y_{combinaison2} = \arg \max_{i=1...C} \|y_i\| \tag{4.11}$$

A partir des apprentissages réalisés précédemment, nous avons établi plusieurs combinaisons. La figure 4.19 présente l'association de trois classifieurs par AdaBoost, chacun utilisant un descripteur différent : ondelettes de Haar, histogrammes de gradients orientés et comparaisons de niveaux de gris. Nous constatons que les deux associations apportent un gain pour la reconnaissance. Mais il s'agit de la première méthode de combinaison par la moyenne qui atteint les meilleurs scores. Pour 10% de fausses alarmes, l'association moyennée des trois classifieurs (chacun basé sur un descripteur différent) atteint $66,8\%$ de bonnes détections, le choix de la valeur maximale donne $60,8\%$ de bonnes détections, tandis que les apprentissages à partir d'un seul descripteur obtiennent des résultats inférieurs. L'analyse de la courbe rappel/précision conduit au même constat : la moyenne des trois donne $72,35\%$ de précision pour un rappel de 90%, le meilleur des trois $68,49\%$ alors qu'avec un seul descripteur, les performances observées sont en dessous ces résultats.
Nous constatons les mêmes résultats avec un classifieur à noyau (au préalable une sélection de variables a été nécessaire pour pouvoir traiter le cas du descripteur binaire). Par exemple, avec un classifieur SVM, pour 10% de fausses alarmes, l'association moyennée obtient $71,4\%$ de bonnes détections, la sélection du meilleur $58,6\%$; les histogrammes de gradients orientés ici sont plus performants que le choix du meilleur ($64,8\%$) mais les deux autres descripteurs sont en-dessous. Pour un taux de rappel de 90%, la précision s'améliore à $75,25\%$ pour la moyenne et à $71,88\%$ pour le meilleur des trois descripteurs.
Il apparaît donc que chacun des trois descripteurs apporte des informations différentes et complémentaires en contribuant à parts égales à améliorer les résultats de classification. Le problème de cette méthode réside dans sa réalisation temps réel. En effet il faut cumuler le temps de calcul des trois descripteurs pour chaque image ; de plus, les vecteurs de caractéristiques des objets seront plus grands et nécessiteront un temps de traitement plus long de la part du classifieur. Or nous ne pouvons pas nous permettre de multiplier les processus dans le cadre d'une application temps réel. Pour tirer partie de ces expérimentations, la solution serait de paralléliser le calcul de chaque descripteur afin de ne pas cumuler sur une seule machine le temps de calcul de chacun de ces descripteurs.
Il serait également bénéfique d'envisager d'autres formes d'association de tous ces descripteurs, mais cela nécessiterait une nouvelle étude approfondie autour de cette question.

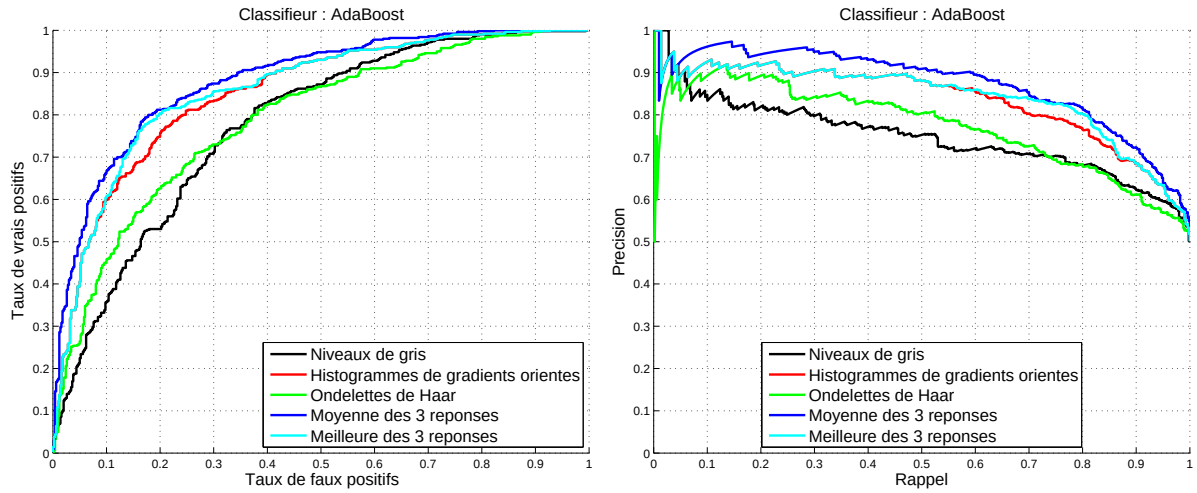


FIGURE 4.19 – Association de descripteurs : classifieur AdaBoost entraîné sur 1000 exemples
L'association moyennée des trois descripteurs offre des performances accrues par rapport à l'utilisation d'un seul descripteur.

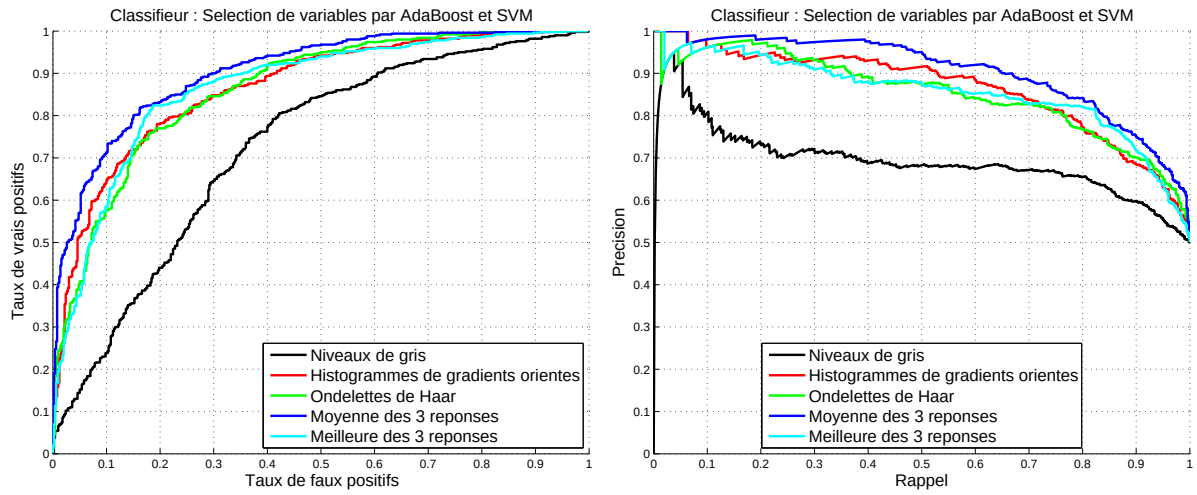


FIGURE 4.20 – Association de descripteurs : classifieur SVM entraîné sur 1000 exemples
Même constat qu'avec un classifieur AdaBoost (voir figure 4.19 : l'association des trois descripteurs obtient avec le classifieur SVM les meilleurs résultats.

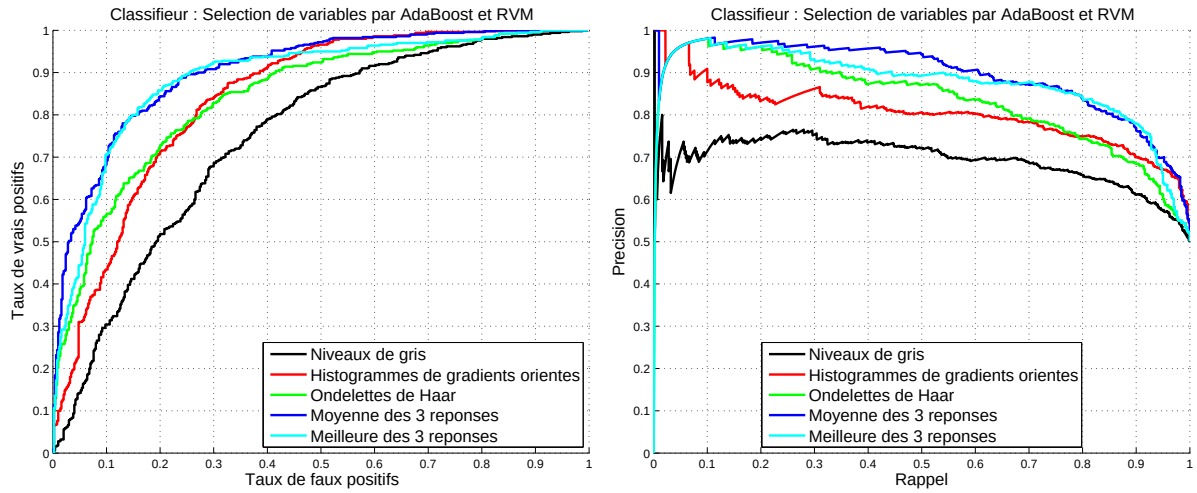


FIGURE 4.21 – Association de descripteurs : classifieur RVM entraîné sur 1000 exemples
Associer les trois classifieurs est plus performant qu'en utiliser un seul, que ce soit en choisissant le meilleur des trois ou en les moyennant.

4.9 Bilan

Au final, que choisir ? Côté classifieur, il apparaît que les méthodes à noyau obtiennent de meilleurs résultats et côté descripteur, ce sont les histogrammes de gradients orientés ; toutefois, les autres méthodes présentées ont des résultats très souvent équivalents. L'objectif de cette thèse étant d'adapter ces travaux dans un système temps réel de reconnaissance de piétons, nous avons donc choisi de garder le descripteur binaire car il est moins coûteux en temps de calcul mais également en espace de stockage. Bien évidemment, pour qu'il soit compatible avec une réalisation temps réel, nous ferons une sélection préalable par AdaBoost pour ne conserver que les couples de points les plus pertinents. Côté classifieur, nous prendrons un classifieur à noyau ; notre choix s'est porté sur le classifieur de type moindres carrés, car son implémentation ne nécessite pas beaucoup de calculs (voir section 3.4.4). Le chapitre suivant (chapitre 5) présente l'intégration de cette méthode dans le système final de détection/reconnaissance de piétons.

Chapitre 5

Application

Le but final de ma thèse est de développer un système complet de détection et de reconnaissance de piétons à partir d'une caméra embarquée dans un véhicule circulant en milieu urbain. Ce travail s'inscrit dans un projet ANR (LOVe), dont le but était de proposer des solutions algorithmiques pour la sécurité routière, et notamment celles des piétons. Ce chapitre présente maintenant le cahier des charges du projet et comment les méthodes présentées précédemment s'y inscrivent. Des expérimentations seront également menées et les résultats analysés.

5.1 Systèmes de détection/reconnaissance de piétons avec une caméra embarquée

Etat de l'art En 1997, Papageorgiou et Poggio introduisent un système de détection et de reconnaissance de piétons [73] dans les images statiques qui va ensuite se généraliser à d'autres objets et devenir très populaire [75]. Ils mettent en place un dictionnaire complet de descripteurs d'image basés sur des ondelettes de Haar (voir section 2.2) avec un classifieur SVM (voir section 3.4.2). Leur méthode est testée pour reconnaître des visages, des piétons et des voitures dans les images. Les auteurs portent également un réel effort d'optimisation afin de réaliser un système permettant de détecter des piétons en temps réel à partir d'une caméra embarquée sur un véhicule dans le cadre d'une application d'aide à la conduite [74].
En 2001, Viola et Jones [94] introduisent un nouveau système de détection, simple de mise en œuvre et réellement dédié aux applications temps réel. Ils développent ainsi une méthode de calcul rapide du descripteur d'ondelettes de Haar de Papageorgiou et Poggio ; celle-ci utilise une nouvelle représentation de l'image appelée « image intégrale »(voir section 2.2.2). Ils proposent également une autre méthode de classification basée sur un algorithme AdaBoost (voir 3.3.2) utilisé en cascade. D'abord implémentée pour faire de la reconnaissance de visages dans les images, leur méthode est ensuite étendue à la détection de piétons dans [96].
En 2005, les travaux de Dalal et Triggs [19, 20] font apparaître un nouveau descripteur d'images pour la reconnaissance de personnes dans les images : les histogrammes de gradients orientés

(voir section 2.3) qui permettent d'exploiter les informations de contours en temps réel. Ce type de descripteurs a ensuite largement été repris. Dans [87] cette méthode est appliquée dans le cas d'un système complet de détection avec une caméra infrarouge. Puis des méthodes associent ces histogrammes à un classifieur AdaBoost : dans [6], les auteurs proposent d'ajouter une composante représentant la norme du gradient aux histogrammes ; une « *image d'histogrammes* » est introduite dans [76] pour les calculer plus rapidement.
Certaines équipes de recherche se sont spécialisées dans les outils d'aide à la conduite.
Les travaux de Gavrila et al. portent sur les véhicules intelligents, notamment au travers de projets européens pour la sécurité routière (voir paragraphe suivant) axés sur la prévention des accidents et sur la protection des piétons, dans le cadre d'un partenariat fort avec l'équipementier automobile allemand *Daimler* [5, 29, 65]. Dans ce contexte, les algorithmes sont mis au point en vue d'être opérationnels sur les véhicules : des capteurs sont embarqués dans les véhicules [30], des bases de données dédiées à l'environnement routier sont réalisées [68], et des tests « grandeur nature » des systèmes sont réalisés. Il s'agit notamment de systèmes mono ou stéréovision permettant une détection de zones d'intérêt, de reconnaissance et de suivi de piétons. La plupart des algorithmes a été évaluée - performances, temps d'exécution - et comparée aux systèmes courants de l'état de l'art cités précédemment. Les travaux [30, 28] ont mis au point des approches de type « *shape contex*t » pour la détection et reconnaissance de piétons dans les images ; il s'agit de représenter les piétons par les différentes formes qu'ils peuvent prendre et de reconnaître les nouveaux objets par un calcul de distance de chanfrein [1] à ces différentes formes apprises ; la validation de la reconnaissance se fait ensuite par un réseau de neurones. Des études [22, 68] ont été menées pour comparer cette méthode à l'état de l'art ; leurs derniers travaux [28] associent cette approche par formes à un modèle bayésien pour estimer la classe de l'objet.
Les travaux de Broggi et al. sont également dédiés aux systèmes embarqués sur les véhicules (détection d'obstacles, de lignes) avec une application sur un véhicule autonome appelé « ARGO ». Des travaux ont aussi été menés pour la détection de piétons, notamment dans les images infrarouges : la détection de piétons se fait soit avec des approches de type « *shape context* » [10], soit avec des méthodes utilisant histogrammes de gradients et SVM [7].
Les travaux de Leibe et al. [57, 58] sur la détection de piétons donnent de très bons résultats ; ils mettent en place un lourd processus d'apprentissage utilisant des « *codebooks* » ou « vocabulaires visuels » décrivant les piétons ; dans une étape suivante, une approche par « *shape context* » est utilisée avec calcul de chanfrein pour améliorer la reconnaissance de l'objet. Le principal inconvénient de ce système est qu'il fait appel à de nombreuses étapes de calculs difficilement applicables en temps réel.
En 2004, Shashua et al. [84] proposent un système complet de détection de piéton pour l'aide à la conduite ; la première phase du processus est une segmentation d'image réalisée avec génération automatique de fenêtres candidates en utilisant des connaissances *a priori* sur la taille d'un piéton. Ensuite intervient le système de reconnaissance : il s'agit d'une approche par com-

1. « *chamfer system* » en anglais.

posants où les différentes parties d'un piéton sont considérées séparément. Pour chacune de ces zones, un descripteur de type « histogrammes de gradients » est utilisé et un algorithme de *Boosting* est mis en œuvre pour reconnaître les piétons. Cette approche est reprise dans [86] en mettant cette fois en œuvre un classifieur de type SVM. Dans [66], Papageorgiou et al. réutilisent le procédé des ondelettes de Haar avec une étude par composants. Ces approches, qui requièrent beaucoup de réglages (choix des composants notamment), sont finalement peu reprises pour la détection de piétons.
Les approches les plus récentes tentent d'associer plusieurs types de descripteurs. Dans [32], Geronimo et al. réalisent un système de détection de piétons en utilisant une association d'ondelettes de Haar et d'histogrammes de gradients orientés dans une cascade de classifieurs AdaBoost, principe à nouveau exploité [44]. Dans [82], des ondelettes de Haar et un descripteur simple de symétrie sont associés avec un classifieur de type SVM. Dans [31], un descripteur d'ondelettes de Haar associé à un algorithme de *Boosting* est d'abord utilisé pour réduire le nombre de fenêtres d'intérêt ; la reconnaissance de piéton se fait ensuite en utilisant un descripteur d'histogrammes de gradients avec un classifieur SVM.

Projets industriels Toutes ces études permettent à des systèmes industriels de voir le jour. Il existe des projets entre industriels et laboratoires de recherche. En Europe, citons notamment :

- *PROTECTOR* (acronyme de *« Preventive Safety for Unprotected Road User »*) en 2000-2003 [29] était un projet allemand et italien comprenant des laboratoires universitaires et des équipementiers automobiles. Il avait pour but de développer un système d'alarme pour les usagers de la route vulnérables, en définissant la faisabilité et les spécifications techniques d'un tel outil, en identifiant des scénarios à risque et en développant un système de perception adéquat ;
- *SAVE-U* (acronyme de *« Sensors and system Architecture for VulnerablE road Users protection »*) en 2002-2005 [65] avait pour but de développer une plateforme innovante de capteurs utilisant trois technologies différentes (image visible, infrarouge et radar) et de fusionner les données pour optimiser la détection des usagers de la route vulnérables (voir site web : http ://www.save-u.org/) ;
- *PReVENT* en 2005-2007 était un programme porté par la Commission Européenne pour contribuer à la sécurité routière en développant et en exposant des applications et des technologies pour l'améliorer. Onze sous-projets ont été retenus pour travailler sur des technologies de freinage automatique, de détection d'obstacles ou de communication entre les véhicules (voir site web : http ://www.prevent-ip.org/en/home.htm) ;
- *WATCH-OVER* en 2006-2008 [5] a été lancé par la Commission Européenne de la Technologie de la Société de l'Information afin de développer un système coopératif pour la prévention des accidents impliquant les usagers de la route dans les zones urbaines et extra-urbaines (voir site web : http ://www.watchover-eu.org/index.html) ;
- *Aktiv* (acronyme pour *« Adaptive and Cooperative Technologies for the Intelligent Traffic »*) en 2008-2010 est un projet de recherche qui comprend 29 partenaires. Le but est

d'améliorer la sécurité et le flot du trafic routier en concevant un nouveau système d'aide à la conduite, basé notamment sur les technologies de l'information et de la communication, qui permettra de développer de futures applications coopératives entre les véhicules (voir site web : http ://www.aktiv-online.org/index.html).

Il existe aussi des sociétés dont l'activité est dédiée vers l'aide à la conduite. Ainsi en 1999, Shashua co-fonde « *Mobileye* », qui propose des outils d'aide à la conduite pour des marques automobiles comme *BMW* ou *Volvo* . Ils ont ainsi proposé un système de détection de piétons à partir d'un système de vision (voir site web : http ://www.mobileye.com/).

5.2 Le projet *LOVe*

Ma thèse se déroule dans le cadre d'un projet du programme *PREDIT*[2] de l'Agence National pour la Recherche (*ANR*) sur des « Logiciels d'Observation des Vulnérables » (acronyme *LOVe*). Ce projet a débuté en septembre 2006 et comprend de nombreux partenaires : des laboratoires de recherche (comme le *LASMEA*), des industriels équipementiers automobiles (*RENAULT*, *VALEO*) et est financé par l'*ANR* et divers pôles de compétitivité (*System@tic*[3], *ViaMéca*[4], *mov'eo*[5].

5.2.1 Contexte

Ce projet s'inscrit dans une problématique forte à l'heure actuelle, à savoir la sécurité routière. Il s'agit plus particulièrement d'apporter une solution pour la protection des piétons et de lutter contre les collisions voitures/piétons. En effet, les piétons représentent environ 900 tués par an en France sur les routes lors de collisions avec des automobiles.

En vue de répondre aux prochaines normes européennes *Euro NCAP* (2011) qui régissent les normes de sécurité pour les véhicules européens, les deux équipementiers automobiles souhaitent mettre en œuvre des solutions pour anticiper et atténuer les collisions frontales avec des piétons. En effet, le crash test *Euro Ncap* intègre un test de collision avec les personnes. En 2011, une législation spécifique pour la protection des piétons va être établie et les limites physiologiques générées lors d'une collision d'un piéton sur l'avant d'un véhicule vont être précisées. Ces limites vont se traduire par des contraintes de construction des éléments qui constituent l'avant des voitures.

2. Programme de Recherche, d'Expérimentation et D'Innovation dans les Transports terrestres, initié et conduit par les ministères chargés de la recherche, des transports, de l'environnement et de l'industrie, l'ADEME et l'ANVAR.

3. Pôle de compétitivité de la région Ile-de-France.

4. Pôle de compétitivité de mécanique.

5. Pôle de compétitivité en recherche et développement automobile et transports publics.

5.2.2 Les objectifs

Deux manières de répondre de à cette problématique sont envisageables : en réalisant soit une sécurité passive, soit active. La sécurité passive consiste à réduire les conséquences d'un accident lorsqu'il n'a pas pu être évité. Une des solutions envisagées est donc la construction d'éléments passifs à l'avant du véhicule qui absorberont le choc de la collision et réduiront les dommages causés aux personnes. Des alternatives peuvent également être envisagées comme le déploiement d'un airbag sous le capot avant du véhicule.
La sécurité active a pour but quant à elle d'éviter les accidents. Les solutions sont multiples : de l'avertissement du conducteur en cas de situation dangereuse au freinage automatique du véhicule. Mais pour tous les cas où des systèmes doivent être activés, la détection, la reconnaissance et le suivi préalables des piétons sont primordiaux et c'est là tout l'enjeu du projet *LOVe*. Le but du projet est donc de développer des systèmes robustes et fiables qui permettront d'équiper les voitures courantes.
La haute technologie s'est déjà installée dans les voitures pour des questions de confort et/ou de sécurité, regroupée sous le terme d'« aide à la conduite ». Il s'agit de systèmes de sécurité dite active qui informent ou assistent le conducteur afin d'éviter les situations dangereuses pouvant mener à un accident. Des systèmes de toute sorte équipent déjà certaines voitures : citons par exemple les limiteurs de vitesse, les contrôles de trajectoire (appelé *Electronic Stability Program* (ESP)), l'aide à la navigation (*Global Positioning System* (GPS)), les avertisseurs de déviation de trajectoire, l'allumage automatique des feux de croisement, les détecteurs de pluie pour le déclenchement automatique des essuie-glaces, les radars de recul, l'alerte de franchissement involontaire de ligne (AFIL)...
Pour créer un outil d'aide à la conduite qui évite ou atténue les collisions entre un véhicule et un piéton, il est donc nécessaire d'avoir au préalable un système qui détecte ces piétons. Le projet *LOVe* s'intègre dans cette thématique. Il doit permettre la création d'un système capable de détecter des piétons pouvant potentiellement entrer en collision avec un véhicule.

5.2.3 Le cahier des charges

Il s'agit plus précisément de fournir des solutions algorithmes et de traitement de données qui permettent de détecter, localiser, reconnaître et suivre des piétons. Pour cela, différents capteurs sont intégrés au véhicule : odomètre, télémètre laser, caméras et radar. Le contexte de recherche est celui d'une voiture circulant en milieu urbain à vitesse modérée (inférieure à 50 km/h). L'objectif est de détecter les piétons qui se trouvent devant le véhicule et pouvant entrer en collision avec celui-ci. En effet, c'est en ville qu'ont lieu le plus d'accidents avec des piétons : le risque d'accident pour un piéton est cinq fois plus élevé en milieu urbain qu'en rase campagne et 68% des piétons sont tués en ville [6]. Cela peut s'expliquer par le fait que la circulation est plus importante en ville et qu'elle recèle bien plus d'éléments qu'un cadre rural que le conducteur doit prendre en compte. Du coup, il est beaucoup plus difficile pour lui de

6. Ces statistiques sont disponibles sur http ://www.securiteroutiere.gouv.fr/observatoire.

percevoir la présence d'un piéton.
Mais ce qui est difficile pour l'être humain l'est aussi pour un système d'intelligence artificielle. Un cahier des charges a été mis en place. Le but est de mettre au point un système d'aide à la conduite qui avertira le conducteur de la présence d'un piéton à risque. Il est essentiel que ce système soit fiable afin que l'automobiliste puisse vraiment prendre en compte les alertes. Des limitations ont été fixées afin de favoriser la réalisation de certains objectifs. En effet, les piétons à détecter impérativement sont entièrement visibles et se tiennent debout, quelque soit leur orientation (de face, de dos ou de profil). Par contre, la détection des piétons masqués n'est pas obligatoire (voir figure 5.1). La taille des piétons suit des paramètres standard disponibles dans le tableau 5.1.

taille	entre 1 et 2 m
largeur (aux épaules)	entre 30 et 70 cm

TABLE 5.1 – Caractéristiques d'un piéton à détecter dans les images

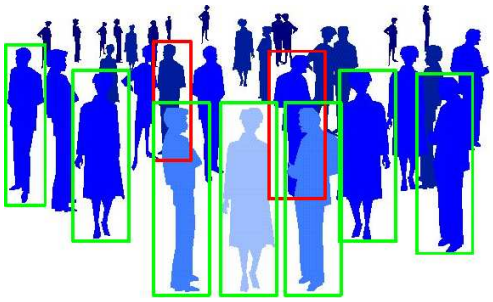


FIGURE 5.1 – Cahier des charges *LOVe* (1) : les piétons
Les piétons que le système doit détecter sont en vert ; par contre la détection de piétons masqués (en rouge) n'est pas requise.

Les piétons qui risquent d'entrer en collision avec le véhicule sont situés immédiatement devant la voiture. Le cahier des charges *LOVe* indique qu'il faut les détecter dans l'espace qu'aura parcouru le véhicule en 2 secondes. Un véhicule se déplace en ville à une vitesse inférieure ou égale à 50 km/h, ce qui donne un espace de recherche qui s'étend jusqu'à 28 mètres devant le véhicule. En considérant qu'un piéton se déplace à une vitesse d'environ 2 m/s de façon perpendiculaire à la trajectoire du véhicule, il faut alors élargir la zone sur une largeur de 10 mètres pour anticiper son déplacement

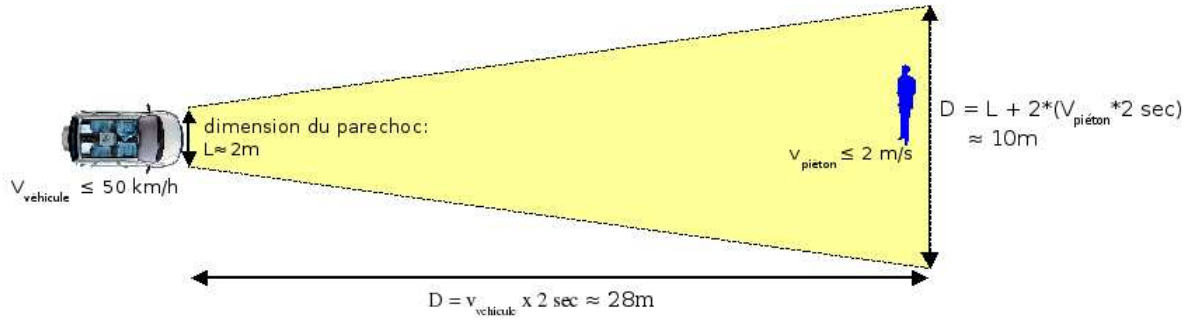


FIGURE 5.2 – Cahier des charges *LOVe* (2) : zone de détection
Le système doit détecter les piétons dans la zone jaune.

5.3 Le système monovision

Ma contribution dans le projet *LOVe* est de fournir des algorithmes permettant la reconnaissance de piétons à partir d'une seule caméra.
Cette dernière est embarquée à bord d'un véhicule et fixée au niveau du rétroviseur intérieur derrière le pare-brise.
Les difficultés sont nombreuses. Tout d'abord, le véhicule est en mouvement et ceci complique énormément la tâche, en particulier pour des capteurs de vision ; le décor change toujours et très rapidement. Certaines solutions très usitées pour des caméras fixes ne peuvent pas être employées telles que de l'extraction de fond qui aurait permis de localiser dans la scène les objets en mouvement.
De plus, les environnements urbains comportent de nombreux éléments qui sont autant de pièges pour un système de reconnaissance visuelle qui peut plus facilement confondre un autre objet avec un piéton.
Enfin, la caméra fournit des images ayant une résolution standard de 640×480 pixels, mais filme la globalité de la scène devant le véhicule. Un piéton présent dans cet environnement ne représente qu'un nombre limité de pixels.
Nous reprenons ici le schéma de fonctionnement présenté dans l'introduction (voir section 1.5). Le système développé dans le projet suit la chaîne présentée, à savoir :

1. segmentation de l'image suivant le cahier des charges *LOVe* pour créer une zone de travail ;
2. découpage de la zone de travail en imagettes ;
3. description de ces imagettes ;
4. reconnaissance de ces imagettes pour détecter les éventuels piétons.

5.3.1 Zone de travail

Pour reconnaître les objets présents dans l'environnement du véhicule, il est tout d'abord nécessaire d'extraire ces objets de l'image envoyée par la caméra. Dans la configuration du projet *LOVe*, celle-ci est fixée au niveau du pare-brise intérieur de la voiture (voir figure 5.3) et filme toute la scène visible à l'avant de celui-ci. Ainsi placée, elle couvre un large champ devant le véhicule. Or, dans le cadre du projet, il faut seulement détecter les piétons dans la trajectoire du véhicule, susceptibles d'être heurtés. Evidemment, le système n'a nullement besoin de chercher d'éventuels piétons dans certaines parties de l'image. Il peut ainsi se limiter à la portion indiquée en jaune dans la figure 5.2, appelée « zone de travail » qui correspond à la zone de détection indiquée dans le cahier des charges *LOVe* (voir paragraphe 5.2.3). Celle-ci peut être mieux délimitée en utilisant l'odométrie de l'automobile (vitesse, angle de braquage des roues) - ce qui a fait l'objet d'une étude à part dans la projet *LOVe*. Mais ici, il faudra se contenter d'une zone de détection approximative en ne prenant en compte que les paramètres de la caméra et son positionnement (l'angle et la hauteur entre la caméra et le sol (considéré localement plat) sont connus). Le véhicule circule en milieu urbain, donc au plus vite à 50 km/h. La zone des « 2 secondes » est fixée à environ 28 mètres devant la voiture.

FIGURE 5.3 – Positionnement de la caméra dans le véhicule
La caméra est fixée à l'intérieur du véhicule au niveau du pare-brise intérieur.

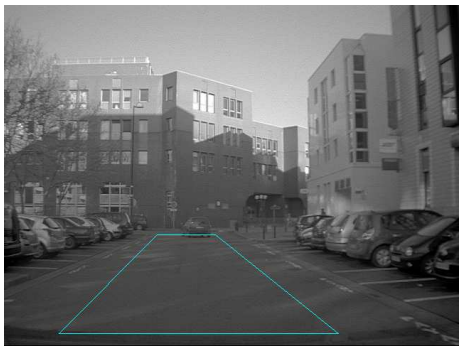


FIGURE 5.4 – Zone de travail
Le système recherche des éventuels piétons dans la zone bleu ciel.

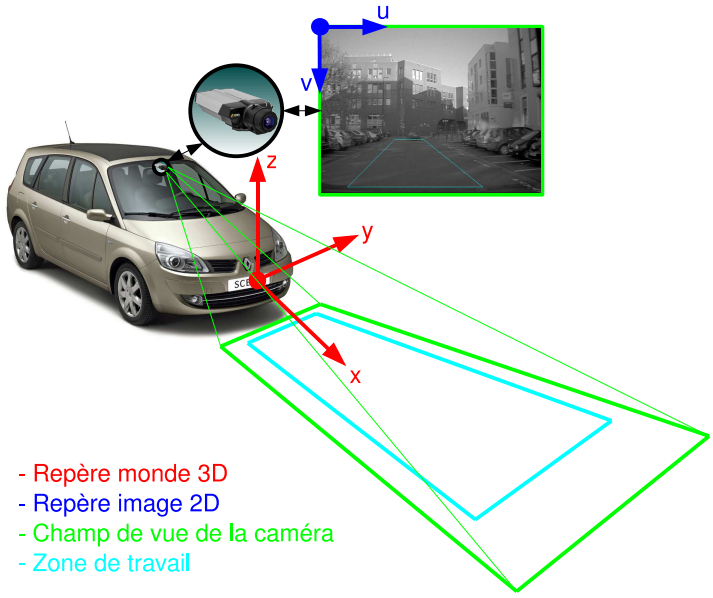


FIGURE 5.5 – Champ de vue de la caméra
La caméra a un champ de vue représenté en vert. Il faut transposer la zone de travail (en bleu ciel) du repère monde 3D (en rouge) au repère image 2D (en bleu).

5.3.2 Calibration de la caméra

Nous savons quelle est la zone de travail où chercher des piétons en 3D. A partir du positionnement de la caméra et de ses paramètres de calibration, il est possible de déterminer sa position 2D dans les images. La caméra est représentée par un modèle sténopé comme montré sur la figure 5.6.

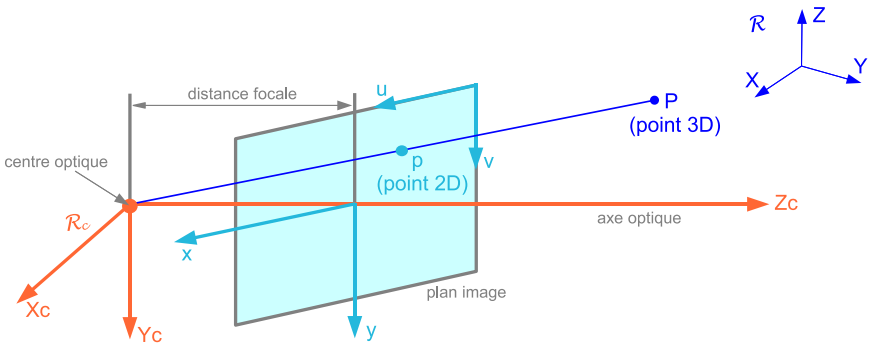


FIGURE 5.6 – Représentation d'une caméra par un modèle sténopé
Passage d'un point réel 3D « P » vers le point image 2D « p ».

La matrice de projection Π permet de passer du monde réel 3D vers le plan image.
Des coordonnées homogènes et des repères cartésiens sont utilisés. Soit p le point image dans le repère 2D et son correspondant P dans le repère 3D avec :

$$p = \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \; et \; P = \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{5.1}$$

Soit Π la matrice de projection telle que :

$$p \propto \Pi P \tag{5.2}$$

Π se décompose ainsi :

$$\Pi = \underbrace{\begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{paramètres \\ intrinsèques}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\substack{projection \\ perspective}} \underbrace{\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{3\times3} & 1 \end{bmatrix}}_{\substack{paramètres \\ extrinsèques}} \tag{5.3}$$

Avec :

- paramètres intrinsèques :
  - f_x et f_y sont les valeurs de la focale (en pixels) suivant les axes x et y respectivement, ;
  - u_0 et v_0 sont les coordonnées (en pixels) dans l'image de l'intersection de l'axe optique et du plan image.
- paramètres extrinsèques :
  - R est la matrice de rotation pour passer des repères R à R_C telle que :

$$\mathtt{R}(\alpha, \beta, \theta) = \begin{bmatrix} cos\theta\ cos\beta & cos\theta\ sin\beta\ sin\alpha - sin\theta\ cos\alpha & cos\theta\ sin\beta\ cos\alpha + sin\theta\ sin\alpha \\ sin\theta\ cos\beta & sin\theta\ sin\beta\ sin\alpha + cos\theta\ cos\alpha & sin\theta\ sin\beta\ cos\alpha - cos\theta\ sin\alpha \\ -sin\beta & cos\beta\ sin\alpha & cos\beta\ cos\alpha \end{bmatrix} \tag{5.4}$$

    avec α,β et θ les angles de rotation autour des axes X,Y et Z respectivement ;
  - $\mathbf{T}$ est le vecteur de translation pour passer des repères R à R_C tel que :

$$\mathbf{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{5.5}$$

    avec T_x,T_y et T_z les translations sur les axes X,Y et Z respectivement.

Les coordonnées des zones de travail sont ainsi obtenues dans le repère caméra. La phase suivante consiste à diviser cette zone d'intérêt en plusieurs objets de la taille d'un piéton potentiel pour que l'algorithme de reconnaissance puisse identifier un éventuel piéton.

5.3.3 Résolution d'un piéton

La caméra retenue pour la tâche de reconnaissance monovision est un capteur vidéo monochrome *Cypress*, filmant à une cadence de 20 images par seconde, d'une résolution de 640×480 pixels et d'un champ de vue de 50 mètres.
Connaissant l'angle d'orientation de la caméra par rapport à l'horizontal, le couple résolution du capteur/champ de vue fixe la taille d'un pixel à une distance donnée. La figure 5.7 montre le nombre de pixels décrivant un piéton dans l'image (cas de la caméra « *Cypress* ») en fonction de la distance d'observation. Pour que les méthodes de reconnaissances fonctionnent correctement, il est indispensable que le piéton soit décrit par au moins 300 pixels. Or, pour les piétons les plus petits, ce seuil est franchi vers 20 mètres. Par contre, pour un piéton de taille moyenne, il est repoussé vers 30 mètres. L'idéal est bien évidemment d'obtenir des images des piétons semblables à celles de la base d'apprentissage (de taille 36×18 soit 648 pixels) pour garantir la pertinence des résultats du classifieur.

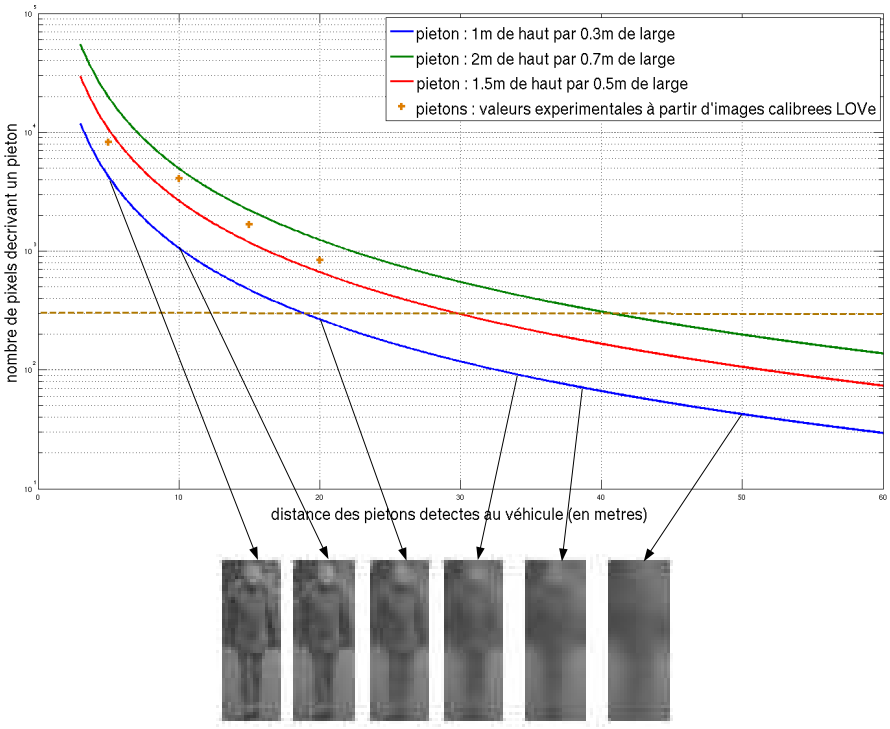


FIGURE 5.7 – Nombre de pixels décrivant un piéton dans l'image en fonction de la distance d'observation
Etude pour la caméra « Cypress » retenue au niveau du cahier des charges, et pour trois gabarits de piétons différents ; un piéton peut être reconnu si son image couvre au moins 300 *pixels.*

5.3.4 Fenêtres glissantes

La zone de travail est maintenant définie dans l'image. Avec une simple caméra, nous ne disposons pas de connaissances *a priori* sur la présence et la localisation d'objets dans cette zone ; nous sommes donc contraints de parcourir toute la zone pour chercher d'éventuels piétons. Le système de reconnaissance développé est basé sur un apprentissage hors ligne ; celui-ci est capable de reconnaître si une imagette (de taille 36×18 pixels) contient un piéton ou non. Il faut donc découper la zone en imagettes ajustées à cette taille. Nous avons réalisé ce découpage sur le principe de fenêtres glissantes (« *sliding windows* » en anglais) qui parcourent toute l'image ou une partie en translation et facteur d'échelle afin de générer un ensemble de sous-images candidates sur lesquelles le processus de reconnaissance (piéton ou non-piéton) est appelé. La figure 5.8 montre ce principe de création des imagettes. Le plan du sol est discrétisé, puis, pour chaque position possible, une fenêtre (ou plusieurs fenêtres si un intervalle sur la taille des piétons à détecter est disponible) englobant un piéton est calculée. Cette dernière est ensuite projetée dans l'image. La fenêtre d'intérêt ainsi obtenue devient une entrée du processus de reconnaissance afin de déterminer si elle contient ou non un piéton.

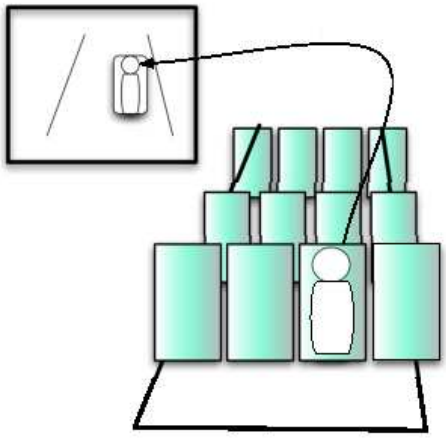


FIGURE 5.8 – Fenêtres glissantes
A partir d'une hypothèse de monde plan et d'un a priori sur la taille des piétons, il est possible de générer des hypothèses de piétons dans l'espace tridimensionnel et de projeter ces hypothèses dans l'image pour obtenir un ensemble de fenêtres candidates.

5.4 Reconnaissance

5.4.1 Fonctionnement

Reprenons le schéma présenté dans le chapitre d'introduction (voir figure 1.3 section 1.5). Les différents éléments présentés dans les chapitres précédents ont été intégrés pour construire la chaîne de développement suivante (voir figure 5.9) :

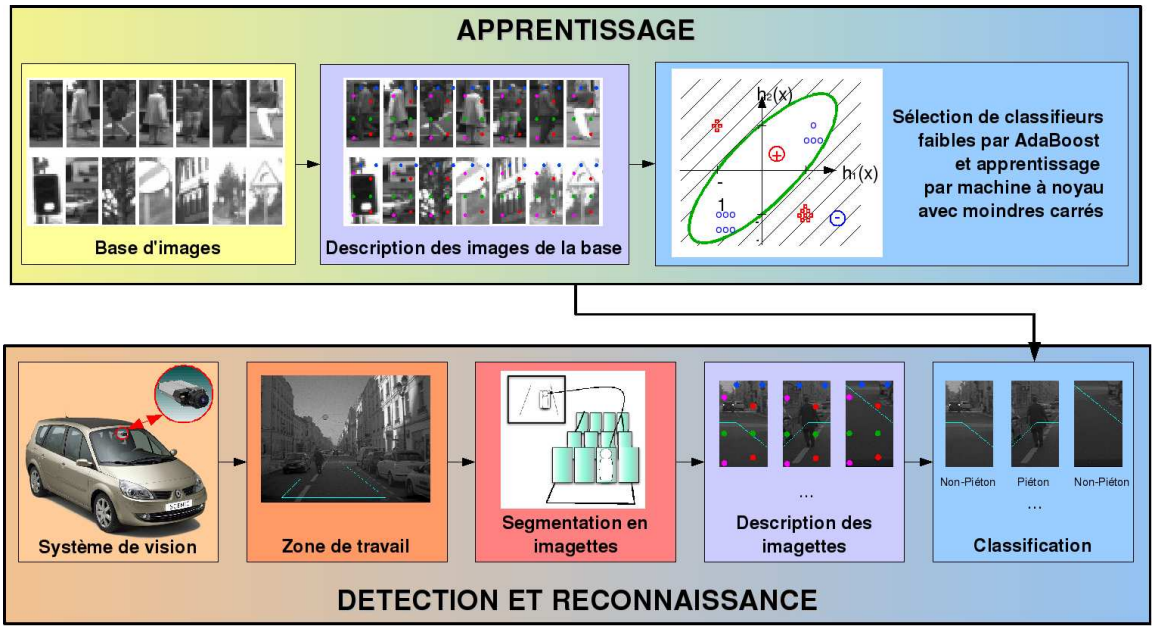


FIGURE 5.9 – Présentation du système complet désiré

La phase d'apprentissage met en œuvre la base d'images présentée dans la section 4.2. Puis nous utilisons le descripteur binaire (voir section 2.5.3), une sélection de classifieurs faibles par AdaBoost (voir section 4.7) et un classifieur de type machine noyau avec moindres carrés (voir section 3.4.4). Le protocole expérimental est le même que dans le chapitre 4.
En reconnaissance, la caméra est embarquée dans le véhicule ; la zone de travail est ensuite découpée en imagettes ; celles-ci sont décrites par le descripteur binaire puis la reconnaissance se fait grâce au classifieur précédemment entraîné.

5.4.2 Résultats

Notre système a été testé sur des séquences vidéo prises dans les conditions du projet. Pour des nécessités techniques, la caméra utilisée par le système monovision est la caméra gauche d'une tête de stéréovision montée sur le toit du véhicule. La zone de recherche a été restreinte à un couloir de recherche de $3\text{m}50$ de large sur une distance de 5m jusqu'à 28m devant le véhicule. Nous explorons 100 imagettes réparties sur toute la zone de recherche.
Nous présentons ici quelques parties représentatives des séquences vidéo où un ou plusieurs piétons sont à détecter. Les passages complets sont présentés en annexe C.
Les zones de recherche des piétons sont encadrées en bleu cyan. Le classifieur retourne un score ramené entre -1 et 1 correspondant à la classe de l'objet ($score < 0$: non-piéton, $score \geqslant 0$: piéton). Les objets identifiés comme des piétons sont encadrés par des boîtes dont le code couleur est le suivant :

- $0 \leqslant score < 0.25$: cadre blanc ;
- $0.25 \leqslant score < 0.5$: cadre jaune ;
- $0.5 \leqslant score < 0.75$: cadre vert ;
- $0.75 \leqslant score < 0.9$: cadre cyan ;
- $0.9 \leqslant score < 1$: cadre bleu.

Les séquences ont été traitées sur un PC quadri-core de $2,4$GHz. L'implémentation a été réalisée en C++, à l'aide de la librairie Nt^2 [24].

Temps de calcul Le tableau 5.2 donne le temps de traitement d'une imagette. Ce temps dépend essentiellement du nombre de vecteurs support conservés. Etant donné que seules 2 secondes sont disponibles pour détecter la présence d'un éventuel piéton, il faut pouvoir traiter une image le plus rapidement possible. Dans la configuration du système présentée ci-dessus (voir paragraphe 5.4.1), le temps de traitement d'une imagette ne permet pas de générer tout un ensemble de fenêtres candidates si ce n'est en parallélisant la reconnaissance sur chacune des imagettes pour ne pas cumuler le temps de traitement de chacune d'entre elles.
Toutefois, des modules utilisant des informations 3D (avec télémètre ou stéréovision) ont aussi été réalisés dans le cadre du projet *LOVe* pour segmenter plus efficacement la scène. En général, ces méthodes n'isolent que quelques objets présents sur la route (moins de 5), et notre système de reconnaissance est alors compatible (voir tableau 5.2).
Le descripteur binaire utilisé permet de limiter les temps de calcul par rapport à un descripteur par histogrammes de gradients orientés dont les temps de calcul sont largement supérieurs dans les mêmes conditions (voir tableau 5.3).

Nombre de vecteurs support	Taille du descripteur	Temps de traitement pour 1 imagette (en ms)	Fps pour 10 objets	Fps pour 3 objets
1000	1000	19,77	5,059	16,864
1000	2468	19,76	5,060	16,868
3000	1000	28,40	3,522	11,739
3000	2468	28,89	3,462	11,540
6000	1000	41,84	2,390	7,967
6000	2468	41,83	2,390	7,968

TABLE 5.2 – Temps de calcul pour le descripteur de comparaisons de niveaux de gris

Nombre de vecteurs support	Taille du descripteur	Temps de traitement pour 1 imagette (en ms)	Fps pour 10 objets	Fps pour 3 objets
1000	576	67,79	1,48	4,92
3000	576	170,84	0,59	1,95
6000	576	322,61	0,31	1,03

TABLE 5.3 – Temps de calcul pour le descripteur d'histogrammes de gradients orientés

Analyse des séquences Voici quelques extraits de séquences vidéo *LOVe* où un ou plusieurs piétons sont à détecter ; une centaine d'imagettes a été générée pour retrouver les éventuels piétons présents dans la zone de travail (en bleu) ; même si nous avons vu que cette solution n'est pas envisageable en temps réel, cela nous permet ici de tester sur des vidéos « réelles » notre système de reconnaissance.
Le tableau suivant (voir tableau 5.4) donne les taux de bonnes détections (les piétons bien reconnus en tant que tels) et de fausses détections (des objets pris pour des piétons alors qu'ils n'en sont pas). Les résultats sont globalement satisfaisants avec en moyenne environ $81,8\%$ de bonnes détections pour $11,1\%$ de fausses détections toutes séquences confondues.

	Séquence 1	Séquence 2	Séquence 3	**Moyenne**
Taux de bonnes détections	70,59%	83,67%	90,91%	**81,72%**
Taux de fausses détections	10,71%	22,45%	0,00%	**11,05%**

TABLE 5.4 – Taux de bonnes détections et de fausses détections pour les 3 séquences

Notre système de reconnaissance a des résultats satisfaisants puisque chacun des piétons a été reconnu sur la majorité des images. Toutefois, des détections manquées sont observables sur certaines images, certainement dues à des conditions d'illumination difficiles (images trop sombres) et à la faible résolution des piétons.
Globalement, le score de reconnaissance des piétons s'améliore au fur et à mesure que le véhicule s'approche d'eux (voir code couleur ci-dessus) (voir images 18, 19 et 20 de la figure 5.10). Toutefois ce score n'est qu'une interprétation du résultat fourni par le classifieur ; il pourrait être plus significatif en le fusionnant par exemple avec un indice correspondant à la qualité de l'image, à l'illumination et/ou à la distance de l'objet.
La génération d'imagettes pose un problème de multiples détections. En effet, un piéton, vu à plusieurs échelles, est reconnu dans chaque imagette où il apparaît même s'il n'est pas vu entièrement. Ce phénomène est assez compréhensible puisque finalement il est logique qu'une portion de piéton soit associée à la classe *piétons* par le classifieur. Cet aspect peut être intéressant pour des occultations mais pose un réel souci si il faut faire un suivi à partir de cette détection. La tâche du module de suivi va alors être plus difficile, puisque il devra initier pour chaque fenêtre candidate une piste qu'il faudra traiter. Bien sûr, il est possible de fusionner des fenêtres proches spatialement mais cela demande encore un temps de traitement précieux ; de plus, peu d'informations sont disponibles pour savoir s'il s'agit d'une multiple détection ou de deux piétons côte à côte (voir figure 5.12). La meilleure solution serait certainement d'avoir une information 3D (stéréovision ou télémètre laser) pour ne générer qu'une imagette pour un objet.
Enfin, des fausses détections sont présentes (voir images 16 et 20 de la figure 5.11). Mais au regard du nombre d'imagettes générées, le nombre de fausses détections reste acceptable. Là encore, une information 3D pourrait nous permettre de limiter ces fausses détections en ne créant pas d'imagette à reconnaître lorsqu'il n'y a pas d'objet sur la route.

5.5 Conclusion

Le système présenté répond au cahier des charges du projet. Les résultats obtenus sont satisfaisants en termes de bonnes détections (en moyenne environ 82%), avec un nombre de fausses détections acceptables (environ 11%). Le temps de traitement de reconnaissance d'un objet est compatible avec le temps réel vidéo, pour détecter un piéton avant la zone des 2 secondes. Par contre la génération de fenêtres s'avère bien trop coûteuse en temps de calcul et crée des fausses détections. Celles-ci pourraient être facilement évitées avec une information 3D quant à la présence d'objets sur la route ce qui permettrait certainement de tirer le meilleur parti du bon taux de reconnaissance de notre système.

FIGURE 5.10 – Extrait de la séquence 1
Détections ratées dans les images 15 *à* 17*, certainement dues à la faible résolution du piéton. Le score de reconnaissance s'améliore au fur et à mesure que la caméra se rapproche du piéton mais de multiples détections apparaissent pour celui-ci (images* 18 *à* 23*).*

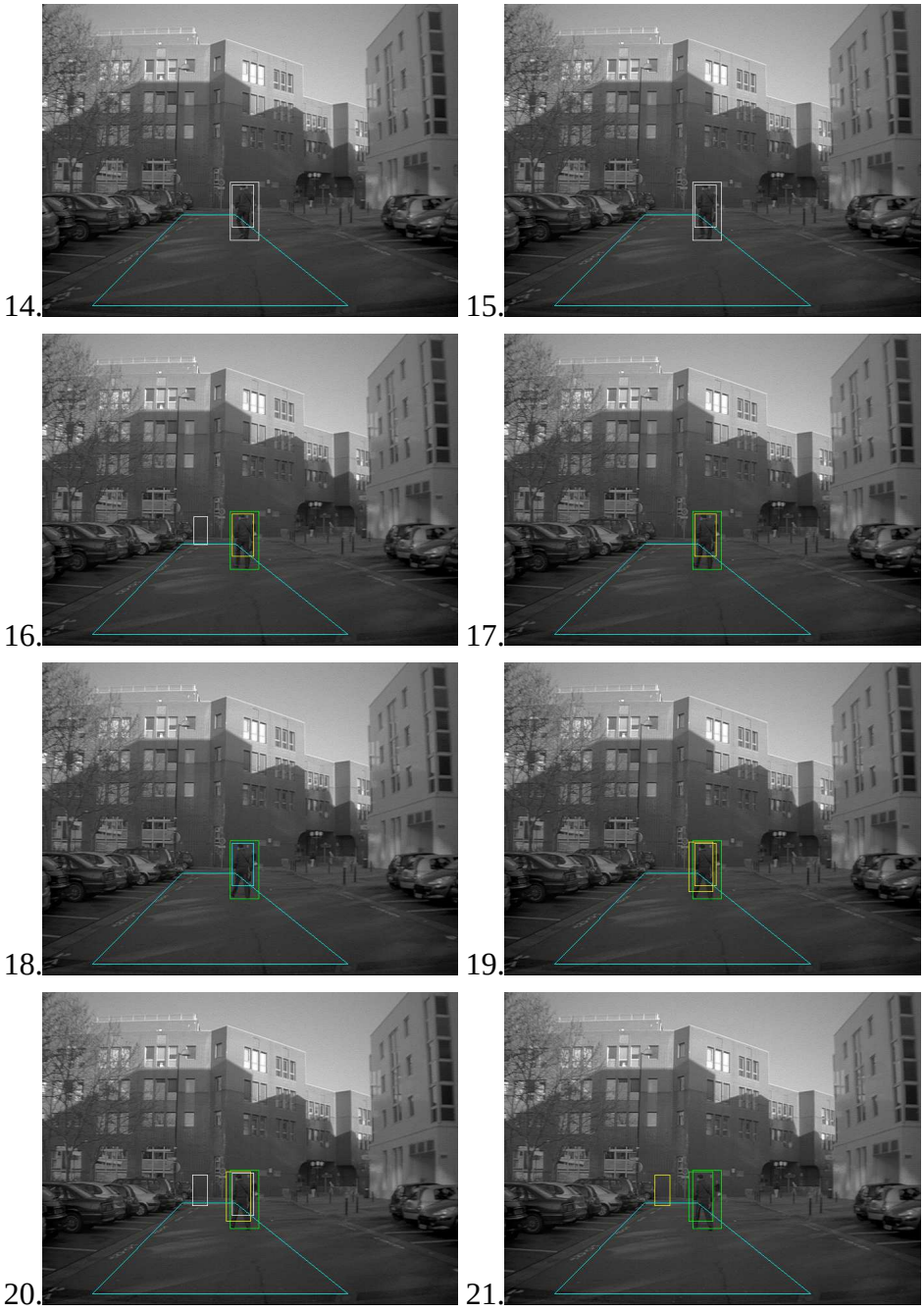


FIGURE 5.11 – Extrait de la séquence 2

Le score de reconnaissance s'améliore au fur et à mesure que la caméra se rapproche du piéton mais de multiples détections apparaissent pour celui-ci. Présence de fausses détections dans les images 16 *et* 20.

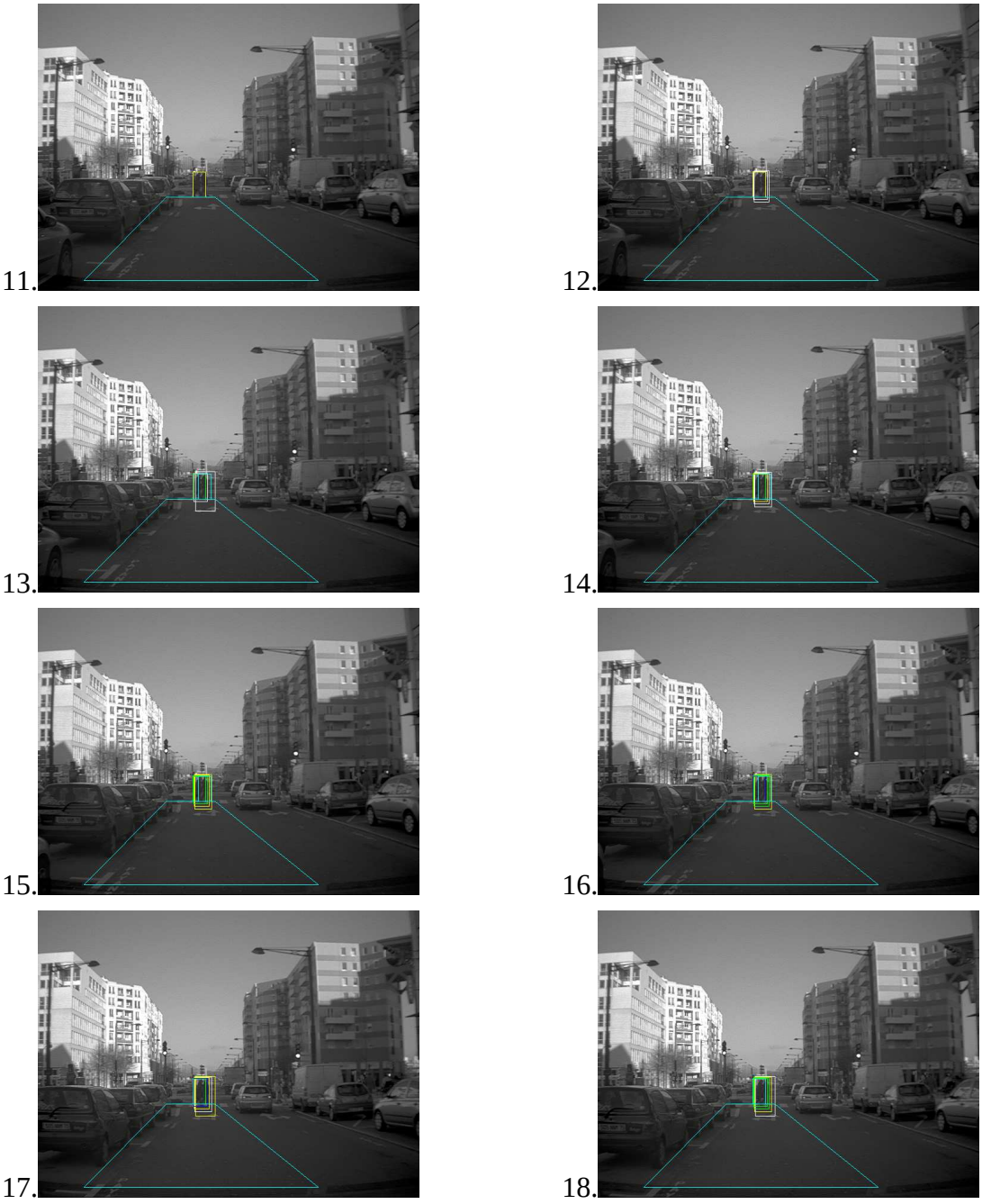


FIGURE 5.12 – Extrait de la séquence 3

Dans cette séquence, deux piétons traversent quasiment côte à côte sur un passage clouté. De multiples détections apparaissent pour ces deux piétons mais il semble très difficile de diviser ces fenêtres en deux piétons distincts.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion

L'objectif de cette thèse était de réaliser de la reconnaissance de piétons. Ce processus devait s'inscrire dans un système complet de détection/reconnaissance traitant en temps-réel les informations provenant d'une caméra embarquée dans un véhicule circulant en milieu urbain.
Nous nous sommes attachés à mettre au point une méthode de reconnaissance de piétons rapide et fiable. Pour cela nous avons dans un premier temps choisi un descripteur d'images, avec pour objectif, qu'il soit suffisamment informatif mais peu coûteux en temps de calcul. Nous avons tout d'abord implémenté des descripteurs classiquement usités dans ce domaine : les ondelettes de Haar et les histogrammes de gradients orientés. Les images à reconnaître lors de l'application finale ont une faible résolution, ce qui nous a poussés à développer un descripteur adapté à ce cas, plus rapide en temps de calcul. C'est pourquoi nous avons exploré une nouvelle piste avec un descripteur binaire.
Vient ensuite la tâche de classification qui utilise les informations fournies par le descripteur en amont. Là encore nous avons mis en œuvre les méthodes les plus populaires de cette dernière décennie : méthodes de *Boosting* avec un algorithme de type AdaBoost et méthodes à noyau avec SVM, RVM ainsi qu'une approche au sens des moindres carrés.
Toutes ces méthodes ont été implémentées pour être testées et validées. De nombreuses combinaisons ont été expérimentées, en interchangeant descripteurs et classifieurs, ce qui a permis de mettre en évidence certains faits.
En ce qui concerne les descripteurs d'images, nous en avons testé trois : les histogrammes de gradients orientés, les ondelettes de Haar et un descripteur de comparaisons de niveaux de gris. Ce sont les histogrammes de gradients qui ont les meilleures performances pour cette application. Toutefois, les deux autres descripteurs obtiennent des résultats proches. Pour le descripteur binaire, il apparaît toutefois difficile de générer un bon modèle à partir de peu d'exemples d'apprentissage, mais, malgré sa simplicité, il obtient des résultats encourageants qui égalent ceux des histogrammes de gradient s'ils sont utilisés sur suffisamment d'exemples. De plus, le pas-

sage vers des applications temps réel est facilité car il est très peu coûteux en temps de calcul (voir 5.4.2). Nous avons aussi exploré une association de descripteurs ; même si les résultats sont prometteurs, l'implémentation temps réel d'une telle méthode n'est pour l'instant pas envisageable.
En classification, les machines à noyau ont montré des performances supérieures à l'AdaBoost. Il est par contre très difficile de départager SVM, RVM ou l'approche par moindres carrés. L'AdaBoost a néanmoins permis de choisir facilement des variables pertinentes pour la classification. Les résultats obtenus avec cette sélection de variables sont quasiment aussi bons que sans sélection de variables mais moins coûteux à l'usage. De plus, l'association de l'AdaBoost avec une méthode à noyau pour la sélection de classifieurs faibles apporte des performances accrues. La réduction de dimensionnalité permet aussi de limiter les temps de calcul dans le but d'une reconnaissance temps réel.
Dans le cadre du projet *LOVe*, nous avons donc choisi d'implémenter le descripteur par niveaux de gris avec une sélection de classifieurs faibles par AdaBoost et apprentissage par machine à noyau avec moindres carrés. Cette méthode a été insérée dans un système complet de détection de piétons à partir d'une caméra embarquée dans un véhicule en milieu urbain. Nous avons mis en place une procédure de fenêtres glissantes pour explorer la scène et détecter les éventuels piétons présents. La contrainte de temps réel est respectée avec un temps de détection inférieur aux 2 secondes imposées, même si des optimisations doivent certainement être encore possibles. Toutefois, la segmentation de l'image proposée ne peut être envisagée telle quelle car elle requiert la reconnaissance de nombreuses imagettes qui n'est alors plus compatible avec une détection temps réel.

6.2 Perspectives

Au terme de ce mémoire, le premier constat, trivial, est que ce type de challenge est très difficile. S'il nous paraît quasiment inné d'interpréter ce que nous voyons et d'agir en conséquence, la tâche est bien plus ardue à partir d'un système de vision.
La grande variabilité des piétons (taille, type d'habillement...) est une grande difficulté pour entraîner un classifieur et trouver un modèle adéquat. De plus, nous sommes confrontés aux problèmes conjoints de la faible résolution de l'image du piéton à celui d'un milieu visuel très riche (panneaux, arbres, façades, véhicules...). Des améliorations pourraient certainement être apportées en utilisant des descripteurs plus performants ou en entraînant des classifieurs sur des bases d'apprentissage gigantesques. Mais l'utilisation de telles méthodes, plus riches en informations, nous est encore impossible pour des questions de temps de calcul. Il faut faire un compromis entre temps de calcul et performances. Une solution envisagée pour diminuer les temps de calcul serait de paralléliser le plus de tâches possibles, comme, par exemple, traiter en simultané la description de chaque imagette. Mais cette parallélisation ne peut pas tout résoudre (le classifieur ne peut être appelé qu'une fois l'image décrite) et demanderait certainement toute une étude à part. En outre, elle pose une difficulté matérielle et financière en

multipliant le nombre de processeurs nécessaires. Des approches en cascade ont été envisagées, mais de part la faible résolution des images, elles se sont avérées peu performantes et ont été écartées. Toutefois, une étude plus approfondie permettrait peut-être de mieux les exploiter.
Le passage à l'application réelle a soulevé un problème pour la segmentation de la scène. La solution de découpage de la scène en imagettes n'est pas viable en l'état actuel. Elle requiert un temps de calcul conséquent qui pourrait peut-être se résoudre en parallélisant le traitement de chaque imagette mais avec les mêmes réserves que celles citées précédemment. Outre ce temps de calcul trop long, nous sommes aussi confrontés à de multiples reconnaissances pour un même piéton, vu à plusieurs échelles. La fusion des imagettes qui correspondent à un même piéton en une seule est peu satisfaisante car elle risque d'être hasardeuse et demanderait encore plus de temps. Une autre solution serait de créer un module plus performant de détection d'objets, mais les méthodes basiques en monovision sont difficilement applicables dans ce contexte : pas d'extraction de fond possible puisque la caméra est en mouvement, et la détection par points d'intérêt ne fonctionne guère car trop d'informations sont présentes en milieu urbain.
Toutefois, cette fonctionnalité n'était pas un enjeu majeur dans le cadre du projet *LOVe*. En effet, d'autres outils ont été spécialement développés pour la détection d'objets. Suite aux travaux menés dans cette thèse, il apparaît qu'une information 3D sur la scène est quasiment indispensable pour effectuer une détection efficace d'objets dans la scène. Citons par exemple des méthodes de disparité par stéréovision [53, 56, 101], ou le traitement de données télémétriques [35] pour détecter des objets sur la route. La fusion multi-capteurs [70] est certainement la clé pour arriver à la meilleure chaîne de traitement. En effet, cela permettrait de mieux tirer parti des informations récoltées par tous les capteurs et programmes. De tels algorithmes ont d'ailleurs été développés dans le projet *LOVe*.
Ces applications multi-capteurs (caméras + télémètre + odométrie) permettraient certainement de mieux détecter les objets présents devant le véhicule, en limitant ainsi le nombre d'objets à reconnaître et le nombre de fausses alarmes. Cependant, elles soulèvent d'autres problèmes. Premièrement, leurs mises en place nécessitent une parfaite calibration de tout le système et il faut que le positionnement ne subisse aucune variation pour garantir le résultat. Mais ce type de contrainte est difficile à respecter pour une instrumentation embarquée dans des véhicules.
Le deuxième problème est financier : instrumenter un véhicule avec autant de capteurs augmente de façon significative le coût du système. En effet, ce type d'études, en lien direct avec des problèmes d'actualité, doit apporter une solution populaire qui pourra être installée sur le plus de voitures possibles. Or multiplier les capteurs pour enrichir les informations sur la scène, interdit un portage sur des automobiles bas de gamme. C'est pourquoi l'utilisation d'une simple caméra a tout son intérêt.
Un autre problème réside dans la présence de fausses détections. Même si la détection en amont du classifieur est meilleure et limite ainsi ces fausses détections, il est quasiment certain qu'il en restera toujours. Une voie envisagée pour répondre à ce problème est d'effectuer un filtrage temporel [23, 36]. Un suivi des objets détectés comme piétons empêcherait de perdre un piéton occulté momentanément ; et cela fournirait des informations supplémentaires primordiales pour limiter les fausses détections.

Fusion de capteurs, suivi, parallélisme sont certainement des clés pour parvenir à un système quasi-parfait même si l'intégration idéale de tous ces processus demandera encore beaucoup de travail. Toutefois, cet axe de recherche ne doit pas limiter les travaux mono-capteur mais l'encourager ; en effet, améliorer ces techniques sera certainement profitable à l'ensemble. Et si le système de détection devient un jour totalement fiable, il faut évidemment envisager d'aller plus loin dans la protection des piétons. Le cadre du projet *LOVe* était de créer un système de détection des piétons. Le but est simplement d'alerter le conducteur, le laissant libre d'action. Mais, dans le cas d'une détection de piétons sûre, il faudra anticiper les éventuelles collisions. Si elles peuvent être évitées, il faudra déterminer quelles actions entreprendre : freiner ou dévier la trajectoire du véhicule par exemple, et alors évaluer tous les risques encourus par le piéton et le conducteur. Si l'impact est inévitable, des solutions ont été évoquées pour atténuer le choc pour le piéton comme le développement d'airbags extérieurs. Toutes ces solutions - et certainement bien d'autres - sont encore au stade d'idées et beaucoup de travaux seront encore indispensables pour voir émerger un véritable système actif de protection des piétons.
Si la recherche se doit d'explorer toutes ces pistes pour s'approcher le plus possible d'une sécurité routière optimale, cela nécessitera encore des efforts d'adaptation considérables pour les voir s'intégrer dans nos véhicules.

Annexe A

Théorème de Mercer

Le théorème de Mercer donne les conditions pour qu'une fonction K - appelée fonction noyau - soit équivalente à un produit scalaire.
Rappelons tout d'abord la définition d'une matrice définie positive :

Matrice définie positive Une matrice $\mathtt{M}$ de dimension $(n \times n)$ dans $\mathbb{R}$ est définie positive SSI :

$$\forall \mathbf{v} \in \mathbb{R}^n \quad \mathbf{v}^T \mathtt{M} \mathbf{v} \geqslant 0 \tag{A.1}$$

Le théorème de Mercer fournit la condition nécessaire et suffisante pour qu'une fonction soit un noyau, à savoir :

Condition de Mercer La fonction $K(\mathbf{x}, \mathbf{z}) : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ est un noyau SSI :

$$G = K(\mathbf{x}_i, \mathbf{x}_j)_{i,j=1}^n \tag{A.2}$$

est définie positive.

Annexe B

Résultats supplémentaires

B.1 Influence de la taille d'apprentissage

Tous les classifieurs ont été testés sur 1000 nouvelles images comprenant autant d'exemples positifs que d'exemples négatifs. Pour chaque figure, le descripteur et le classifieur sont les mêmes, seule change la taille de l'ensemble d'apprentissage. Plus l'ensemble d'apprentissage est important est meilleur est le classifieur. Toutefois, dans certains cas, un phénomène de sur-apprentissage apparaît : le classifieur fournit de moins bons résultats alors qu'il a traité plus d'exemples en apprentissage.

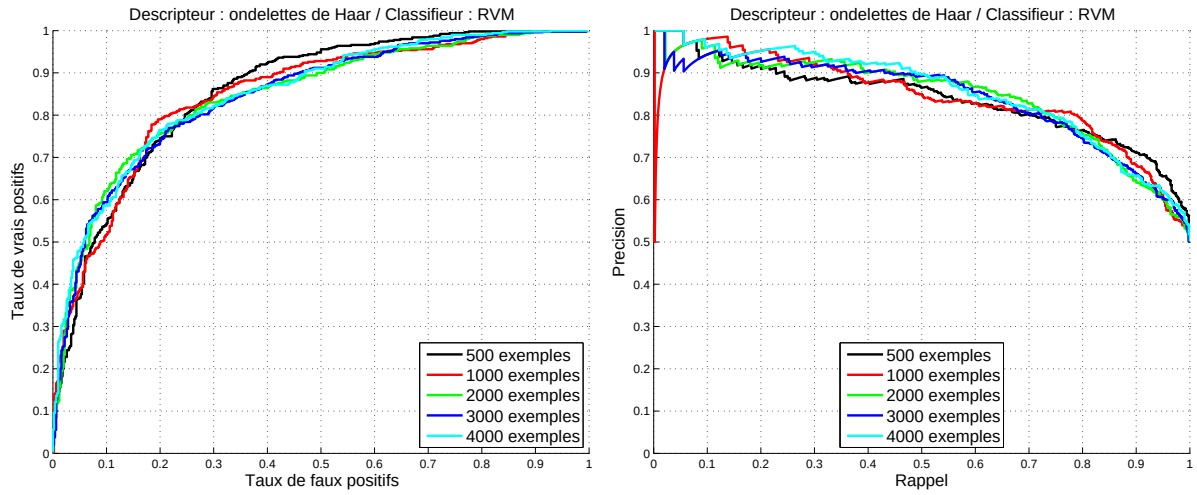


FIGURE B.1 – Influence de la taille de l'ensemble d'apprentissage : descripteur par ondelettes de Haar et apprentissage par RVM

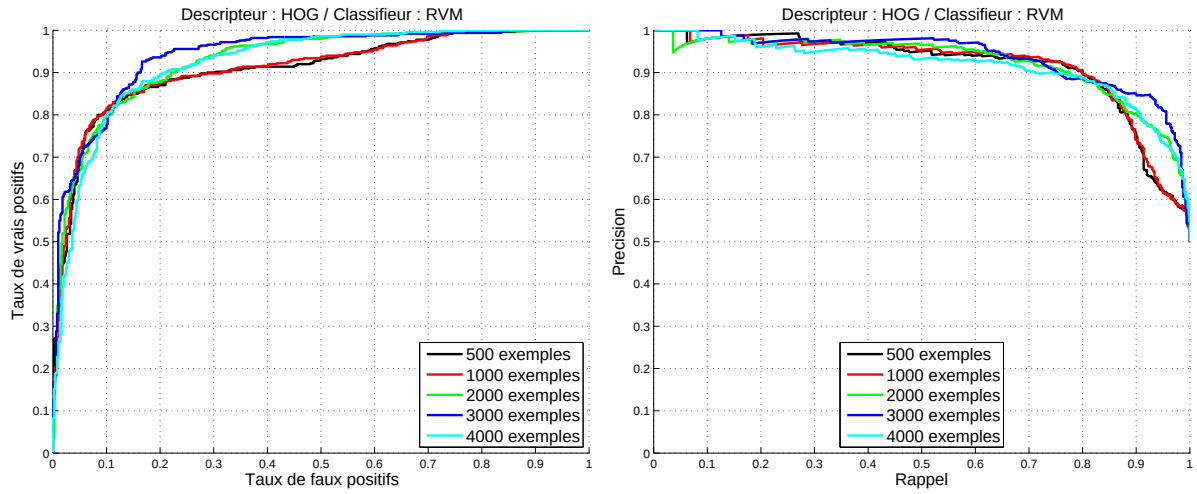


FIGURE B.2 – Influence de la taille de l'ensemble d'apprentissage : descripteur par histogrammes de gradients orientés et apprentissage par RVM

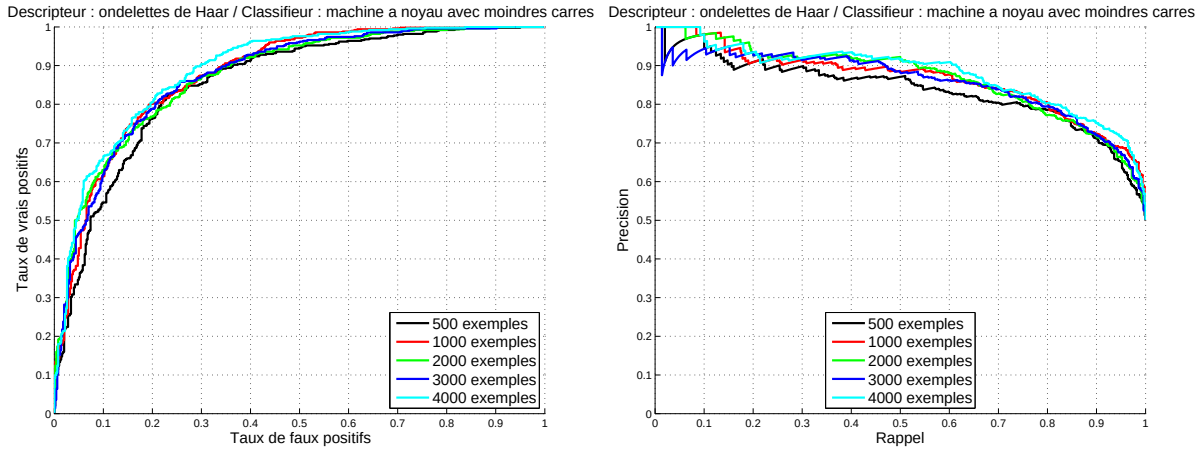


FIGURE B.3 – Influence de la taille de l'ensemble d'apprentissage : descripteur par ondelettes de Haar et apprentissage par moindres carrés

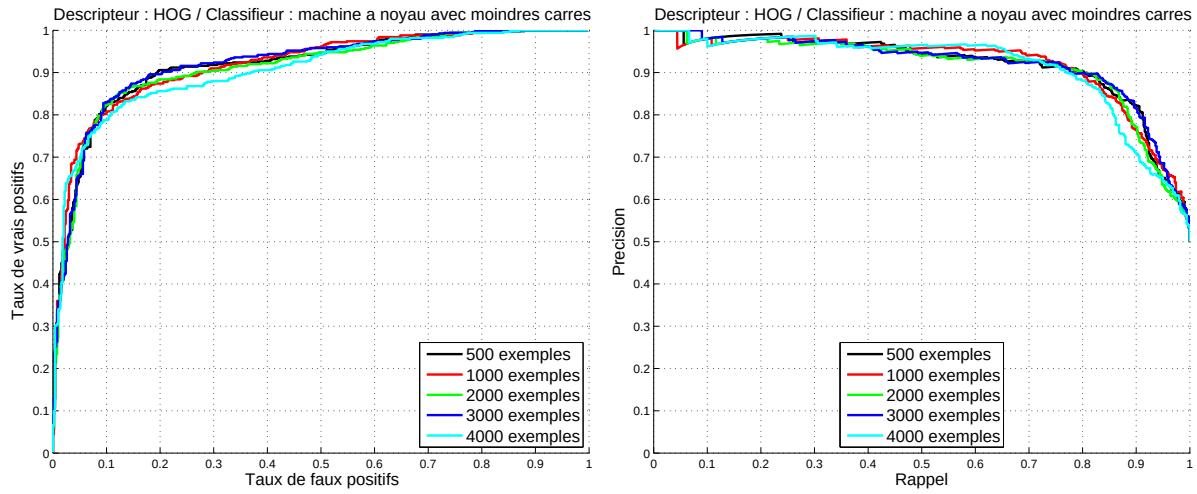


FIGURE B.4 – Influence de la taille de l'ensemble d'apprentissage : descripteur par histogrammes de gradients orientés et apprentissage par moindres carrés

B.2 Performances des classifieurs

Ici, les classifieurs ont été entraînés sur la même base d'apprentissage comprenant 2000 images, soit décrites par ondelettes de Haar (voir figure B.5), soit par histogrammes de gradients orientés (voir figure B.6). Les résultats montrent que l'AdaBoost est moins performant que les méthodes à noyaux, qui sont quant à elles difficiles à départager.

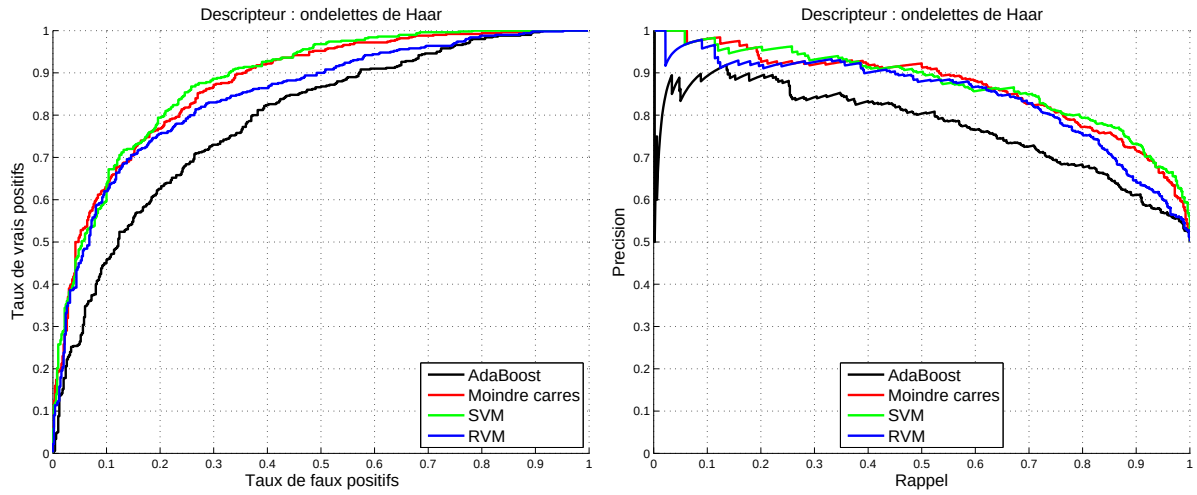


FIGURE B.5 – Comparaisons des classifieurs : descripteur par ondelettes de Haar et apprentissage sur 2000 exemples

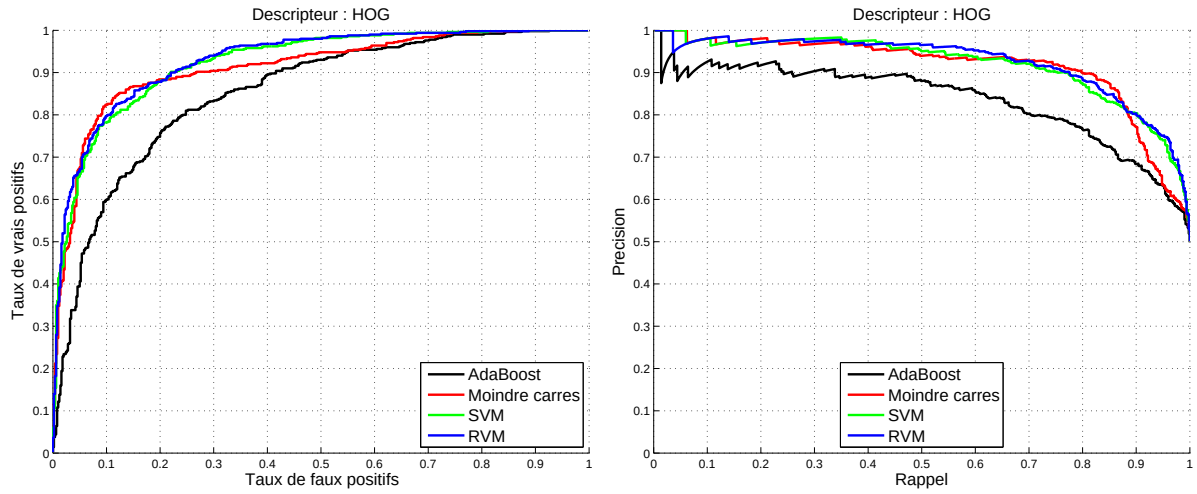


FIGURE B.6 – Comparaisons des classifieurs : descripteur par histogrammes de gradients orientés et apprentissage sur 2000 exemples

B.3 AdaBoost pour la sélection de variables et de classifieurs faibles

Voici les résultats de la sélection de variables par AdaBoost, avec un apprentissage à noyau de type SVM (voir figures B.7 et B.8) et de type moindres carrés (voir figures B.9, B.10 et B.11). Cette méthode permet d'améliorer les résultats par rapport à l'AdaBoost seul, mais ne parvient pas tout à fait à égaler les performances du classifieur à noyau entraîné sur une base d'apprentissage initiale sans sélection de variables. Ce constat se fait pour les histogrammes de gradients (figures B.7 et B.9) et les ondelettes de Haar (figures B.8 et B.10). Le nombre de descriptions disponibles au départ est déjà peu important et la sélection de variables a été trop importante pour que le classifieur à noyau réussisse à créer un aussi bon modèle qu'avec toutes les descriptions. Toutefois, au regard du faible nombre de descriptions conservées, les résultats sont satisfaisants.
De plus, cette méthode pour le descripteur binaire s'avère plus performante qu'à partir de l'ensemble d'apprentissage initial. En se plaçant dans le cadre de la méthode proposée dans la section 4.7, cela permet ainsi de supprimer un nombre conséquent de descriptions inutiles, tout en améliorant les performances du classifieur à noyau qui n'a plus de données superflues à traiter (voir figure B.11).

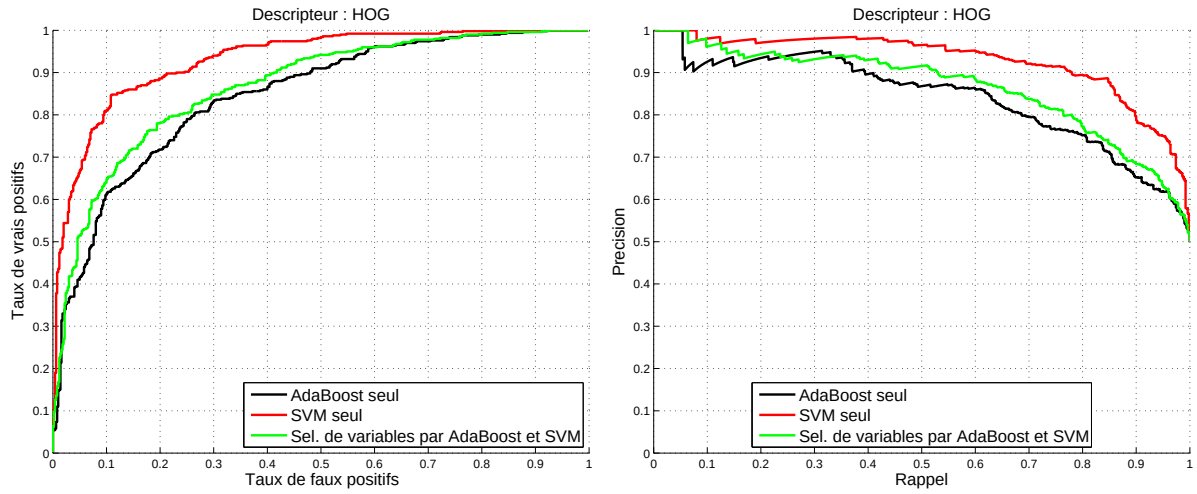


FIGURE B.7 – AdaBoost pour la sélection de variables : descripteur par histogrammes de gradients orientés et apprentissage sur 1000 exemples
L'association AdaBoost (pour la sélection de variables) et SVM permet d'améliorer les résultats par rapport à un AdaBoost seul. Cette association ne parvient toutefois pas à égaler les performances du classifieur SVM seul mais les performances sont tout de même satisfaisantes au regard du nombre restreint de variables qui ont été utilisées (313 sur 576).

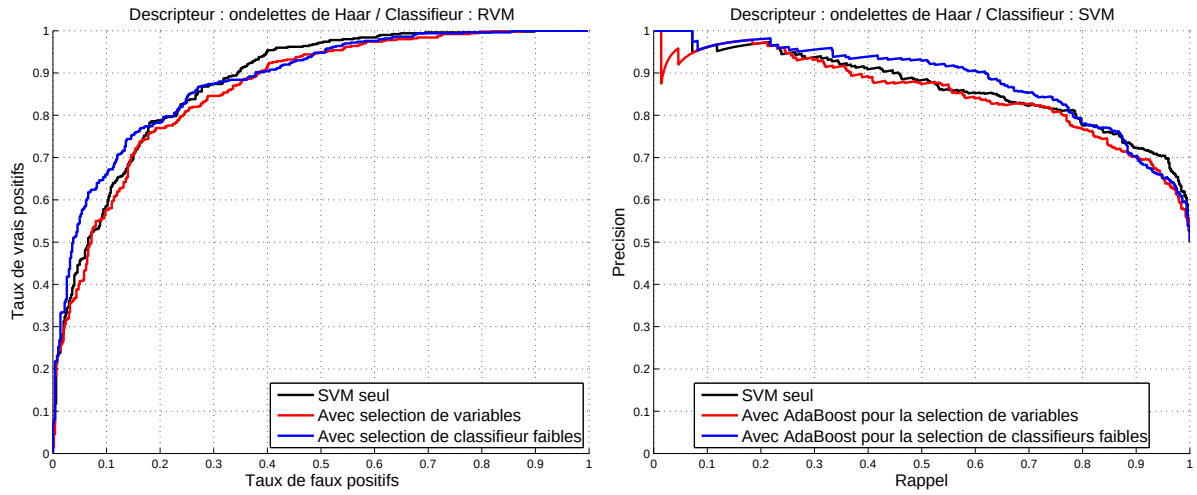


FIGURE B.8 – AdaBoost pour la sélection de variables : descripteur par ondelettes de Haar et apprentissage sur 1000 exemples
L'association AdaBoost (pour la sélection de variables) et SVM permet d'améliorer les résultats par rapport à un AdaBoost seul. Ici, pour le descripteur par ondelettes de Haar, les performances de cette association (avec un nombre restreint de variables (947 sur 2442) sont quasiment équivalentes à celles du classifieur SVM entraîné sur l'ensemble d'apprentissage sans sélection de variables.

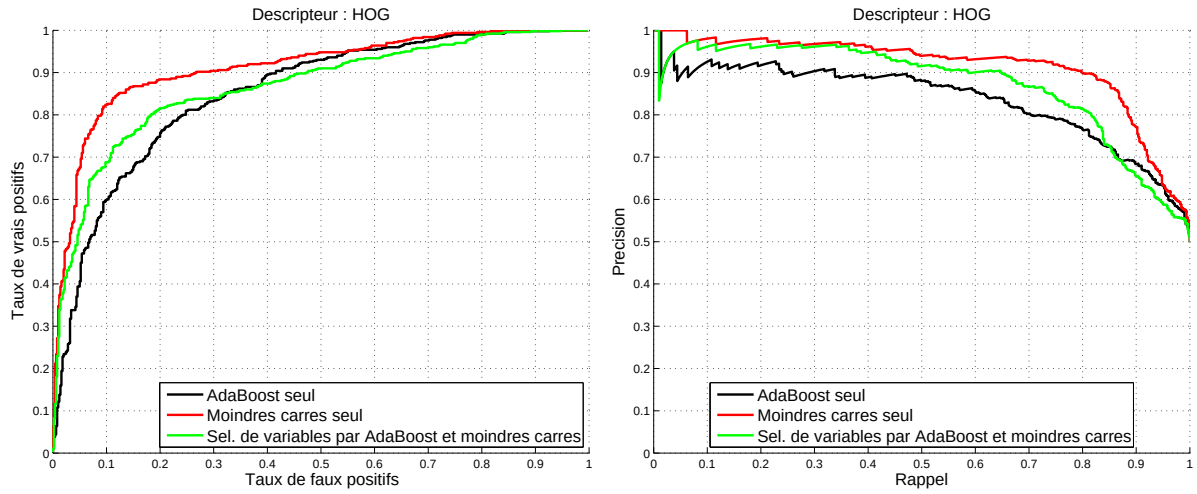


FIGURE B.9 – AdaBoost pour la sélection de variables : descripteur par histogrammes de gradients orientés et apprentissage sur 2000 exemples

L'AdaBoost a permis de sélectionner 152 *variables sur* 576.

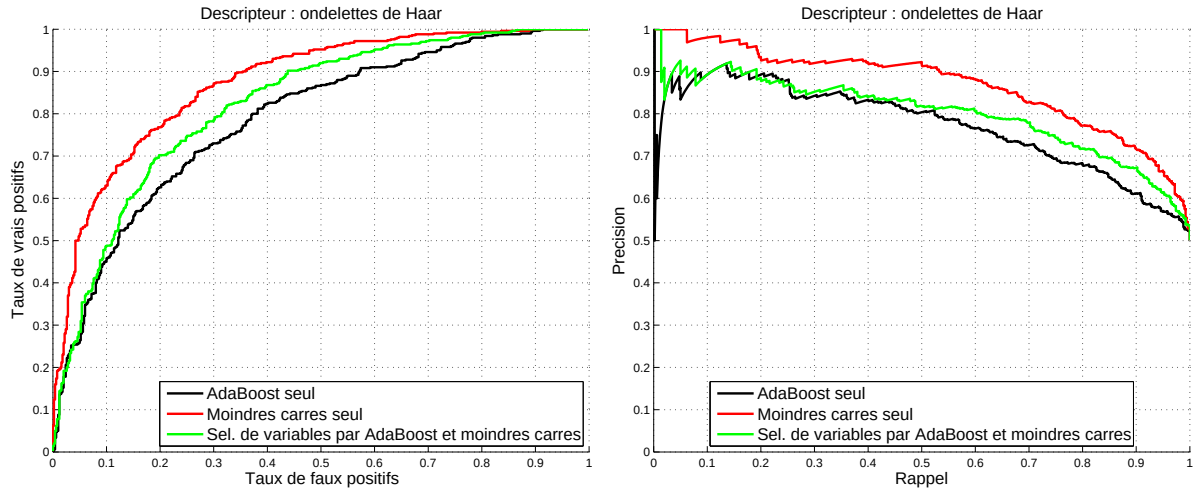


FIGURE B.10 – AdaBoost pour la sélection de variables : descripteur par ondelettes de Haar et apprentissage sur 2000 exemples

Sélection de 863 *variables sur* 2442 *par AdaBoost.*

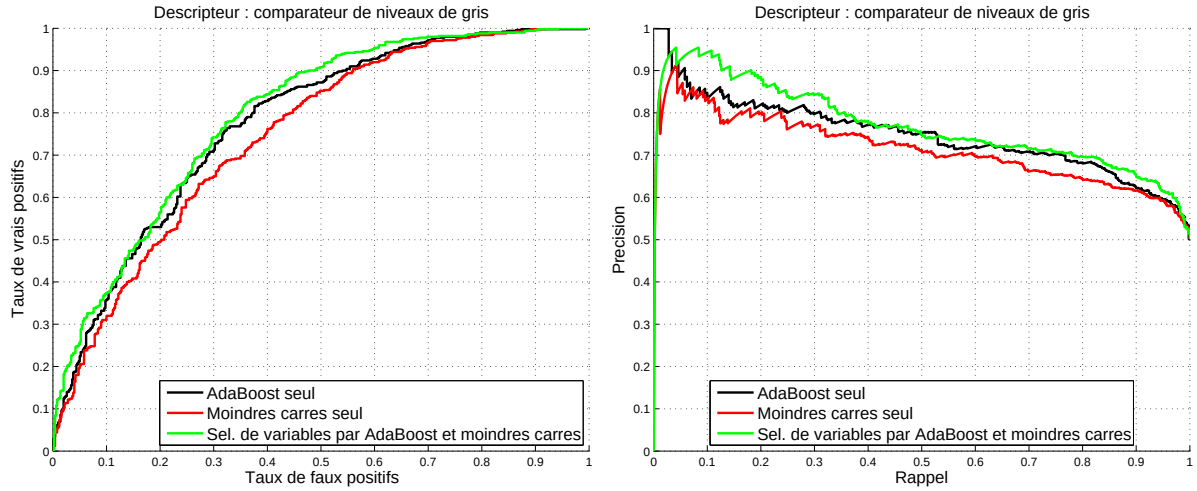


FIGURE B.11 – AdaBoost pour la sélection de variables : descripteur par comparaisons de niveaux de gris et apprentissage sur 2000 exemples
La sélection de variables par AdaBoost permet d'améliorer les résultats par rapport à l'AdaBoost seul mais aussi par rapport à l'apprentissage du classifieur à noyau sur toutes les données. Le nombre de caractéristiques conservées est de 495 *sur* 16848.

Annexe C

Séquences vidéo

Les zones de recherche des piétons sont en bleu cyan. Les objets identifiés comme des piétons sont encadrés. Le code couleur est le suivant :

- $0 \leqslant score < 25$: cadre blanc ;
- $25 \leqslant score < 50$: cadre jaune ;
- $50 \leqslant score < 75$: cadre vert ;
- $75 \leqslant score < 90$: cadre cyan ;
- $90 \leqslant score < 100$: cadre bleu.

C.1 Séquence 1

FIGURE C.1 – Séquence 1 - Planche 1

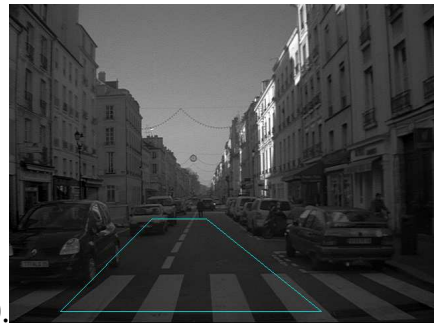
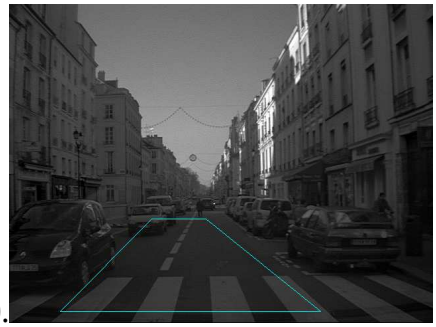
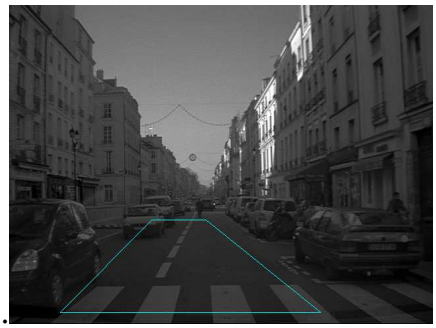
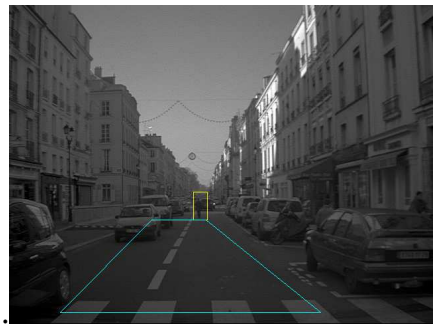
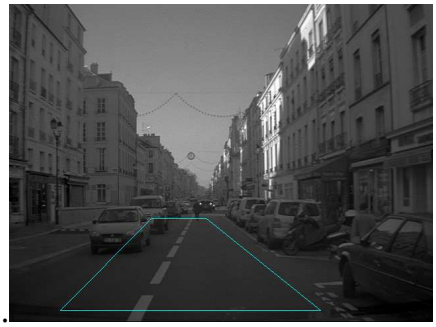
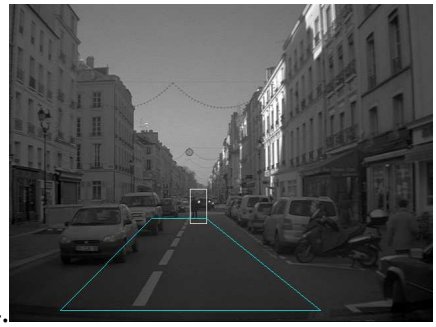
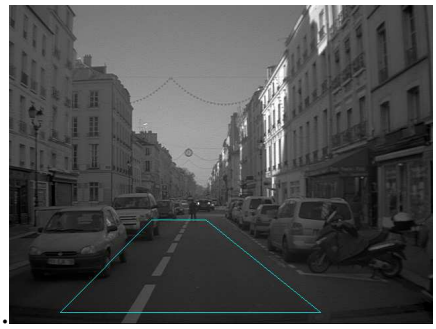
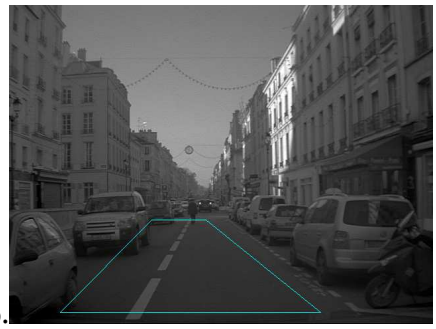
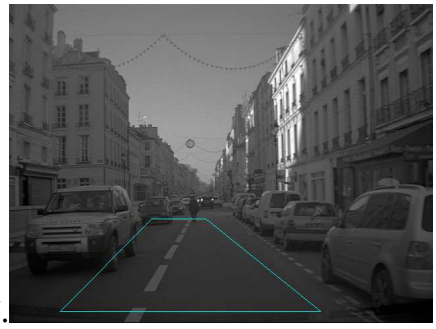
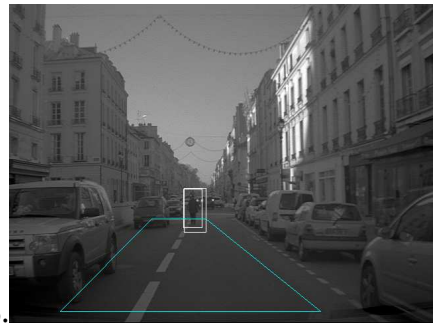
9. 
10. 
11. 
12. 
13. 
14. 
15. 
16. 
17. 
18. 

FIGURE C.2 – Séquence 1 - Planche 2

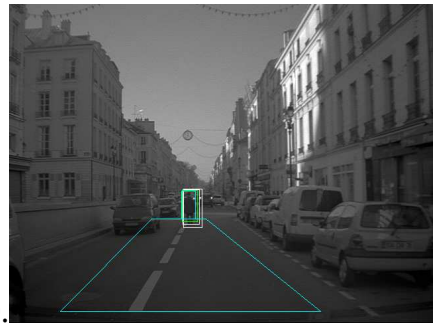
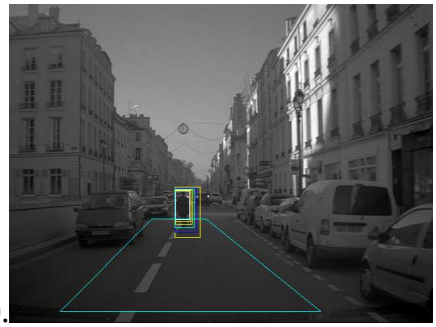
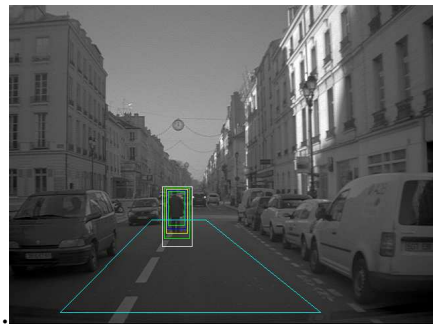
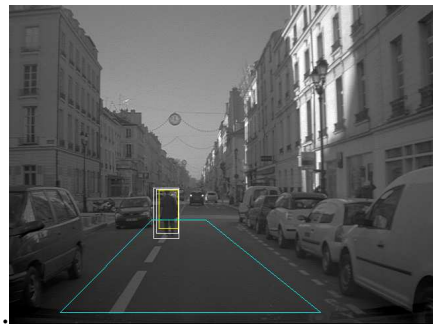
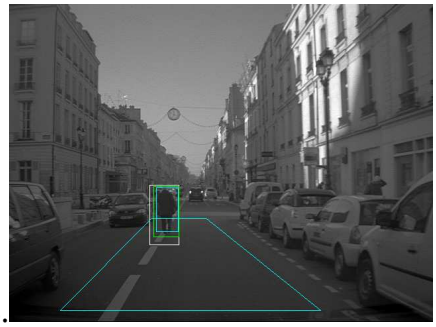
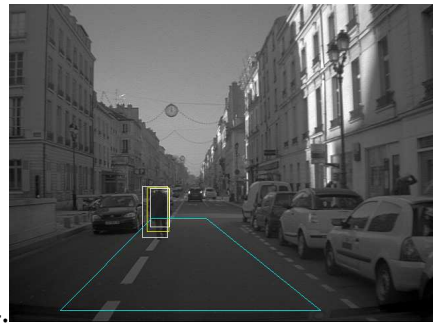
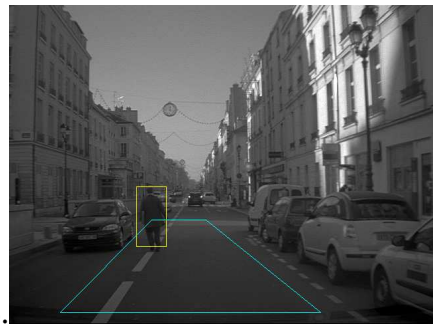
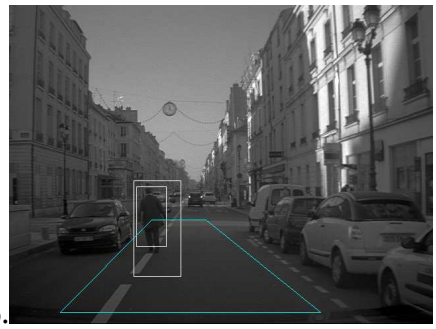
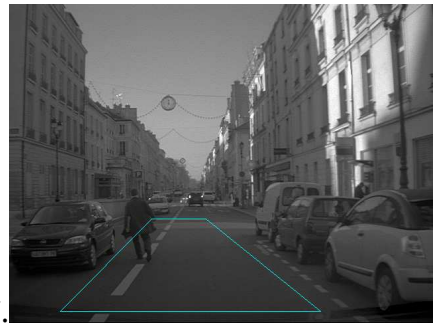
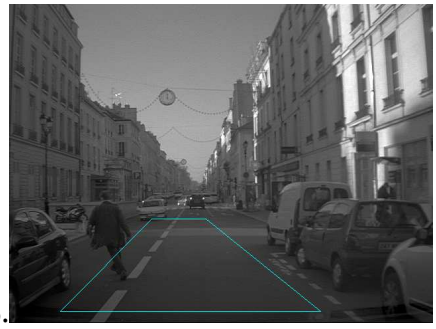
19. 
20. 
21. 
22. 
23. 
24. 
25. 
26. 
27. 
28. 

FIGURE C.3 – Séquence 1 - Planche 3

C.2 Séquence 2

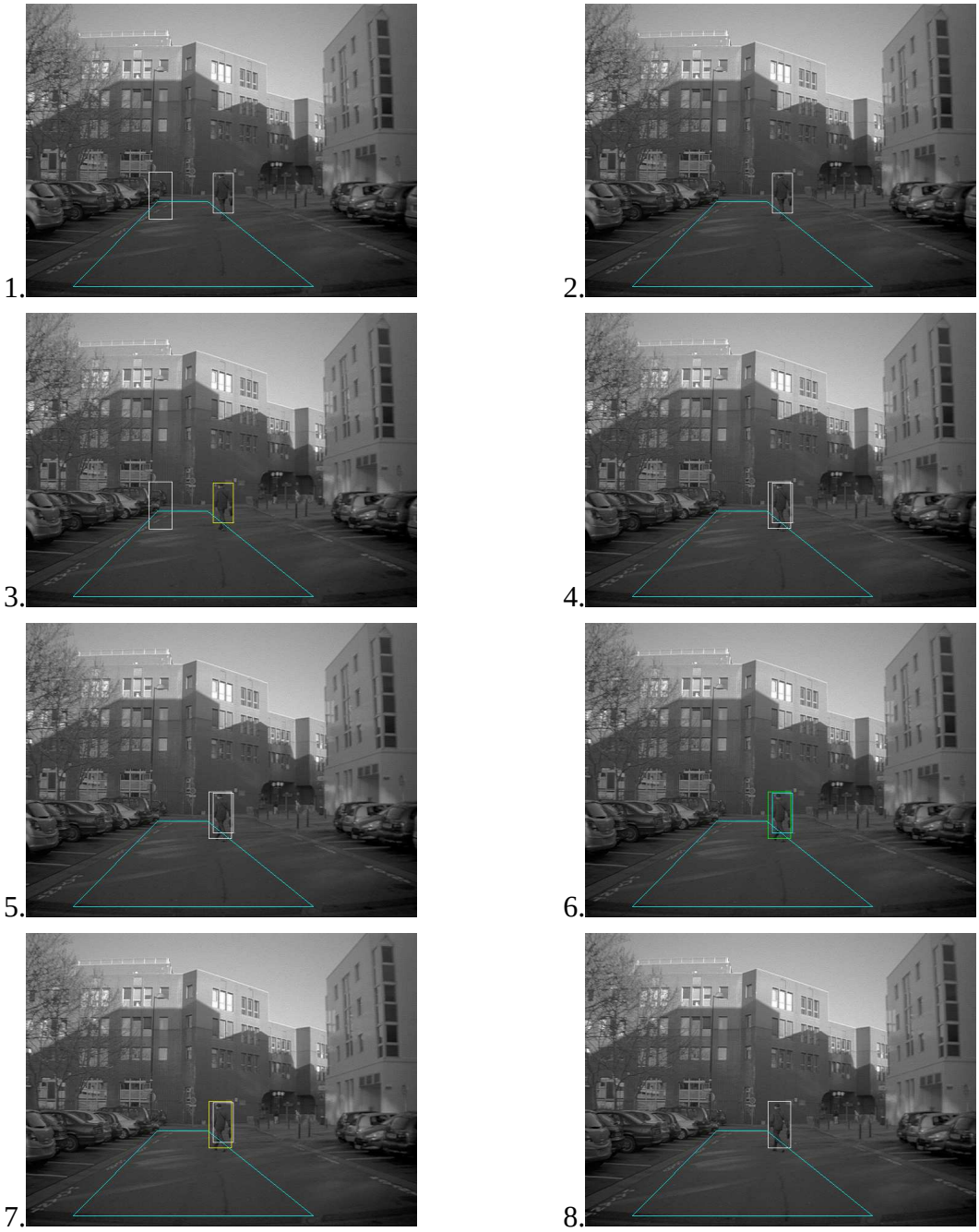


FIGURE C.4 – Séquence 2 - Planche 1

9. 
10. 
11. 
12. 
13. 
14. 
15. 
16. 
17. 
18. 

FIGURE C.5 – Séquence 2 - Planche 2

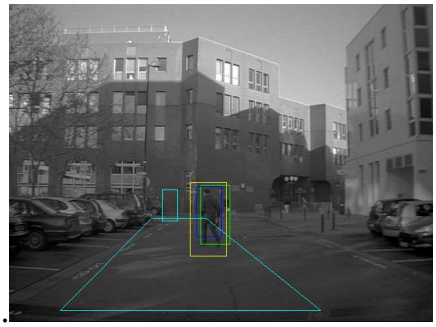
19. 
20. 
21. 
22. 
23. 
24. 
25. 
26. 
27. 
28. 

FIGURE C.6 – Séquence 2 - Planche 3

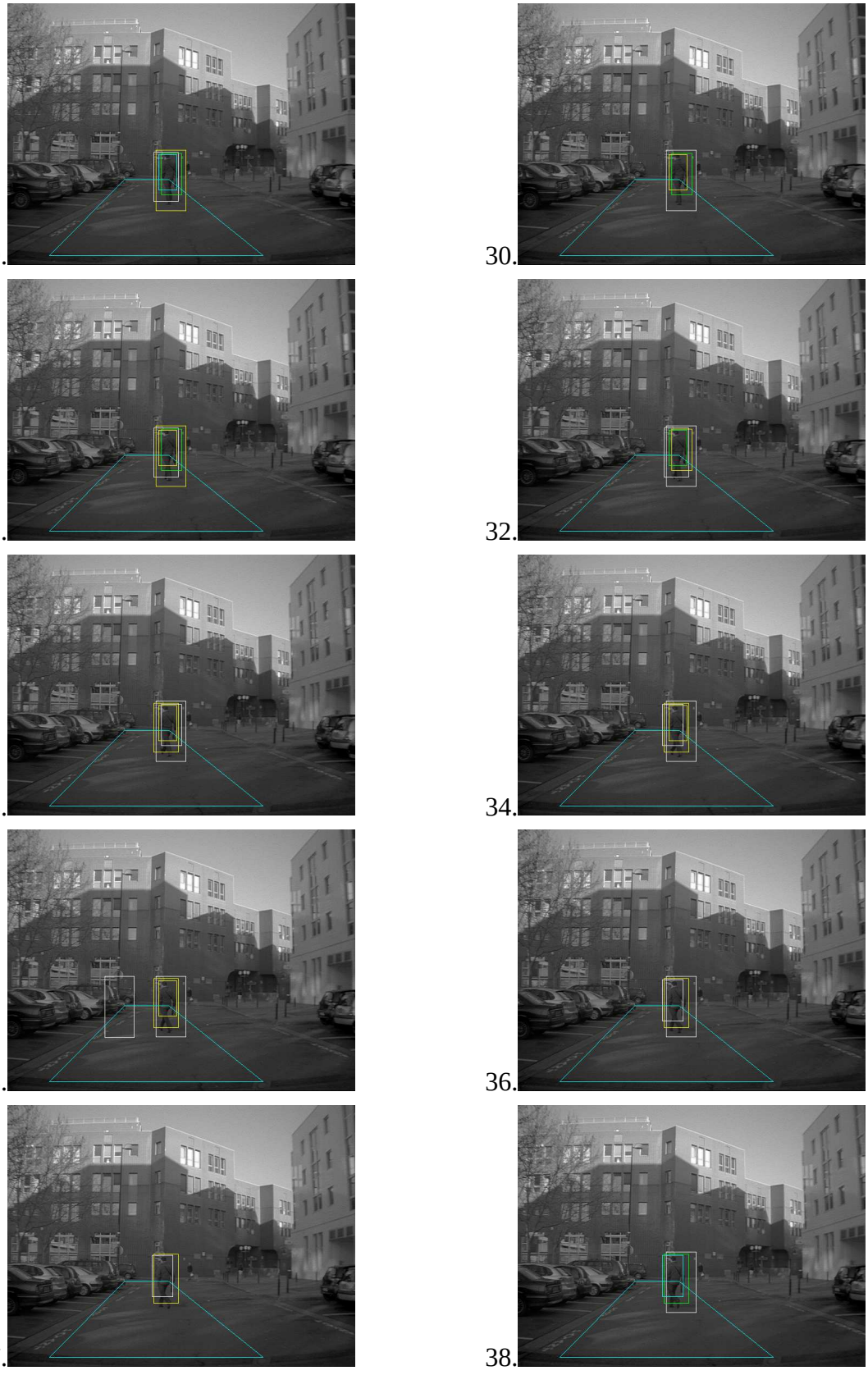


FIGURE C.7 – Séquence 2 - Planche 4

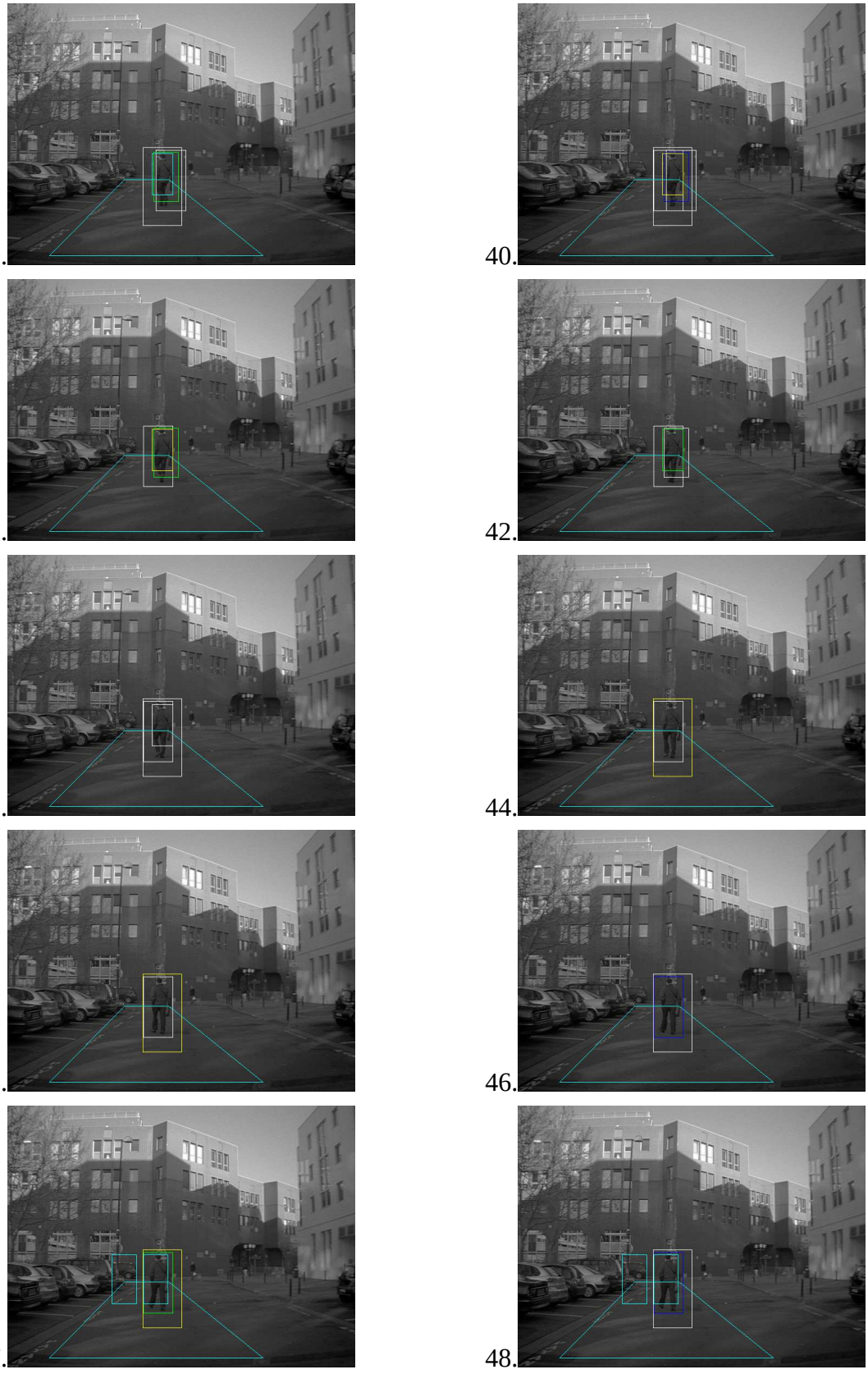


FIGURE C.8 – Séquence 2 - Planche 5

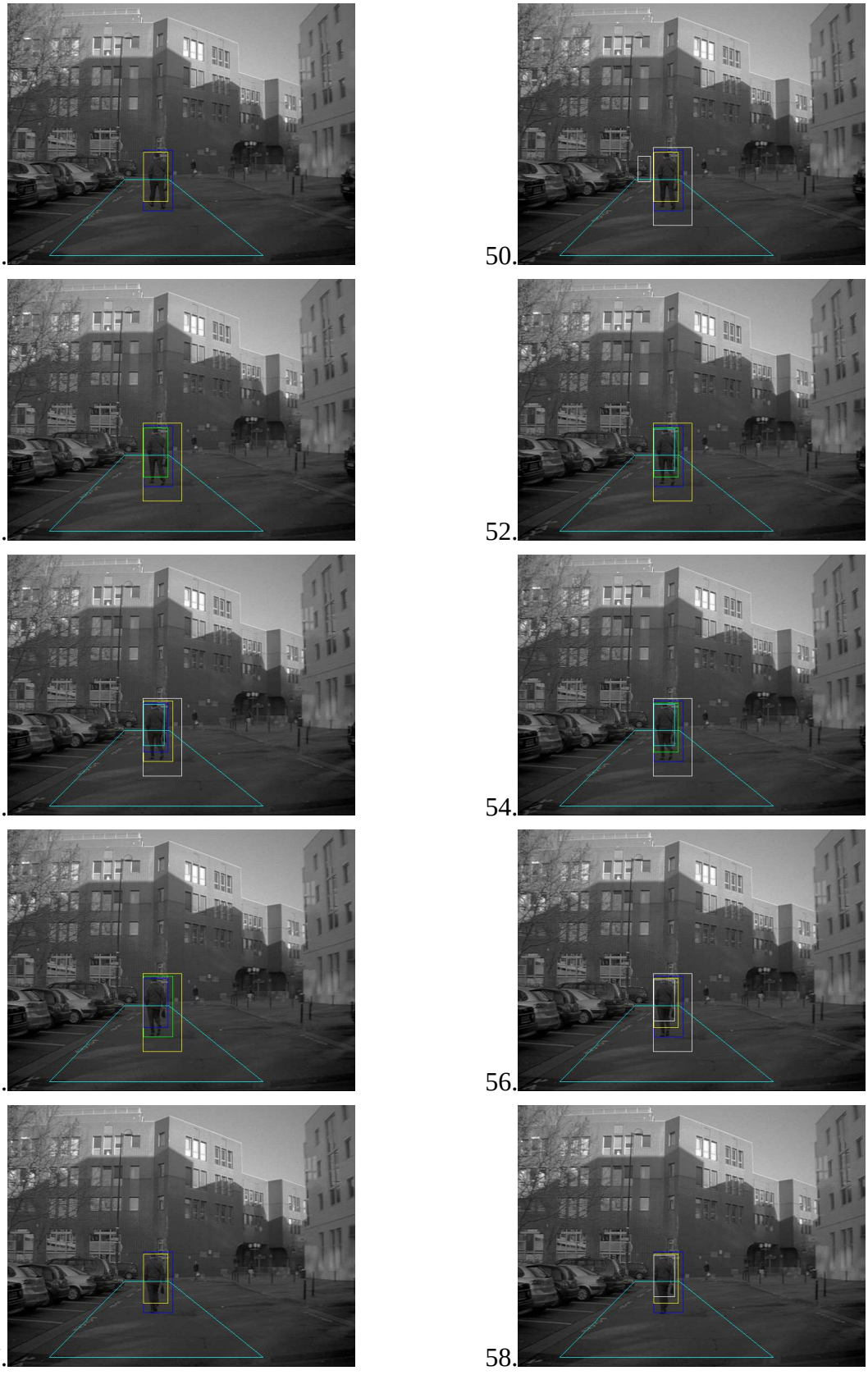


FIGURE C.9 – Séquence 2 - Planche 6

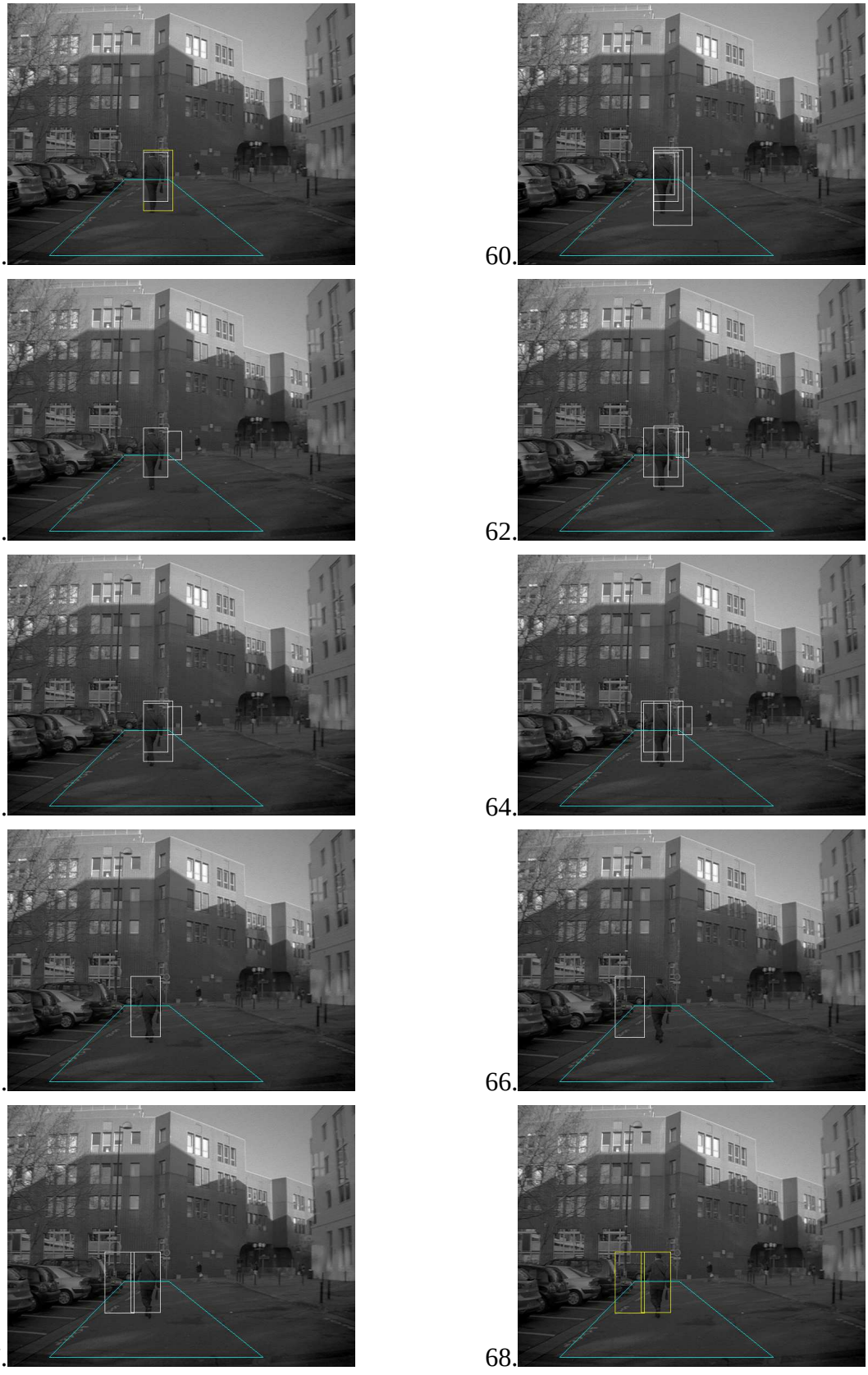


FIGURE C.10 – Séquence 2 - Planche 7

69. 
70. 
71. 
72. 
73. 
74. 
75. 
76. 
77. 
78. 

FIGURE C.11 – Séquence 2 - Planche 8

FIGURE C.12 – Séquence 2 - Planche 9

89. 
90. 
91. 
92. 
93. 
94. 
95. 
96. 
97. 

FIGURE C.13 – Séquence 2 - Planche 10

C.3 Séquence 3

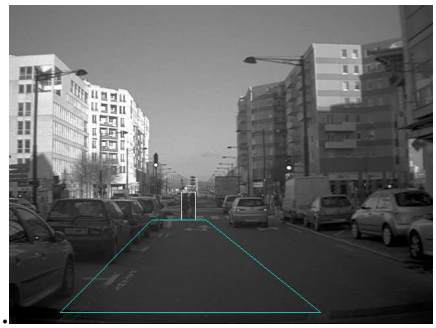
1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 

FIGURE C.14 – Séquence 3 - Planche 1

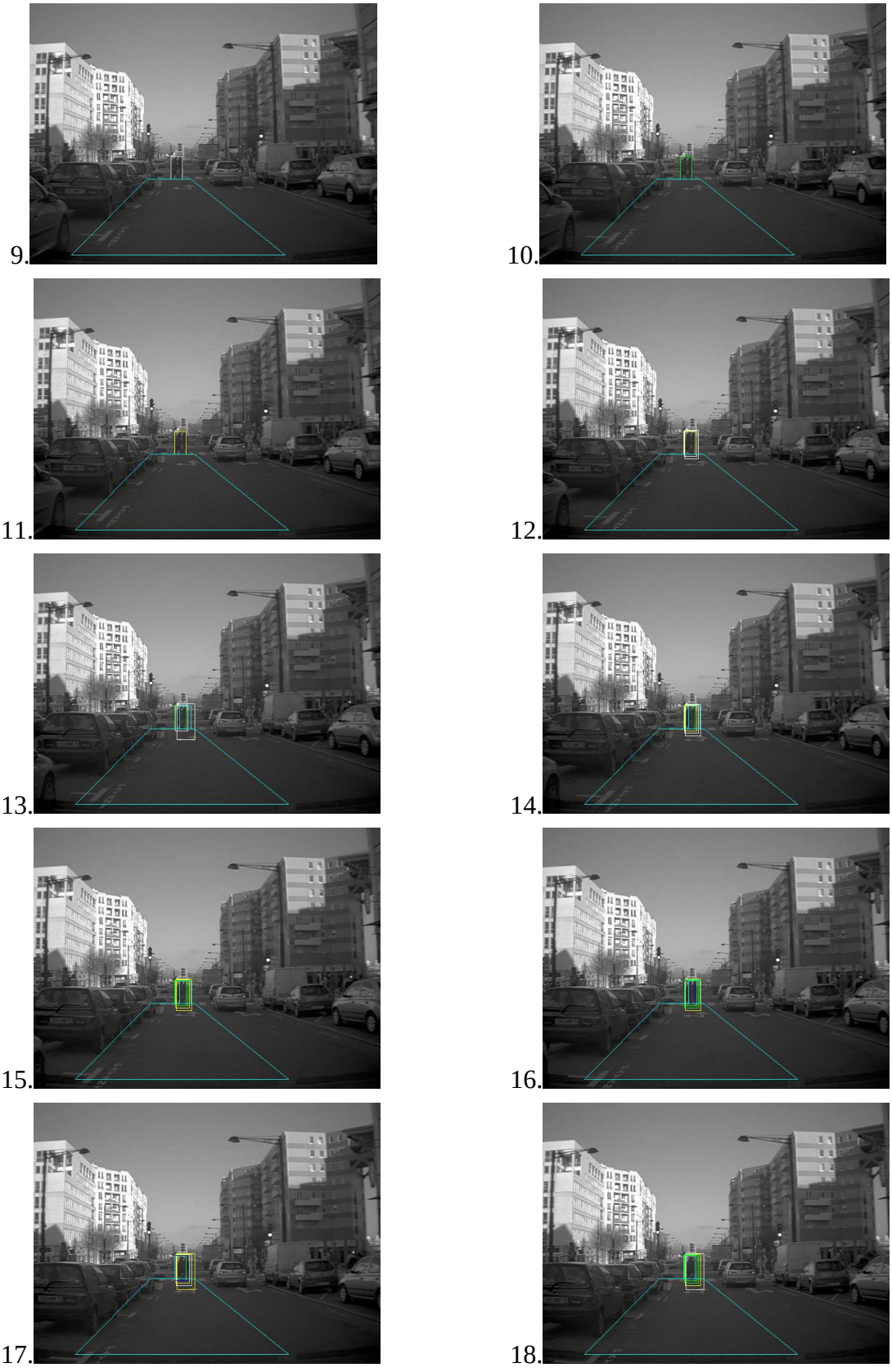


FIGURE C.15 – Séquence 3 - Planche 2

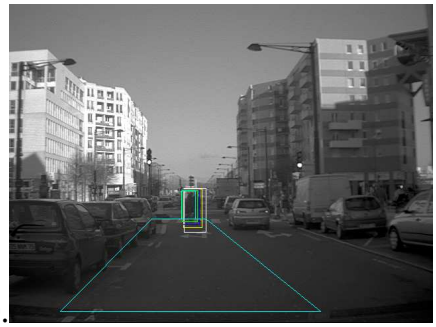
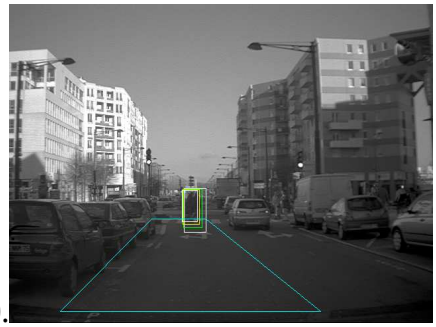
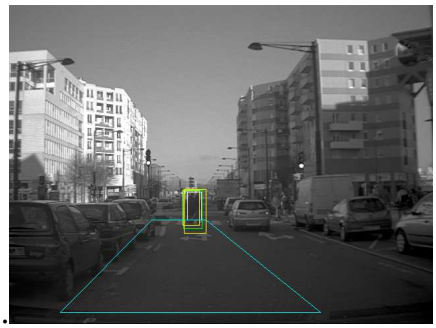
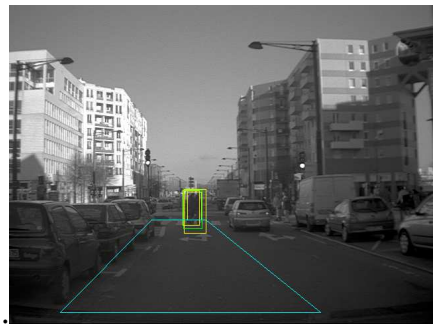
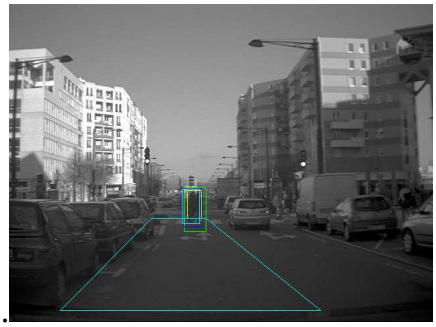
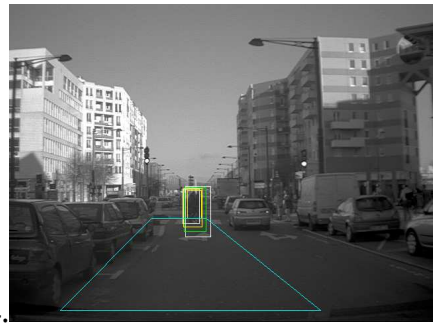
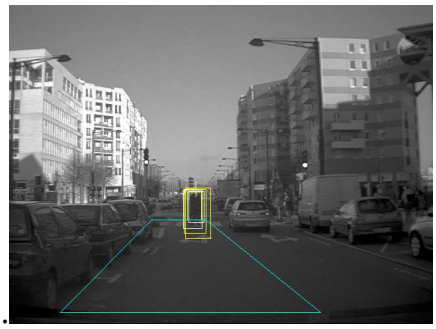
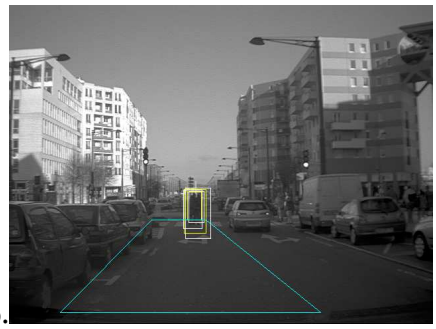
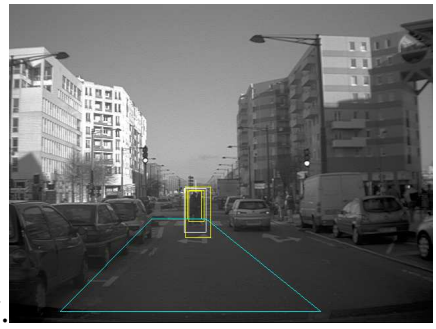
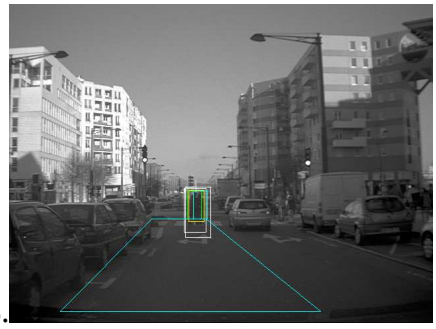
19. 
20. 
21. 
22. 
23. 
24. 
25. 
26. 
27. 
28. 

FIGURE C.16 – Séquence 3 - Planche 3

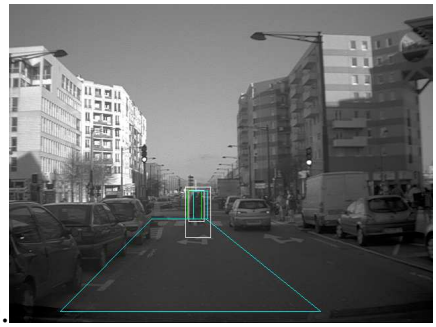
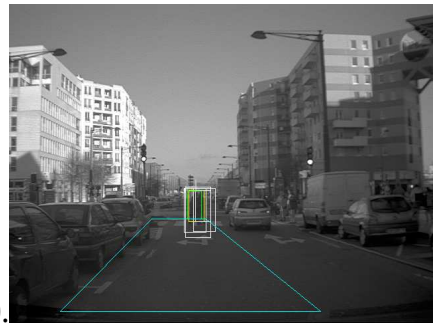
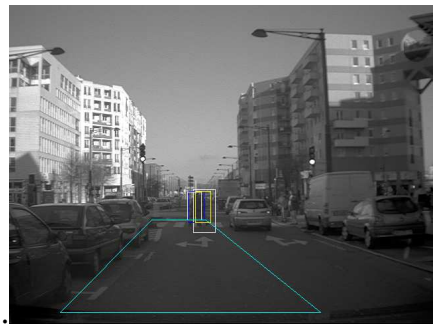
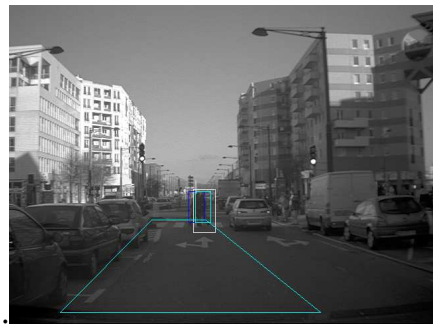
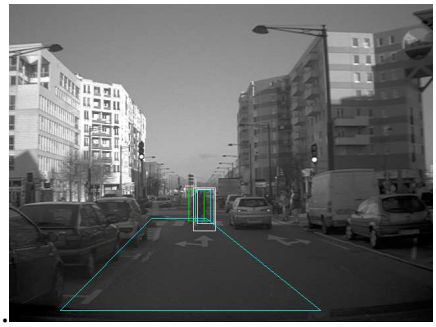
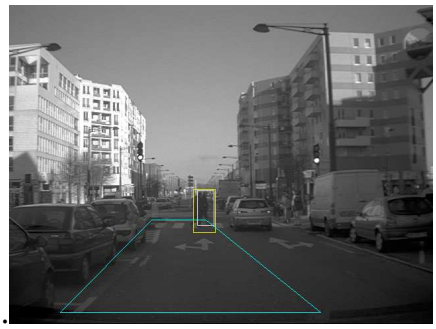
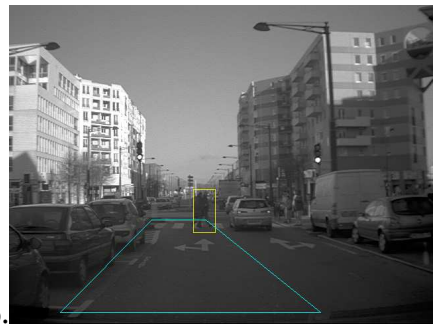
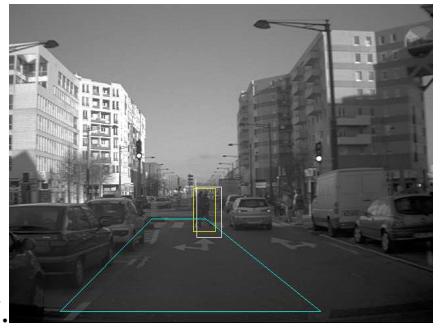
29. 
30. 
31. 
32. 
33. 
34. 
35. 
36. 
37. 

FIGURE C.17 – Séquence 3 - Planche 4

Publications dans le cadre de cette thèse

[Leyrit 08a] Laetitia Leyrit, Thierry Chateau, Christophe Tournayre & Jean-Thierry Lapresté. *Association of AdaBoost and Kernel Based Machine Learning Methods for Visual Pedestrian Recognition*. In IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, 4 June 2008.

[Leyrit 08b] Laetitia Leyrit, Thierry Chateau, Christophe Tournayre & Jean-Thierry Lapresté. *Visual Pedestrian Recognition in Weak Classifier Space Using Nonlinear Parametric Models*. In IEEE International Conference on Image Processing, San Diego, USA, 12 October 2008.

[Leyrit 09a] Laetitia Leyrit, Thierry Chateau & Jean-Thierry Lapresté. *Descripteurs pour la reconnaissance de piétons*. In ORASIS Congrès des jeunes chercheurs en vision par ordinateur, Trégastel, France, 8 June 2009.

[Leyrit 09b] Laetitia Leyrit, Thierry Chateau, Christophe Tournayre & Jean-Thierry Lapresté. *Association de classifieurs pour la reconnaissance de piétons dans les images*. In COmpression et Representation des Signaux Audiovisuels, pages 215–221, Toulouse, France, 18 March 2009.

[Leyrit 10] Laetitia Leyrit, Thierry Chateau & Jean-Thierry Lapresté. New advances in machine learning, chapitre Classifiers association for high dimensional problem : application to pedestrian recognition, pages 93–104. In-Tech, February 2010.

Bibliographie

[1] Yotam Abramson, Bruno Steux, and Hicham Ghorayeb. Yef (yet even faster) real-time object detection. In *International Workshop on Automatic Learning and Real-Time*, pages 5–13, Siegen, Germany, 7 September 2005.

[2] Ankur Agarwal and Bill Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Washington, U.S.A., 27 June 2004.

[3] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face description with local binary patterns : application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12) :2037–2041, December 2006.

[4] Yali Amit, Donald Geman, and Kenneth Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11) :1300–1305, November 1997.

[5] Luisa Andreone, Andrea Guarise, Francesco Lilli, Dariu M. Gavrila, and Marco Pieve. Cooperative systems for vulnerable road users : the concept of the WATCH-OVER project. In *Intelligent Transport Systems and Services World Congress*, Brussels, Belgium, 8 October 2006.

[6] Julien Bégard, Nicolas Allezard, and Patrick Sayd. Real-time humans detection in urban scenes : local descriptors and classfiers selection with AdaBoost-like algorithms. In *Workshop on Object Tracking and Classification in and Beyond the Visible Spectrum in IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, U.S.A., 24 June 2008.

[7] Massimo Bertozzi, Alberto Broggi, Mike Del Rose, Mirko Felisa, Alain Rakotomamonjy, and Frédéric Suard. A pedestrian detector using histograms of oriented gradients and a support vector machine classifier. In *IEEE Conference on Intelligent Transportation Systems*, pages 144–148, Seattle, WA, U.S.A., 30 September 2007.

[8] Serge Beucher and Christian Lantuéjoul. Use of watersheds in contour detection. In *International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, September 1979.

[9] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 1-2(97) :245–271, December 1997.

[10] Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, and Massimiliano Sechi. Shape-based pedestrian detection. In *IEEE Intelligent Vehicles Symposium*, pages 215–220,, Detroit, U.S.A., 3 October 2000.

[11] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, June 1998.

[12] Marine Campedel and Eric Moulines. Classification et sélection de caractéristiques de textures. *Revue d'Intelligence Artificielle*, 19 :633–659, 4 May 2005.

[13] Marine Campedel, Eric Moulines, Henri Matre, and Mihai Dactu. Feature selection for satellite image indexing. In *Image Information Mining - Theory and Application to Earth Observation*, 5 October 2005.

[14] John Canny. A computational approach to edge detection. *IEEE Transanctions on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, November 1986.

[15] Hong Cheng, Nanning Zheng, and Chong Sun. Boosted Gabor features applied to vehicle detection. In *International Conference on Pattern Recognition*, 20 August 2006.

[16] Nello Christianni and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[17] Antoine Cornuéjols and Laurent Miclet. *Apprentissage artificiel, concepts et algorithmes*. Editions Eyrolles, August 2002.

[18] Franklin C. Crow. Summed-Area Tables for Texture Mapping. *Computer Graphics*, 18(3) :207–212, July 1984.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, San Diego, U.S.A., 20 June 2005.

[20] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision*, Graz, Austria, 8 May 2006.

[21] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2001.

[22] Markus Enzweiler and M. Dariu Gavrila. Monocular pedestrian detection : survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12) :2179–2195, December 2009.

[23] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, U.S.A., 24 June 2008.

[24] Joël Falcou and Jocelyn Serot. Application of template-based metaprogramming compilation techniques to the efficient implementation of image processing algorithms on SIMD-capable processors. In *Advanced Concepts for Intelligent Vision Systems*, Brussels, Belgium, 31 August 2004.

[25] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5 :1531–1555, November 2004.

[26] Yoav Freund. An Adaptive Version of the Boost by Majority Algorithm. *Machine Learning*, 43(3) :293–318, June 2001.

[27] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 148–156, Bari, Italia, 3 June 1996.

[28] Dariu M. Gavrila. A Bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8) :1–13, August 2007.

[29] Dariu M. Gavrila, Martin Kunert, and Ulrich Lages. A multi-sensor approach for the protection of vulnerable traffic participants - the PROTECTOR project. In *IEEE Instrumentation and Measurement Technology Conference*, 21 May 2001.

[30] Dariu M. Gavrila and Stephen Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1) :41–59, June 2007.

[31] Philip Geismann and Georg Schneider. A two-staged approach to vision-based pedestrian recognition using Haar and HOG features. In *IEEE Intelligent Vehicles Symposium*, Eindoven, The Netherlands, 4 June 2008.

[32] David Gerónimo, Antonio López, Daniel Ponsa, and Angel Domingo Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, Girona, Spain, 6 June 2007.

[33] David Gerónimo, Antonio López, Angel Domingo Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7) :1239–1258, July 2010.

[34] Samuel Gidel. *Méthodes de détection et de suivi multi-piétons multi-capteurs embarquées sur un véhicule routier : application à un environnement urbain*. PhD thesis, Ecole Doctorale des Sciences Pour l'Ingénieur - Université Blaise Pascal, Clermont-Ferrand, 29 April 2010.

[35] Samuel Gidel, Paul Checchin, Christophe Blanc, Thierry Chateau, and Laurent Trassoudaine. Pedestrian Detection Method using a Multilayer Laserscanner : Application in Urban Environment. In *IEEE Conference on Intelligent Robots and Systems*, pages 173–178, Nice, France, 22 September 2008.

[36] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *British Machine Vision Conference*, Edinburgh, United Kingdom, 4 September 2006.

[37] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3 :1157–1182, March 2003.

[38] Mark Andrew Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *International Conference on Machine Learning*, pages 359–366, San Fransisco, U.S.A., 29 June 2000.

[39] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, Manchester - United Kingdom, 31 August 1988.

[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Elements of Statistical Learning : Data Mining, Inference and Prediction*. Springer-Verlag, New York, 2001.

[41] Marko Heikkilä, Matti Pietikainen, and Cornelia Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3) :425–436, March 2009.

[42] Bernd Heisele, Thomas Serre, Sayan Mukherjee, and Tomaso Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 18–24, Kauai, Hawaii, 9 December 2001.

[43] Anne Herbin and Sebastien Cornou. Résumé des contraintes des cahiers de charges. Technical report, Projet ANR LOVe, 30 January 2007.

[44] Bin Hu, Shengjin Wang, and Xiaoqing Ding. Multi features combination for pedestrian dectection. *Journal of multimedia*, 5(1) :79–84, February 2010.

[45] Anil Jain and Douglas Zongker. Feature selection : evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2) :153–158, February 1997.

[46] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[47] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition : a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :4–37, January 2000.

[48] Jan Sochman. *Learning for Sequential Classification*. PhD thesis, Czech Technical University, Prague, Czech Republic, February 2009.

[49] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes : active contour models. In *International Conference on Computer Vision*, June 1987.

[50] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *International Workshop on Machine Learning*, pages 249–256, Aberdeen, Scotland, United Kingdom, 1 July 1992. Morgan Kaufmann Publishers Inc.

[51] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2) :273–324, December 1997.

[52] Igor Kononenko. Estimating attributes : analysis and extensions of RELIEF. In *European Conference on Machine Learning*, Catania, Italy, April 1994.

[53] Raphael Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real time obstacle detection in stereovision on non flat road geometry through 'V-disparity' representation. In *IEEE Intelligent Vehicle Symposium*, Versailles, France, 18 June 2002.

[54] Diane Larlus and Frédéric Jurie. Latent mixture vocabularies for object categorization and segmentation. *Journal of Image and Vision Computing*, 27(5) :523–534, 2009.

[55] Duy Dinh Le and Shin'ichi Satoh. Feature selection by AdaBoost for SVM-based face detection. *Forum on Information Technology*, pages 183–186, 2004.

[56] David Lefée, Stéphane Mousset, Abdelaziz Bensrhair, and Massimo Bertozzi. Cooperation of Passive Vision Systems in Detection and Tracking of Pedestrians. In *IEEE Intelligent Vehicles Symposium*, Parma, Italy, 14 June 2004.

[57] Bastian Leibe, Konrad Schindler, Cornelis Nico, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1683-1698), October 2008.

[58] Bastian Leibe, Edgar Seeman, and Bernt Schiele. Pedestrian detection in crowned scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 878–885, San Diego, U.S.A., 20 June 2005.

[59] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9) :1465–1479, September 2006.

[60] Kobi Levi and Yair Weiss. Learning object detection from a small number of examples : the importance of good features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 53–60, Washington, DC, 27 June 2004.

[61] Huan Liu and Rudy Setiono. A probabilistic approach to feature selection - a filter solution. In *International Conference on Machine Learning*, pages 319–327, Bari, Italy, 3 June 1996.

[62] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(20) :91–110, November 2004.

[63] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5) :720–736, September 1992.

[64] David J.C. MacKay. *Models of Neural Networks II*, volume 3, chapter Bayesian methods for backpropagation networks, pages 211–254. Springer, 1994.

[65] Philippe Marchal, Dariu Gavrila, Laurent Letellier, Marc-Michael Meinecke, Richard Morris, and Mathias Töns. SAVE-U : An innovative sensor platform for vulnerable road user protection. In *Intelligent Transport Systems and Services*. Madrid, Spain, 16 November 2003.

[66] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4) :349–361, April 2001.

[67] Fabien Moutarde, Bogdan Stanciulescu, and Amaury Breheret. Real-time visual detection of vehicles and pedestrians with new efficient AdaBoost features. In *Workshop on Planning, Perception and Navigation for Intelligent Vehicles of International Conference in IEEE Intelligent RObots and Systems*, Nice, FRANCE, 26 September 2008.

[68] Stefan Munder and Dariu M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11), November 2006.

[69] Pablo Negri, Xavier Clady, Shehzad Muhammad Hanif, and Lionel Prevost. A Cascade of Boosted Generative and Discriminative Classifiers for Vehicle Detection. *Eurasip Journal on Advances in Signal Processing*, 2008, 2008.

[70] Laurence Ngako Pangop, Frederic Chausse, Sébastien Cornou, and Roland Chapuis. Adaptive Bayesian Combination of Features from Laser scanner and Camera for Pedestrian Detection. In *IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France, 3 September 2007.

[71] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1) :51–59, January 1996.

[72] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7) :971–987, July 2002.

[73] Mickael Oren, Constantine Papageorgiou, Pawan Sinha, Edgar Osuna, and Tomaso Poggio. Pedestrian detection using wavelet templates. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 193–199, San Juan, Porto-Rico, 17 June 1997.

[74] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1) :15–33, June 2000.

[75] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *International Conference on Computer et Vision (ICCV)*, Bombay,India, 4 January 1998.

[76] Niklas Pettersson, Lars Petersson, and Lars Andersson. The histogram feature - a resource-efficient weak classifier. In *IEEE Intelligent Vehicle Symposium*, Eindoven, The Netherlands, 4 June 2008.

[77] Fatih Porikli. Integral Histogram : a fast way to extract histograms in cartesian cpaces. In *Conference on Computer Vision and Pattern Recognition*, San Diego, USA, 20 June 2005.

[78] Alain Rakotomamonjy. Variable selection using svm-based criteria. *Journal of Machine Learning Research*, 3 :1357–1370, March 2003.

[79] Franck Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, November 1958.

[80] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, (323) :533–536, 9 October 1986.

[81] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2) :197–227, June 1990.

[82] Sam Schauland, Anton Kummert, Su-Birm Park, Uri Iurgel, and Yan Zhang. Vision-based pedestrian detection - Improvement and verification of feature extraction methods and SVM-based classification. In *IEEE Conference of Intelligent Transportation Systems*, Toronto, Canada, 17 September 2006.

[83] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S. Davis. Human detection using partial least squares analysis. In *IEEE International Conference on Computer Vision*, Kyoto, Japan, 27 September 2009.

[84] Amon Shashua, Yoram Gdalyahu, and Gaby Hayun. Pedestrian detection for driving assistance : single-frame classification and system level performance. In *IEEE Intelligent Vehicles Symposium*, Parma, Italy, 14 June 2004.

[85] Piyanuch Silapachote, Deepak R. Karuppiah, and Allen R. Hanson. Feature selection using AdaBoost for face expression recognition. In *Visualization, Imaging, and Image Processing*, 6 September 2004.

[86] Miguel Angel Sotelo, Ignacio Parra, David Fernandez, and Eugenio Naranjo. Pedestrian detection using SVM and multi-feature combination. In *Conference of Intelligent Transportation Systems*, Toronto, Canada, 17 September 2006.

[87] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, and Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *IEEE Intelligent Vehicles Symposium*, pages 206–212, Tokyo, Japan, 13 June 2006.

[88] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection using evolutionary Gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, June 2005.

[89] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1 :211–244, 6 May 2001.

[90] Michael E. Tipping. Bayesian inference : An introduction to principles and practice in machine learning. In Ulrike von Luxburg and Gunnar Rätsch Olivier Bousquet, editor, *Advanced Lectures on Machine Learning*, pages 41–62. Springer Verlag, 2004.

[91] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region Covariance A Fast Descriptor for Detection and Classification. In *European Conference on Computer Vision*, Graz, Austria, 8 May 2006.

[92] Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian Detection via Classification on Riemannian Manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10) :1713–1727, October 2008.

[93] Vladimir Vapnik. *Statistical learning theory*. Wiley, October 1998.

[94] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Kauai, Hawaii, 9 December 2001.

[95] Paul Viola and Michael Jones. Robust real-time object detection. In *International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*, Vancouver, Canada, 13 July 2001.

[96] Paul Viola, Michael Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *International Conference on Computer Vision (ICCV)*, 13 October 2003.

[97] Christopher J.C.H. Watkins and Peter Dayan. Technical note : Q-learning. *Machine Learning*, 8(3-4) :279–292, May 1992.

[98] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for SVMs. In *Neural Information Processing Systems*, pages 668–674, Denver, U.S.A., 27 November 2000.

[99] Jian Yao and Jean-Marc Odobez. Fast human detection from videos using covariance features. In *International Workshop on Visual Surveillance in European Conference on Computer Vision*, Marseille, France, 17 October 2008.

[100] Lei Yu and Huan Liu. Feature selection for high-dimensional data : a fast correlation-based filter solution. In *International Conference on Machine Learning*, pages 856–863, Washington, U.S.A., 21 August 2003.

[101] Liang Zhao and Charles E. Thorpe. Stereo- and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3) :148–154, September 2000.

[102] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class AdaBoost. *Statistics and Its Interface*, 2 :349–360, 2009.

Glossaire

AdaBoost (ou *Adaptative Boosting*) une des premières méthodes de *Boosting* introduite par Freund et Schapire.

Apprentissage ensemble de méthodes d'intelligence artificielle visant à éduquer un système pour qu'il soit capable d'agir en conséquence de ce qu'il sait.

Boosting domaine de l'apprentissage automatique regroupant de nombreux algorithmes qui combinent des classifieurs faibles pour construire un classifieur fort.

Catégorisation (d'images) ensemble de méthodes permettant de classer des images en fonction de la présence ou de l'absence d'objets particuliers.

Classification les systèmes de classification ont pour but de trier des objets en les associant à la bonne classe, c'est-à-dire au bon ensemble d'objets du même type.

Détection action, opération permettant de déceler la présence d'un objet, et éventuellement de préciser sa position.

En ligne un processus est dit « en ligne »lorsqu'il s'exécute au fur et à mesure du travail principal qu'il doit réaliser.

Fonction noyau fonction bilinéaire symétrique positive qui correspond à un produit scalaire dans un espace de grande dimension.

Hors ligne un processus hors-ligne réalise des tâches en différé du travail principal, soit avant en prévision d'une opération particulière à effectuer pour le travail principal, soit après pour traiter par exemple des informations récoltées pendant le travail principal.

Identification action permettant de préciser la nature de quelque chose, son type, sa catégorie. Plutôt employé pour les personnes, il s'agit dans ce cas d'établir l'identité de quelqu'un.

Intelligence Artificielle *(IA)* ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine.

Machine à noyau ce terme regroupe un ensemble de méthodes d'apprentissage artificiel destinées à résoudre des problèmes de reconnaissance ou de régression qui ont toutes recours à une fonction noyau pour résoudre le problème plus facilement en transformant l'espace d'entrée en un espace de dimension supérieure.

Précision nombre qui permet de quantifier la capacité d'un classifieur à reconnaître les objets positifs.

Rappel nombre qui permet de quantifier la capacité d'un classifieur à associer correctement les objets à la bonne classe.

Reconnaissance des formes ensemble de techniques et méthodes visant à identifier des motifs à partir de données brutes afin de prendre une décision quant à la catégorie à attribuer à ce motif.

ROC acronyme de *« Receiver Operating Curves »*, courbe couramment utilisée en apprentissage automatique pour évaluer les performances d'une classification, représentant le taux de bonnes détection en fonction des fausses alarmes.

RVM acronyme de *« Relevance Vector Machine »*, machine à noyau utilisant une formulation probabiliste.

Segmentation (d'images) technique visant à découper l'image en plusieurs zones homogènes.

SVM acronyme de *« Support Vector Machine »* (ou *« Séparateur à Vaste Marge »*), première machine à noyau développée par Vapnik [93].

Temps réel en informatique industrielle, on parle d'un système temps réel lorsque celui-ci contrôle un procédé physique à une vitesse adaptée à l'évolution de ce procédé.

Résumé

Cette thèse a pour but de détecter et de reconnaître les piétons dans les images. Celles-ci proviennent d'une caméra embarquée dans un véhicule circulant en milieu urbain. Le cahier des charges implique de nombreuses contraintes. Il faut notamment obtenir un système fonctionnant en temps réel pour être capable de détecter les piétons avant un éventuel impact. De plus, ces piétons peuvent être sujets à de nombreuses variations (taille, type de vêtements...), ce qui rend la tâche de reconnaissance d'autant plus ardue. La caméra étant mobile, aucune information ne pourra être extraite du fond. Nous mettons en œuvre différentes méthodes de vision par ordinateur, toutes basées apprentissage qui permettent de répondre à ces attentes. Le problème se traite en deux phases.

Dans un premier temps, une étape de traitement hors ligne nous permet de concevoir une méthode valide pour reconnaître des piétons. Nous faisons appel à une base d'apprentissage. Tout d'abord, un descripteur d'images est employé pour extraire des informations de ces images. Puis, à partir de ces informations, un classifieur est entraîné à différencier les piétons des autres objets. Chaque méthode a été paramétrée, testée et validée, tant au niveau description d'images que classification. La meilleure association de toutes ces méthodes a également été recherchée.

Dans un second temps, nous développons un système embarqué temps réel, qui soit capable de détecter les piétons avant une éventuelle collision. Nous exploitons directement des images brutes en provenance de la caméra et ajoutons un module pour segmenter l'image, afin de pouvoir intégrer les méthodes de description et classification précédentes et ainsi répondre à la problématique initiale.

Mots-clés : reconnaissance d'objets, détection, apprentissage, classification, description d'images, base d'apprentissage, caméra embarquée, temps-réel, piétons.

Abstract

This thesis aims to detect and to recognize pedestrians in images. These come from a camera embedded into a vehicle, which is driven in urban areas. These specifications involve many constraints. We have to obtain a real-time system for detect pedestrians before a possible collision. Moreover, pedestrians should be very variable (size, clothes, ...), which make the recognition more complicated. As the camera is moving, no information could be taken from the background. In my thesis, we implement several methods of computer vision, all based on a learning stage, which answer to all theses expectations. The problem is solved in two steps.

Firstly, a off-line stage allows us to design a method able to recognize pedestrians. We use a learning database. First of all, an image descriptor is used to extract informations of the images. Then, from these informations, a classifior is trained to differentiate pedestrians to others objects. We define all the parameters, and each method - of description or classification - is then tested and validated. The best association of these methods will be also searched.

Secondly, we realize an embedded real-time system, which is able to detect pedestrians before a possible collision. We directly use raw images coming from the camera et add a segmentation stage, so as to insert previous description and classification méthods and thus to answer to the initial problem.

Key-words : object recognition, detection, learning, classification, image description, learning database, embedded camera, real-time, pedestrians.