Association de classifieurs pour la reconnaissance de piétons dans les images

L. Leyrit T. Chateau C. Tournayre J.-T. Lapresté

LASMEA - UMR 6602 CNRS

Université Blaise Pascal
24 avenue des Landais 63177 AUBIERE CEDEX

{Laetitia.Leyrit,Thierry.Chateau,Christophe.Tournayre,Jean-Thierry.Lapreste}@lasmea.univ-bpclermont.fr

Résumé

Cet article présente une solution temps réel pour la détection de piétons dans une image, à partir d'une caméra embarquée dans un véhicule. L'enjeu d'une telle méthode est de définir un modèle générique capable de décrire la grande variabilité des piétons. Pour construire ce modèle, nous utilisons un ensemble d'apprentissage composé d'exemples positifs et négatifs. Une simple description de chaque exemple fournit un important vecteur de caractéristiques à partir duquel un classifieur faible peut être construit. Nous sélectionnons alors les classifieurs faibles pertinents par AdaBoost. L'échantillon résultant est ensuite utilisé comme vecteur binaire dans un classifieur de type machine à noyaux. La majeure contribution de ce papier est une association originale d'un AdaBoost pour sélectionner des classifieurs faibles pertinents, qui sont ensuite utilisés par un classifieur à noyau. Ainsi, les machines à noyaux permettent d'obtenir des séparateurs non-linéaires dans l'espace des classifieurs faibles, tandis qu'un AdaBoost en donne un linéaire. Les performances de cette méthode et l'application temps réel sont présentées.

Mots clefs

Reconnaissance d'objets, Classification, Apprentissage, Machines à noyaux, Détection de piétons.

1 Introduction

Cet article présente une approche pour détecter les piétons dans les images issues d'une caméra mobile. De telles méthodes peuvent être employées par des systèmes d'évitement de collision utilisant un système de vision embarqué sur un véhicule. Beaucoup de méthodes de détection existent déjà et peuvent être classifiées en deux grandes catégories : les approches basées sur un modèle et les méthodes basées sur un apprentissage. Ce papier s'intéresse tout particulièrement sur les approches basées apprentissage, où un ensemble d'apprentissage est nécessaire pour construire un modèle de piétons. Dans notre cas précis, 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, ...).

Comme les piétons sont très changeants, l'ensemble d'apprentissage utilisé doit être très important. De plus, chaque exemple est souvent décrit par un vecteur de grande taille, dont seulement un sous-échantillon est pertinent pour reconnaître la classe de l'objet. Plusieurs approches ont été proposées pour réduire ce nombre de caractéristiques [1]. Elles sont de deux types : les méthodes par filtrage (*filters*) et les méthodes par enveloppage (*wrappers*). Les méthodes par filtrage ([2],[3]) utilisent seulement l'ensemble d'apprentissage. Elles traitent toutes les données avant le début de l'apprentissage et conservent uniquement les caractéristiques utiles. 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 [4]. Des solutions ont été apportées pour les SVM dans [5] et [6].

Pour un traitement hors-ligne, le temps de calcul n'est pas le principal problème. Toutefois, des études comme [7] 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 mémoires sont employés inutilement. La réduction de la dimensionalité des données d'entrée permet ainsi d'améliorer les performances du classifieur. Dans une optique de gain de temps, le challenge est 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.

Les algorithmes de type AdaBoost peuvent également être utilisés pour la sélection de caractéristiques. Dans [8], 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. Une extension de cette méthode en cascade est développée dans [9] avec un classifieur final de type SVM. Les premières étapes

du classifieur utilisent un algorithme AdaBoost pour réduire l'espace des caractéristiques en ne sélectionnant que les pertinentes. La dernière étape consiste à entraîner un classifieur SVM qui construit un modèle de visage à partir des caractéristiques sélectionnées précédemment. Cette méthode combinée permet de réduire le nombre de caractéristiques dans les premières étapes de la cascade. Dans cette approche, l'algorithme AdaBoost est seulement utilisé pour sélectionner les caractéristiques pertinentes à partir d'un ensemble initial de grande taille.

Dans ce papier, nous proposons une approche originale qui consiste à sélectionner des classifieurs faibles par AdaBoost (et non de sélectionner des caractéristiques), et ensuite utiliser ces classifieurs faibles comme nouveaux vecteurs d'entrée pour entraîner un classifieur à noyau. Cette méthode fournit des séparateurs non-linéaires dans l'espace des classifieurs faibles et classifie correctement plus d'exemples comme montré sur la figure 1. Dans la première partie, nous expliciterons l'approche d'apprentissage combinée, la sélection de classifieurs faibles par AdaBoost et l'apprentissage par un classifieur fort à noyau ; dans la seconde partie, nous comparerons la méthode proposée aux autres classifieurs et montrerons des résultats expérimentaux.

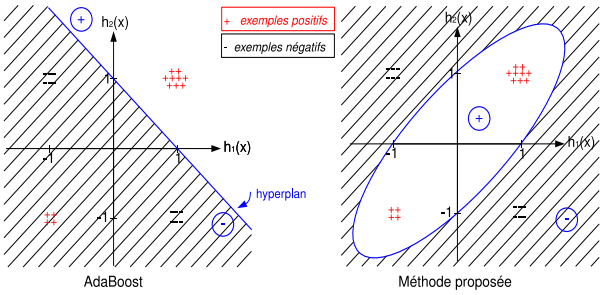

Figure 1 – ***La méthode présentée fournit des séparateurs non-linéaires alors que l'AdaBoost fournit des séparateurs linéaires***

Les exemples positifs sont notés par le symbole $+$, les exemples négatifs par $-$. $h_1(x)$ 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. La méthode proposée fournit des séparateurs non-linéaires et classifie correctement tous les exemples.*

2 Méthode proposée

Cette section décrit la méthode de classification par apprentissage que nous avons développée pour détecter des piétons. Dans cette partie, nous garderons des notations standards en représentant les étiquettes de sortie par un scalaire y qui pourra prendra prendre deux valeurs discrètes possibles correspondant à la classe de l'objet : $y = -1$ pour les exemples négatifs (non-piétons) et $y = 1$ pour les exemples positifs (piétons). Le vecteur $\mathbf{x} \in I\!R^Q$ représente les caractéristiques d'entrée fournies par les descripteurs d'image. Notons $S \doteq \{(\mathbf{x}^i, y^i)\}_{i=1}^N$ l'ensemble d'apprentissage composé par N échantillons de vecteurs de caractéristiques associées à leur étiquette correspondant à leur classe.

2.1 Sélection de classifieurs faibles

La reconnaissance de piétons est un problème de grande dimension et sélectionner des caractéristiques pertinentes à partir d'un grand ensemble de valeurs possibles est une tâche difficile. Les algorithmes de type AdaBoost peuvent permettre d'extraire en même temps ces caractéristiques intéressantes et des classifieurs faibles.

Les algorithmes de type AdaBoost introduits par Freund et Shapire [10] partent d'un principe simple. Comme l'opinion de plusieurs experts est meilleur que celui d'un seul, cet algorithme combine les décisions de plusieurs classifieurs faibles. Un poids uniforme est donné à chaque exemple de l'ensemble d'apprentissage. A chaque itération, un sous-ensemble est créé à partir de l'ensemble d'apprentissage S en accord avec les poids attribués à la précédente itération. Un classifieur faible h_t est créé à partir de cet échantillon et l'erreur ε_t est calculée pour tous les exemples de l'ensemble d'apprentissage. Le vecteur poids est mis à jour : le poids des éléments bien classés diminue tandis que celui des mal-classés augmente. De cette façon, le prochain classifieur faible se concentrera sur les éléments jusque là mal classés. Le processus est itéré jusqu'à obtenir un nombre préalablement fixé de classifieurs faibles sélectionnés ou jusqu'à ce que l'erreur sur l'ensemble d'apprentissage soit en-dessous d'un seuil donné. La règle de décision associée à l'AdaBoost est donc une combinaison linéaire des classifieurs faibles sélectionnés :

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

où α_t sont les poids estimés à chaque pas du processus d'apprentissage. L'équation (1) montre que la règle de classification est donnée par le signe d'une équation linéaire (hyperplan) dans l'espace des classifieurs faibles sélectionnés. Nous proposons d'introduire la non-linéarité dans la règle de classification en utilisant des fonctions de base à noyaux. Nous définissons $S^h \doteq \{(\mathbf{h}(\mathbf{x}^i), y^i)\}_{i=1}^N$ un nouvel ensemble d'apprentissage où $\mathbf{h}(\mathbf{x}^i) \in I\!R^T$ est un vecteur composé par l'estimation de la sortie de chaque classifieur sélectionné sur les paramètres $\mathbf{x}$ tels que $\mathbf{h}(x^i) \doteq \left(h_1(x^i), \ldots, h_T(x^i)\right)^T$.

2.2 Machines à noyaux

Les machines à noyaux ont été largement utilisées pour la classification ([11],[12],[13]). La formulation générale est donnée par :

$$y = \text{sign}\left(\sum_{m=1}^{M} w_m \phi_m(\mathbf{h}(\mathbf{x}))\right) \tag{2}$$

$\{\phi_m(\mathbf{h}(\mathbf{x}))|m = 1\ldots M\}$ sont des fonctions de base et $\{w_m|m = 1\ldots M\}$ sont les poids associés. Si $\phi_m(\mathbf{h}(\mathbf{x})) = h_m(\mathbf{x})$, $M = T$

et $w_m = \alpha_t$, nous obtenons la règle de classification linéaire de l'AdaBoost (voir équation (1)). Cependant, au lieu d'utiliser une forme linéaire, nous proposons des fonctions de base non-linéaires :

$$\phi_m(\mathbf{h}(\mathbf{x})) = k(\mathbf{h}(\mathbf{x}), h^m(x)) \tag{3}$$

où $k(\mathbf{h}(\mathbf{x}), \mathbf{h}^m(\mathbf{x}))$ est une fonction noyau.
La règle de classification peut être écrite sous une forme plus compacte dans l'équation suivante :

$$y = \text{sign}(\mathbf{w}^T \phi(\mathbf{h}(\mathbf{x}))) \tag{4}$$

où $\mathbf{w}^T = (w_1, w_2, ..., w_M)$ est un vecteur poids et $\phi(\mathbf{h}(\mathbf{x})) = (\phi(\mathbf{h}(\mathbf{x}^1)), \phi(\mathbf{h}(\mathbf{x}^2)), ..., \phi(\mathbf{h}(\mathbf{x}^N))^T$. Pour entraîner le modèle (estimation de $\mathbf{w}$), nous avons l'ensemble d'apprentissage $S^h = \{(\mathbf{h}(\mathbf{x}^i), y^i)\}_{i=1}^N$. Nous utilisons la norme Euclidienne pour mesurer l'erreur de prédiction dans l'espace des y, et donc le problème d'estimation revient à la formulation suivante :

$$\mathbf{w} := \arg\min_{\mathbf{w}} \{||\mathbf{w}^T \Phi - \mathbf{y}||^2\} \tag{5}$$

où $\Phi \doteq \left(\phi(\mathbf{h}(\mathbf{x}^1)), \phi(\mathbf{h}(\mathbf{x}^2)), ..., \phi(\mathbf{h}(\mathbf{x}^N))\right)$ est la matrice de *design* et $\mathbf{y} \doteq (y^1, ...y^N)^T$ est le vecteur de classe de l'ensemble d'apprentissage.
L'estimation du vecteur de paramètres $\mathbf{w}$ défini dans l'équation (5) peut se faire en utilisant un critère au sens des moindres carrés :

$$\mathbf{w}_{ls} = \mathbf{y}\Phi^+ \tag{6}$$

où Φ^+ correspond à la pseudo-inverse de Φ.
Des méthodes alternatives peuvent être utilisées pour estimer $\mathbf{w}$. Une solution consiste à placer des poids sur $\mathbf{w}$ avec l'objectif de fixer un grand nombre d'entre eux à zéro. Ces méthodes sont alors dites parcimonieuses car seul un petit nombre de fonctions de base sont conservées. Le modèle résultant est donc un modèle linéaire épars. Les SVM (*Support Vector Machine*) [14] sont un modèle linéaire épars où les poids sont estimés par une minimisation d'une fonction basée sur les multiplicateurs de Lagrange. D'autres modèles linéaires épars, comme les RVM (*Relevant Vector Machines*) [15] peuvent aussi être employés.
Les vecteurs utilisés pour les fonctions de base sont généralement composés par un échantillon de l'ensemble d'apprentissage S^h. Il est également possible d'utiliser tout l'ensemble d'apprentissage et dans ce cas $M = N$. La matrice Φ est symétrique et le système peut être résolu plus efficacement par une décomposition de Cholesky.
Nous avons fait le choix commun de prendre une Gaussienne comme fonction de base :

$$\phi_m(\mathbf{h}(\mathbf{x})) = \exp\left[-\left(\mathbf{h}(\mathbf{x}) - \mathbf{h}^m(\mathbf{x})\right)^2 / \sigma^2\right], \tag{7}$$

qui nous donne un modèle avec une fonction à base radiale (RBF) pour lequel le paramètre σ doit être ajusté. Si σ est trop petit, la matrice de *design* Φ est quasiment composée de zéros, et si σ est trop grand, Φ est quasiment composée de uns. Nous proposons d'ajuster σ en utilisant une optimisation non-linéaire qui maximise un critère empirique basé sur la somme des variances calculées pour chaque ligne de la matrice de *design* Φ :

$$\sigma := \arg\max_{\sigma}[-C(\sigma)] \tag{8}$$

avec

$$C(\sigma) = \sum_{n=1}^{N} \sum_{m=1}^{M} \left(\phi_m(\mathbf{h}(\mathbf{x})) - \overline{\phi}(\mathbf{h}^{(n)}(\mathbf{x}))\right)^2 \tag{9}$$

et

$$\overline{\phi}(\mathbf{h}(\mathbf{x})) = \frac{1}{M} \sum_{m=1}^{M} \phi_m(\mathbf{h}^{(n)}(\mathbf{x})) \tag{10}$$

Le classifieur ainsi obtenu sera désigné par KHA (*Kernel Hyperplane Approximation*) dans la section 3.

3 Résultats

Cette section décrit les expérimentations réalisées pour montrer les performances de la méthode proposée. De plus, nous décrirons une application avec une caméra embarquée sur un véhicule pour détecter des piétons en temps réel.

3.1 Performances du classifieur

Nous avons appliqué la méthode présentée dans la section 2 sur de la reconnaissance de piétons. Nous utilisons la base d'images fournies par Gavrila and Munder introduite dans [16]. Cette base est subdivisée en cinq parties ; chacune contient 4500 images positives et 5000 négatives. Il s'agit d'images de luminance, de taille 36x18 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 conditions d'illumination de fond variables. Chaque image de piéton a été aléatoirement reflétée et décalée de quelques pixels dans les directions horizontale et verticale. Les images d'exemples négatifs représentent l'environnement urbain : bâtiments, arbres, voitures, panneaux de signalisations, ... Cette base constitue les données utilisées pour l'entraîner et valider la méthode proposée. Deux sous-ensembles d'images différentes de piétons et de non-piétons sont sélectionnés pour l'apprentissage et la reconnaissance.

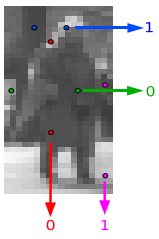

Figure 2 – ***Descripteurs par comparaison de niveaux de gris***

Pour un couple de points donnés, le descripteur retourne une valeur logique $d \in \{0,1\}$ correspondant au résultat du test $\{Intensite(point_1) \geq Intensite(point_2)\}$

Chaque image de l'ensemble d'apprentissage est décrite au travers plusieurs caractéristiques. Pour la détection de visages, beaucoup de méthodes utilisent des ondelettes de Haar parce que certaines sont très distinctives pour le nez, la bouche ou les yeux. Mais un piéton est un objet difficile à détecter à cause des différences d'habits, de taille, ... ajoutées aux variations classiques d'illumination et de fond. Des études ont développé des descripteurs de niveaux de gris comme dans [17] et [18]. Ainsi nous proposons d'utiliser ici un simple descripteur d'image basé sur le résultat de la comparaison de niveaux de gris comme présenté dans la figure 2. Notons $\mathbf{u}$ les coordonnées d'un point de l'image. $I(\mathbf{u})$ est l'intensité du pixel à ce point, c'est-à-dire le niveau de gris associé à ces coordonnées dans l'image. Considérons deux points de l'image, $\mathbf{u}_1$ et $\mathbf{u}_2$. Le descripteur effectue la comparaison suivante :

$$(I(\mathbf{u}_1) \geq I(\mathbf{u}_2)) \tag{11}$$

La valeur logique 1 est retournée si le test est vrai et 0 si le test est faux.
La taille de l'image est de 36x18 pixels. Nous faisons ces comparaisons entre deux points distincts appartenant à la même ligne ou à la même colonne. Ceci nous donne 5508 caractéristiques binaires pour les lignes et 11340 pour les colonnes.
Chaque image de l'ensemble d'apprentissage est donc représentée par un vecteur de caractéristiques $\mathbf{x}_i$ et sa classe y_i. Un classifieur faible est construit à partir de chaque attribut par $\mathbf{h}_i = \mathbf{P}_i.\mathbf{x}_i$. $\mathbf{P}_i$ est la polarité associée à chaque descripteur de façon à obtenir une erreur $e \leq 50\%$ sur l'ensemble d'apprentissage.
Nous obtenons ainsi un ensemble d'apprentissage
$S \doteq \{(\mathbf{x}^i, y^i)\}_{i=1}^N$, qui contient N images de piétons et non-piétons. La sélection de ces meilleurs classifieurs faibles est réalisée par AdaBoost, comme expliqué précédemment (voir paragraphe 2.1). Un nouvel ensemble d'apprentissage $S^h \doteq \{(\mathbf{h}(\mathbf{x}^i), y^i)\}_{i=1}^N$ est créé avec la sortie fournie par les classifieurs faibles sélectionnés pour chaque vecteur $\mathbf{x}_i$. Ce nouvel ensemble d'apprentissage est envoyé à un classifieur fort (avec une fonction à noyau gaussien). Nous avons testé différents types de machines à noyaux : SVM, RVM et KHA. En reconnaissance, le but est de déterminer la classe d'une nouvelle image. Celle-ci est décrite par tous les classifieurs faibles sélectionnés par AdaBoost durant la phase d'apprentissage. Le vecteur $\mathbf{h}(\mathbf{x})$ est ainsi obtenu. Ce vecteur est fourni en entrée au classifieur à noyau utilisé pendant l'apprentissage (SVM, RVM ou KHA), qui donne une décision sur la classe de l'objet. Les résultats obtenus par les différentes méthodes sont représentées par des courbes (*Receiver Operating Characteristic*). Une courbe ROC présente les variations et la sensibilité d'un test pour différentes valeurs du seuil de discrimination. L'axe des x représente le taux des faux négatifs (les non-piétons classifiés comme piétons) tandis que l'axe des y correspond au taux des vrais positifs (des piétons qui sont bien classifiés). Supposons qu'une courbe ROC passe par le point (0.1 ;0.9). Cela signifie que 90% piétons sont classifiés comme piétons, et que 10 % des non-piétons sont classés comme des piétons.
La sélection de classifieurs faibles par AdaBoost a retenu 2000 classifieurs faibles sur les 16848 disponibles. Ensuite différents apprentissages ont été réalisés : nous avons modifié la taille de l'ensemble d'apprentissage et le type de machine à noyau employé. Les performances de chaque méthode ont été évaluées en reconnaissance sur 1000 images de piétons et 1000 de non-piétons.
La figure 3 est une comparaison des différentes méthodes à noyau implémentées. L'apprentissage a été réalisé pour chacun sur 1000 images positifs et 3000 négatifs. Les résultats sont équivalents : aucune méthode n'a d'avantages ou de désavantages particuliers dans notre application. Pour 10% de fausses détections, les RVM donnent 97,6% de bonnes détections, les KHA 96,5% et les SVM 94,8%. Même si les RVM auraient tendance à donner des scores de reconnaissance supérieurs aux deux autres méthodes, les variations ne sont pas assez significatives pour sélectionner une méthode en particulier.
La figure 4 montre l'importance de l'ensemble d'apprentissage. Notre méthode a été testée ici avec l'approche KHA en augmentant graduellement la taille de l'ensemble d'apprentissage. Quand le nombre d'exemples augmente, le taux de reconnaissance augmente également.
Enfin, pour le dernier test, nous avons utilisé les trois premières parties de la base de données pour constituer l'ensemble d'apprentissage et les deux dernières pour créer l'ensemble de validation (comme dans [16]). Nous avons entraîné la méthode proposée (avec KHA) sur 4500 imagettes de piétons et 4500 de non-piétons sélectionnées dans les trois premières parties. Le classifieur ainsi obtenu est évalué sur toutes les imagettes contenues dans l'ensemble de test. Nous avons comparé ces résultats avec une méthode standard de classification par AdaBoost. Les résultats de la méthode proposée dépasse ceux d'un AdaBoost classique : pour 10% de fausses détections, l'AdaBoost donne 60,2% de bonnes détections alors que la méthode proposée en donne 76,5% (voir figure 5).

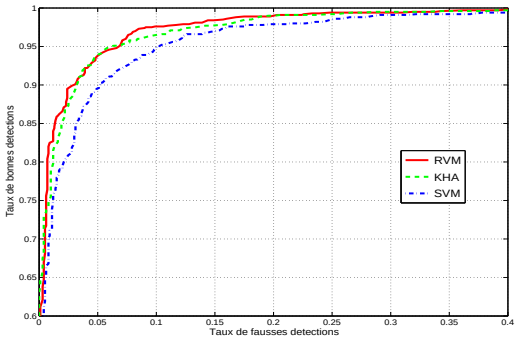

Figure 3 – ***Apprentissage avec différentes machines à noyaux***

Pour un ensemble d'apprentissage identique, les trois méthodes à noyaux donnent quasiment les mêmes scores de reconnaissance.

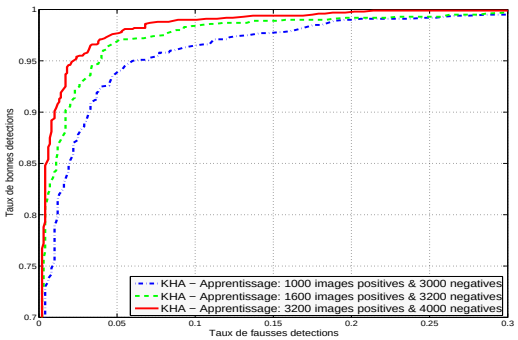

Figure 4 – ***Influence de la taille de l'ensemble d'apprentissage***

Le score de reconnaissance s'améliore lorsque la taille de l'ensemble d'apprentissage augmente.

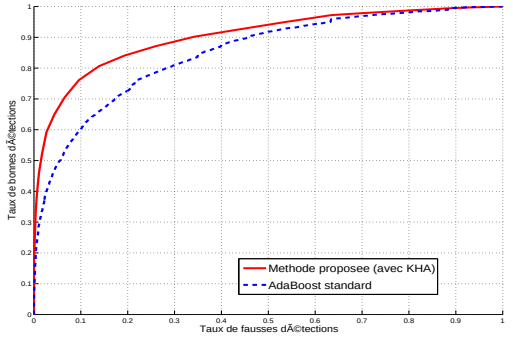

Figure 5 – ***Comparaison entre un classifieur AdaBoost classique et la méthode proposée***

La méthode proposée atteint des scores de reconnaissance supérieurs à ceux obtenus par un AdaBoost classique.

3.2 Application pour la détection temps réel de piétons à partir d'une caméra embarquée

Nous avons également testé la méthode proposée dans le cadre d'une application temps réel. Notre système est composé d'une caméra monoculaire embarquée sur le véhicule. Nous désirons détecter des piétons situés devant le véhicule afin d'avertir le conducteur de leur présence. Tout d'abord, nous définissons un espace de travail comme dans la figure 7. Nous utilisons des fenêtres glissantes *"Smart Sliding Window"* pour couvrir tout l'espace de travail (voir figure 6) : avec une supposition de monde planaire et des *a priori* sur la taille des piétons, des hypothèses sont générées dans l'espace 3D et projetées dans l'image. Nous obtenons ainsi un ensemble de sous-fenêtres candidates. Toutes ces sous-fenêtres sont réajustées en taille 36x18 pixels. Nous appliquons ensuite la méthode décrite dans la section 3.1 et une réponse est fournie pour chaque sous-fenêtre. Un exemple de détection est fourni dans la figure 8. L'implémentation a été réalisée en C++ sur un ordinateur classique double coeur 2.66 GHz. Il faut environ 220 ms pour traiter une image de taille 640x480 pixels. Les fenêtres glissantes donnent 82 imagettes de taille 36x18 pixels soit un temps de traitement de 2.7 ms par imagette.

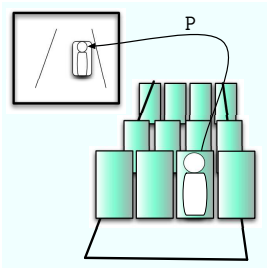

Figure 6 – ***Smart Sliding Window***

Avec une supposition de monde planaire et un a priori sur la taille des piétons, des hypothèses sont générés dans le monde 3D afin d'obtenir un ensemble de fenêtres candidates.

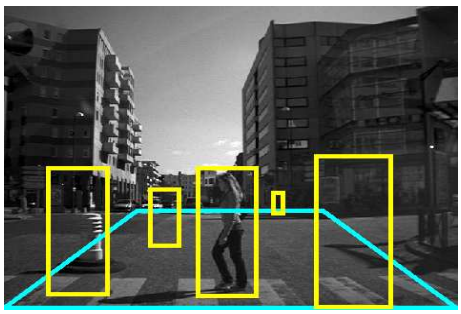

Figure 7 – ***L'espace de travail et les fenêtres glissantes***

L'espace de travail où nous cherchons la présence d'un piéton possible est en bleue ; nous utilisons ensuite les fenêtres glissantes comme celles en jaune pour couvrir tout l'espace de travail.

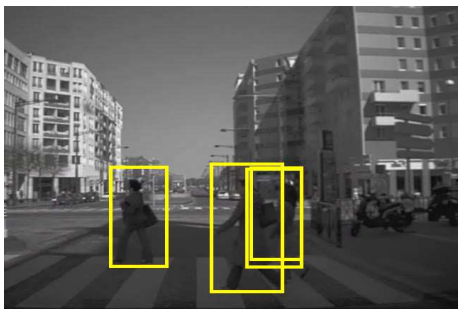

Figure 8 – ***Détection de piétons***

Les boîtes jaunes correspondent à des piétons détectés par le système.

4 Conclusion

Nous avons proposé une approche par apprentissage pour la détection de piétons. La méthode utilise un algorithme AdaBoost pour sélectionner des classifieurs faibles pertinents, qui sont ensuite injectés comme vecteurs binaires en entrée d'un classifieur à noyau. Les résultats montrent que la méthode donne des performances intéressantes, malgré la simplicité des descripteurs utilisés. De plus l'efficacité a été prouvée par une utilisation dans le cadre d'une détection temps réel de piétons avec un système de vision monoculaire embarqué sur un véhicule. Des travaux sont actuellement en cours pour tester les performances de cette méthode avec des caractéristiques plus complexes (histogrammes de gradients orientés [19] ou ondelettes de Haar [12]).

5 Remerciements

Nous remercions D. Gavrila and S. Munder pour la mise à disposition de leur base de données.
Ce travail est présenté dans le cadre du projet LOVe (*Logiciels d'Observation des VulnérablEs*) qui propose de contribuer à la sécurité routière, et en particulier à celle des piétons. Le but est d'obtenir à terme des logiciels fiables d'observation des vulnérables.

Références

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

[2] Kenji Kira et Larry A. Rendell. A practical approach to feature selection. Dans *Proceedings of the 9th International Workshop on Machine Learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[3] Mark Andrew Hall. Correlation-based feature selection for discrete and numeric class machine learning. Dans *Proceedings 17th International Conference on Machine Learning*, pages 359–366. Morgan Kaufmann, June 2000.

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

[5] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, et Vladimir Vapnik. Feature Selection for SVMs. Dans *Neural Information Processing Systems*, pages 668–674, 2000.

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

[7] Marine Campedel, Eric Moulines, Henri Matre, et Mihai Dactu. Feature Selection for Satellite Image Indexing. Dans *Image Information Mining - Theory and Application to Earth Observation (ESA-EUSC)*, Octobre 2005.

[8] Paul Viola et Michael Jones. Rapid object detection using a boosted cascade of simple features. Dans *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

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

[10] Yoav Freund et Robert E. Schapire. Experiments with a Nex Boosting Algorithm. Dans *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[11] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, et Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. Dans *Proceedings of the IEEE Conference of Intelligent Vehicles*, pages 206–212, Tokyo, Japan, June 2006.

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

[13] Amon Shashua, Yoram Gdalyahu, et Gaby Hayun. Pedestrian Detection for Driving Assistance : Single-frame Classification and System Level Performance. Dans *Proceedings of the IEEE Intelligent Vehicle Symposium (IV)*, Parma, Italy, 2004.

[14] Vladimir Vapnik. *Statistical Learning Theroy*. Wiley, Octobre 1998.

[15] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Resarch*, 1 :211–244, Mai 2001.

[16] Stefan Munder et Dariu M. Gavrila. An Experimental Study on Pedestrian Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(11), Novembre 2006.

[17] Yotam Abramson, Bruno Steux, et Hicham Ghorayeb. Yef (yet even faster) real-time object detection. Dans *ALaRT*, pages 5–13, 2005.

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

[19] Navneet Dalal et Bill Triggs. Histograms of Oriented Gradients for Human Detection. Dans *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, San Diego, California, USA, 2005.