Adaboost Discret Hétérogène à Contrainte Temps Réel : Application à la Détection de Piétons par Stéréovision

Loïc Jourdheuil, Nicolas Allezard, Thierry Chateau

Adaboost Discret Hétérogène à Contrainte Temps Réel : Application à la Détection de Piétons par Stéréovision

Loïc Jourdheuil[1] Nicolas Allezard[1] Thierry Chateau[2]

[1] CEA, LIST, Laboratoire Vision et Ingénierie des Contenus
[2] LASMEA - UMR UBP-CNRS 6602

CEA, LIST, Point Courrier 94, Gif-sur-Yvette, F-91191 France
loic.jourdheuil@cea.fr

Résumé

Cet article présente une méthode de détection de piétons basée sur la combinaison de descripteurs d'apparence et de profondeur. Des travaux récents ont montré l'intérêt de cette approche. Nous proposons deux contributions : 1) une étude comparative de différents descripteurs de profondeur, dans laquelle nous montrons que les meilleures performances sont atteintes par un descripteur simple, basé sur la moyenne des distances dans une sous-fenêtre de la région testée et 2) une adaptation de l'algorithme d'apprentissage Adaboost prenant en compte des classifieurs hétérogènes en terme de coût algorithmique. L'objectif de cette approche est de construire un classifieur à la fois performant en terme de taux de détection et de temps d'exécution. Nous montrons la pertinence de l'algorithme ainsi développé sur des séquences d'images réelles.

Mots Clef

Adaboost, stéréovision, temps réel.

Abstract

This paper presents a learning based method for pedestrians detection, combining both descriptors of appearance and depth. Recent work has shown the value of this combination. We propose two contributions : 1) a comparative study of various depth descriptors in which we propose a new fast descriptor based on the average distances in a sub-window in the tested area and 2) adaptation of the Adaboost algorithm in order to handle heterogeneous descriptors in terms of computational cost. The goal is to build a powerful descriptor according to both detection rate and execution time. We show the relevance of the proposed algorithm on real video data.

Keywords

Adaboost, stereovision, real time.

1 Introduction

La classification de piétons est un des outils les plus demandés dans le domaine de la vidéo-surveillance. Des solutions partielles existent dans le cas d'hypothèses restrictives telles que l'utilisation de caméras statiques dans des environnements épars, pour lesquelles la combinaison d'indices d'extraction fond-forme et d'apparence est utilisée dans une détection temps réel basée apprentissage [14]. D'autres travaux [3], [12] portent sur la fusion de l'apparence et du mouvement dans l'image dans le but d'améliorer les résultats.

La méthode présentée dans cet article ne peut pas utiliser de telles combinaisons car elle se focalise sur l'utilisation de capteurs de types caméras embarqués sur un objet mobile. Les problématiques actuelles associées à cette classe d'application sont essentiellement la fiabilité et le temps de traitement associé au détecteur. Ces dernières années, la communauté a porté une attention particulière à l'information de la carte de profondeur issue de capteurs de type stéréovision [7], [11], [4], [5] , ceci afin d'améliorer les performances de la classification. Nous proposons plusieurs de ces descripteurs et leurs performances lorsqu'ils sont utilisés dans un algorithme d'apprentissage de type Adaboost.

Des travaux récents [11] combinent à la fois des descripteurs d'apparence avec des descripteurs de disparité dans un unique Adaboost. Cependant, la fusion de plusieurs descripteurs de compléxité algorithmique hétérogène augmente généralement le temps de traitement. D'autre part, cette différence de temps de calcul n'est absolument pas prise en compte dans la phase d'apprentissage du détecteur alors qu'un des objectifs applicatifs est d'aboutir à une solution temps réelle. Les travaux de Saberian et al. [8] introduisent avec l'algorithme de boosting FCBoost, une prise en compte du coût algorithmique dans l'algorithme de boosting ChainBoost [13] en pénalisant en fonction du nombre de fausse détection. Nous proposons une modification d'Adaboost afin d'introduire une pénalité liée à la compléxité algorithmique de chaque classifieur faible dans la construction du classifieur fort.

Dans un premier temps, nous définirons et évaluerons différents descripteurs basés sur l'information de profondeur en les comparant à la méthode *Histogram of Oriented Gradian (HOG)* sur l'image de luminance. La deuxième partie présente une méthode novatrice d'apprentissage appelée adaboost discret hétérogène (ADH) permettant de fusionner plusieurs descripteurs tout en prenant en compte le coût algorithmique de chacun.

2 Descripteurs de profondeur

Cette section présente tout d'abord quatre descripteurs de la carte de profondeur. Les trois premiers sont basés sur les dérives locales de la carte. Le dernier est calculé sur la profondeur elle-même.
Dans la figure 1, l'image de gauche représente un piéton évoluant dans un environnement industriel. La pseudo image de droite de la figure montre la représentation de la carte de profondeur associée.

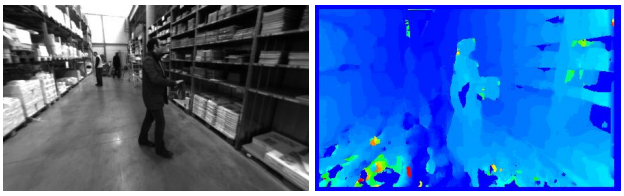


FIGURE 1 – Exemple d'image (à gauche) et la carte de profondeurs associé (à droite).

La figure 2 illustre, dans la partie haute, un ensemble de cartes de profondeur de piétons (exemples positifs dans un contexte d'apprentissage), et dans la partie basse, des exemples de cartes de profondeur de l'environnement (exemples négatifs). La création de cette carte est effectuée par une méthode classique de stéréovision dense temps réel, non abordée dans cet article.

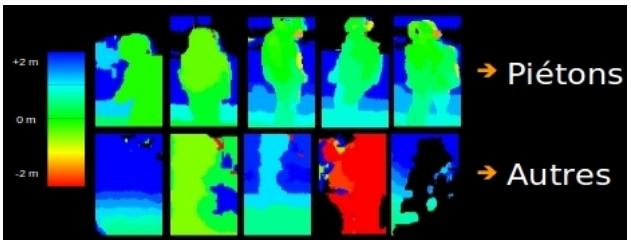


FIGURE 2 – Exemples positifs (haut) et exemples négatifs (bas) dans la carte de profondeur.

Une stratégie de fenêtre glissante (de ratio largeur sur hauteur $\frac{1}{2}$) parcourant le plan du sol, est utilisée pour évaluer l'image de profondeur (voir figure 3). Pour chaque fenêtre, un vecteur de caractéristiques est extrait, regroupant un ensemble de descripteurs locaux issus d'une carte de profondeur. Les vecteurs sont ensuite évalués selon un modèle créé par un apprentissage hors ligne de type Boosting.

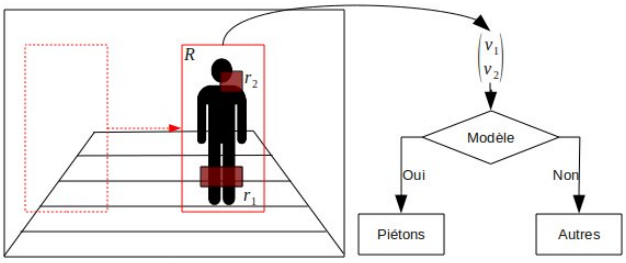


FIGURE 3 – Schéma d'évaluation de la classification d'un piéton.

2.1 Les descripteurs comparés

L'utilisation de descripteurs sur la carte de profondeur est récent. Nous avons adapté des descripteurs connus et efficaces sur l'apparence pour une utilisation sur la carte de profondeur. Nous avons également créé un descripteur spécifique à la carte de profondeur. Nous présentons ici quatre descripteurs de la carte de profondeur que nous comparons à une référence (HOG) sur des images réelles.

HOG-depth. Ce descripteur consiste à décrire la carte de profondeur à l'aide d'histogrammes d'orientation des gradients des distances (HOG). Ce type de descripteur est habituellement utilisé pour la représentation d'images d'apparence [2]. La version utilisée est enrichie de la magnitude du gradient, [1] : à partir de la carte de gradient des distances, un histogramme normalisé de 9 magnitudes d'orientation (de 0° à 180°) est construit. Cette histogramme est enrichi d'une 10^e valeur : la magnitude du gradient.

Matrice de Covariance (MatCov). Ce descripteur, dérivé des travaux de Tuzel [9], propose un codage de piétons à partir des matrices de covariances de l'apparence et de sa dérivée spatiale, calculées sur des sous-fenêtres de la zone d'analyse. Nous proposons un descripteur identique, mais utilisant des informations de la carte de profondeur avec les données $[x\ y\ d]$ où x et y sont les coordonnées du pixel et d la valeur de la profondeur de ce pixel. La matrice de covariance se trouve alors dans une variété riemannienne connexe $\mathcal{M}_{3\times3}$.

$$f_{cov} : \chi \rightarrow \mathcal{M}_{3\times3} \tag{1}$$

où $\mathcal{M}_{3\times3}$ est inclus dans $\mathbb{R}^{3\times3}$. Une fonction $\psi : \mathcal{M}_{3\times3} \rightarrow \mathbb{R}^6$ projette la variété dans un espace tangent selon la formule :

$$\psi(X) = vec_\mu(log_\mu(X)) \tag{2}$$

Où X et μ sont deux matrices symétriques positives. Avec :

$$vec_\mu(X) = upper(\mu^{-\frac{1}{2}} X \mu^{-\frac{1}{2}}) \tag{3}$$

Et :

$$log_\mu(X) = \mu^{\frac{1}{2}} log(\mu^{-\frac{1}{2}} X \mu^{-\frac{1}{2}}) \mu^{\frac{1}{2}} \tag{4}$$

μ est le point de la moyenne pondérée des matrices de covariances, résultant de la formule :

$$\mu = arg \min_{Y \in \mathcal{M}_{3\times 3}} \sum_{i=1}^{N} d^2(X_i, Y) \quad (5)$$

Une métrique indispensable pour comparer deux matrices de covariances est ainsi défini.

Covariance, Variance de la profondeur (CovVar). Le descripteur précédent (MatCov) présente comme inconvénient d'être assez lourd en terme de temps de calcul. Aussi une version simplifiée consiste à définir une distance qui ne tient plus compte des propriétés de la variété riemannienne. Dans la matrice de covariance $3 * 3$ créée avec les données $[x\ y\ d]$, les deux premières colonnes gardent leurs constantes. Ainsi, en réduisant la matrice à sa dernière colonne, nous obtenons le vecteur $[cov(x,d)\ cov(y,d)\ var(d)]$. Cette réduction nous permet d'utiliser une norme $L2$ pour définir la distance entre deux descripteurs.

Moyenne. Nous proposons un descripteur simple, basé sur la moyenne des différences entre la distance de test d et la distance mesurée z dans une sous-fenêtre de la zone d'analyse. Ce descripteur, associé à la région r_i, noté M_{r_i} se définit comme suit :

$$M_{r_i} = \frac{1}{n} * \sum_{d \in r_i} \begin{cases} z - d & \text{si } z \text{ défini} \\ 0 & \text{si } z \text{ indéfini} \end{cases} \quad (6)$$

Où n est le nombre de points de distances z définies. Des régions r_i de tailles variables, sont définies au cours de la phase d'apprentissage.
Ce descripteur basé sur les moyennes de distances peut rappeler le descripteur *DispStat* [11] utilisant un découpage fixe de la carte de disparité.

2.2 Comparaison des performances

La comparaison des performances des différents descripteurs a été réalisée en utilisant une machine d'apprentissage de type *Adaptive Boosting (Adaboost)* [6] et un descripteur de type *decision stump* pour HOG-depth, CovVar, Moyenne et HOG sur l'apparence. Dans le cas de l'apprentissage avec MatCov, le descripteur faible est une régression linéaire effectuée sur les composantes de la matrice symétrique.

Protocole d'évaluation. L'apprentissage a été effectué sur une vidéo tournée à l'intérieur d'un entrepôt où 10 000 imagettes d'exemples négatifs et 1 500 imagettes d'exemples positifs ont été extraites. Chaque imagette R est découpée en 1 482 régions rectangulaires r_i combinaison de zone de $\frac{1}{8}$ de la largeur et de $\frac{1}{16}$ de la hauteur de R. Dans les méthodes de type *boosting*, un nouveau classifieur faible est choisi à chaque *rounds* n d'apprentissage. Les n classifieurs faibles choisis sont regroupés dans un nouveau classifieur fort. Chaque classifieur fort a été évalué sur 10 000 imagettes d'exemples négatifs et 1 500 imagettes d'exemples positifs extraites d'une vidéo de test tournée dans un entrepôt différent de celui de l'apprentissage. Suivant les tests effectués, le maximum de performance sur la base test peut être atteint grâce à un classifieur fort, créé dans la phase d'apprentissage, après celui ayant permis de réduire l'erreur d'apprentissage à 0. Comme environ 100 *rounds* d'Adaboost sont nécessaires à nos classifieurs sur cette base d'apprentissage pour réduire l'erreur d'apprentissage à 0, nous avons choisi d'effectuer 300 *rounds* sur la base d'apprentissage afin de maximiser la probabilité d'encadrement du maximum de performance sur la base test.
Les 300 courbes *Receiver Operating Characteristics* (ROC) associées à chaque classifieur ont été comparées et la meilleure courbe a été retenue, suivant le critère :

$$Best = \arg \max_{t=1}^{300} Aire(Courbe_t) \quad (7)$$

Ainsi, chaque type de descripteurs pourra être évalué, à son maximum de performance, indépendamment du nombre de *rounds* effectués. Un test identique du descripteur HOG sur l'apparence servira de courbe de référence.

Résultats. La figure 4 présente les courbes ROC obtenues selon le protocole précédent. Les courbes seront comparées pour un taux de détection de 90% à la courbe de référence HOG (rouge). La courbe HOG-Depth (verte) a un taux de faux positifs de 15%, soit 11% supérieur à la référence (4%). Les classifieurs forts sélectionnés des courbes HOG-Depth et HOG ont été appris avec un nombre de *rounds* respectif de 270 et de 290. Le taux de faux positifs de la courbe MatCov (violette) s'élève à 8% (*rounds* 120) , celui de la courbe CovVar (bleu foncé) atteint 10% (*rounds* 208), soit respectivement à 4% et 6% supérieur à la référence. Le pourcentage de faux positifs de la courbe Moyenne (bleu claire) se situe près de 4% (*rounds* 212) comme celui de la référence.
Notre descripteur Moyenne créé spécifiquement pour travailler sur la carte de profondeur obtient des résultats aussi performant que le descripteur HOG utilisant l'apparence. Ceci confirme notre hypothèse de départ à savoir que la carte de profondeur est suffisante pour classifier des piétons. D'autres détecteurs conçus pour l'apparence peuvent être adapté à la carte de profondeur. Ces résultats encourageants sur l'utilisation de la carte de profondeur pour la classification des piétons, ouvre une nouvelle voie de fusion avec un descripteur d'apparence.

3 Adaboost Discret Hétérogène

Les travaux [11], de Walk et al. montrent que la fusion entre des descripteurs d'apparence, de mouvement et de disparité améliore, notablement, les résultats de la classification. Toutefois, elle génère une augmentation du temps de calcul qui peut représenter une contrainte non acceptable pour une application temps réel.
Dans cette seconde section, nous proposons de prendre en compte, dans l'algorithme Adaboost (Adaboost Discret

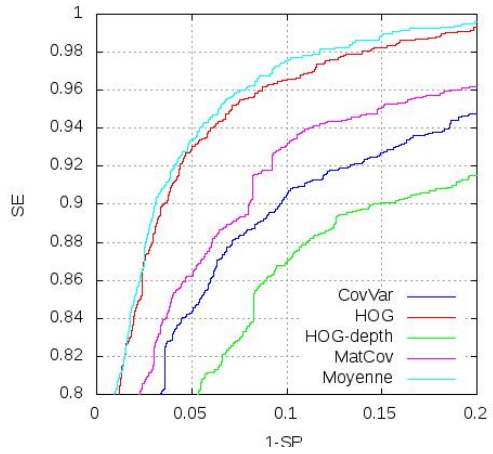


FIGURE 4 – Comparaison des performances de détection des classifieurs faibles sur la carte de profondeur (HOG sert de référence).

Hétérogène , AHD), une contrainte de coût algorithmique, adoucie par $\alpha \in [0;1]$, comme critères de sélection des descripteurs hétérogènes afin de créer un classifieur fort et rapide. Aussi, les descripteurs de distance présentés dans la section précédente seront combinés avec des HOG d'apparence dans un apprentissage global.
Suite aux résultats de l'étude précédente, la méthode CovVar sera préférée à MatCov puisque le passage de la méthode MatCov à CovVar dégrade peu les résultats tout en diminuant le temps de calcul.

3.1 Algorithme ADH

Soit C l'ensemble des algorithmes d'apprentissage faible d'Adaboost. Nous définissons un algorithme comme une composante des descripteurs. Nous associons à chaque algorithme, son coût algorithmique λ_c. Le coût algorithme peut être estimé à partir de la complexité du descripteur ou mesuré de manière statistique sur une machine cible.
Un coût algorithmique adouci normalisé $\tilde{\lambda}_c$, permettant une gestion de l'influence du coût algorithmique dans le calcul de la pseudo erreur pénalisée, est calculé selon l'équation :

$$\tilde{\lambda}_c = \frac{\lambda_c^\alpha}{\sum_{c=1,..,C} \lambda_c^\alpha} \tag{8}$$

où $\alpha \in [0;1]$ est un coefficient d'adoucissement que nous choisirons empiriquement.
Dans l'adaboost classique, la pseudo erreur est calculée de la façon suivante :

$$\epsilon_t^c = \sum_{i=1}^{N} p_i^t \left| \tilde{h}_t^c(\mathbf{x}_i) - y_i \right| \tag{9}$$

Avec p_i^t le poids normalisé et $\tilde{h}_t^c : \mathbf{x} \to \{0;1\}$ la réponse du classifieur faible .

Nous proposons de pénaliser la pseudo erreur ϵ_t^c par le coût algorithmique $\tilde{\lambda}_c$ dans ADH. Ainsi, la nouvelle équation de la pseudo erreur (pénalisée) devient :

$$\tilde{\epsilon}_t^c = \tilde{\lambda}_c \sum_{i=1}^{N} p_i^t \left| \tilde{h}_t^c(\mathbf{x}_i) - y_i \right| \tag{10}$$

Lorsque qu'une composante d'un classifieur est choisie par l'algorithme, les autres composantes liées du même classifieur sont calculées en même temps. En conséquence, le coût algorithmique λ_c de toutes les composantes des classifieurs déjà calculés change et doit être mis à jour. Nous avons donc introduit le coût algorithmique λ_0 de la classification sans calcul de méthode.
Pour chaque composante c déjà calculée, λ_c est remplacé par le coût algorithmique à vide λ_0. Le nouveau coût algorithmique adouci normalisé $\tilde{\lambda}_c$ est recalculé.
L'algorithme 1 page 8 détaille les différentes étapes de l'Adaboost Discret Hétérogène .

3.2 Coût Algorithmique

L'évaluation de la complexité du classifieur et la mesure statistique du temps de traitement sur une machine cible font partie des possibilités permettant d'estimer le coût algorithme λ_c des classifieurs faibles $c \in C$. Nous avons choisi de mesurer le temps (ns) moyen de calcul nécessaire pour calculer un classifieur faible. La figure 5 représente le coût algorithmique [1]. des méthodes CovVar, HOG, HOG-depth et Moyenne, ainsi que le coût algorithmique λ_0 (nommé vide) cité dans l'algorithme ADH, évalué par un classifieur retournant un vecteur vide.

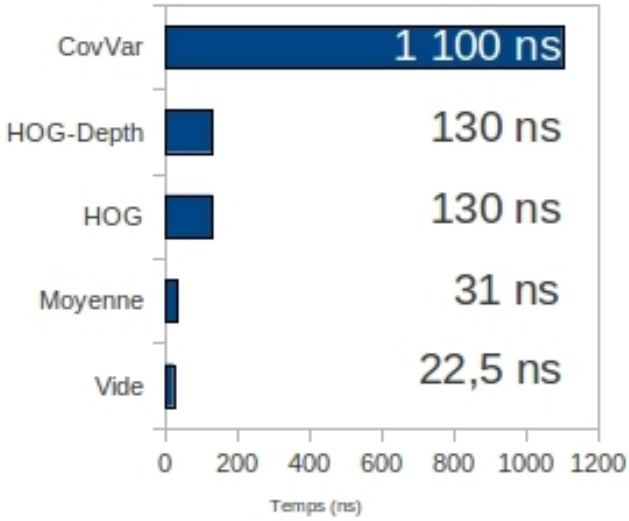


FIGURE 5 – Coût algorithmique λ_c des méthodes CovVar, HOG, HOG-depth et Moyenne et du cas vide en nanosecondes (ns) par classifieur faible.

1. Tous les tests ont été effectués sur un ordinateur de type PC équipé d'un processeur de $2.8GHz$

Le coût algorithmique varie de 1 $100ns$ pour CovVar à $31ns$ pour le classifieur Moyenne qui est proche de $22.5ns$ du classifieur vide. Ce classifieur est, aussi, quatre fois plus rapide que la référence HOG et sa dérivée HOG-Depth ($130ns$).

Nous définissons le coût algorithmique Λ_t du classifieur fort en sortie du *rounds* t de ADH suivant l'équation :

$$\Lambda_t = \Lambda_{t-1} + \lambda_c \tag{11}$$

où λ_c est le coût algorithmique du classifieur faible choisi au *rounds* t.

3.3 Choix du coefficient d'adoucissement α

L'ordre de grandeur du coût algorithmique $\tilde{\lambda}_c$ influe fortement sur le choix du classifieur faible dans l'algorithme ADH. Afin de gérer cette influence, un coefficient d'adoucissement α lui est appliqué (équation 8). Ce dernier est choisi en fonction de l'erreur d'apprentissage et du temps de calcul Λ_t.
Dans cette partie, nous présentons l'influence et le choix du coefficient α pour un apprentissage ADH avec HOG+Moyenne.

Erreur d'apprentissage. L'algorithme adaboost minimise l'erreur d'apprentissage à chacun de ses *rounds*. La figure 6 représente l'erreur d'apprentissage ϵ_t en fonction du nombre de *rounds* t pour un apprentissage Adaboost classique (rouge) et l'algorithme ADH (bleu) où, à chaque *rounds*, l'erreur d'apprentissage d'Adaboost est inférieure à celle d'ADH.

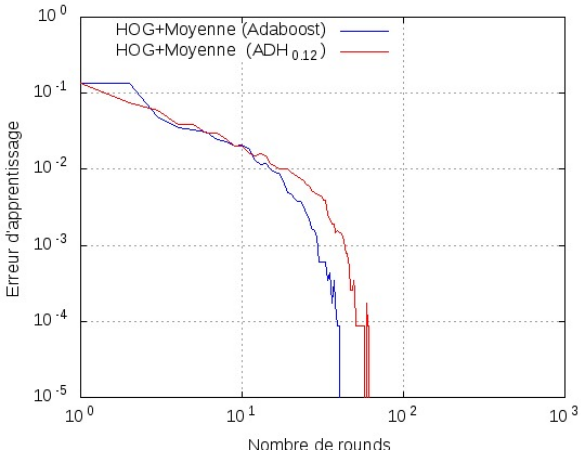


FIGURE 6 – Erreur d'apprentissage en fonction du nombre de *rounds* pour un apprentissage HOG+Moyenne avec Adaboost classique et ADH.

Dans la figure 7, nous avons représenté l'erreur d'apprentissage en fonction du coût algorithmique des classifieurs forts Λ_t (équation 11) pour prendre en compte le coût algorithmique. Dans ce cas, l'apprentissage ADH (bleu) réduit plus rapidement l'erreur d'apprentissage qu'un apprentissage Adaboost classique (rouge) quelque soit le coût algorithmique (Temps).

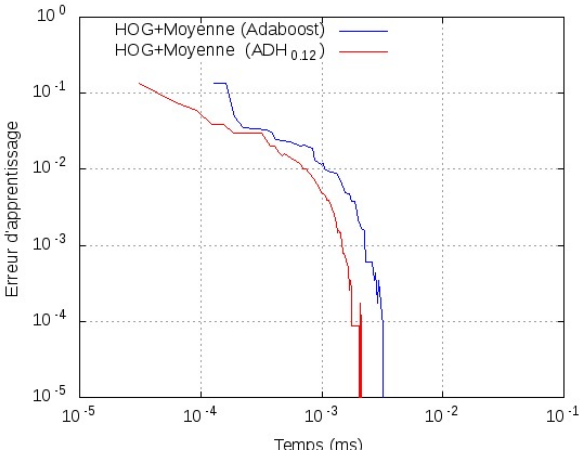


FIGURE 7 – Erreur d'apprentissage en fonction du temps de calcul (ms) pour un apprentissage HOG+Moyenne avec Adaboost classique et ADH.

Détermination du coefficient α. Les expériences ont montré que le coût algorithmique est très influant sur la sélection des classifieurs faibles. Nous avons cherché à gérer cette influence en lui attribuant un coefficient d'adoucissement α (équation 8). La figure 8 représente l'erreur d'apprentissage en fonction du coût algorithmique (Temps) pour un échantillon représentatif des valeurs possibles $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$. Dans cette figure, la courbe α_1 (noire) représente l'erreur d'apprentissage sans coefficient d'adoucissement. Le pourcentage d'erreur d'apprentissage de cette courbe a atteint seulement 5% d'erreur à la fin des 300 *rounds* d'apprentissage. La courbe α_0 (bleu foncé) représente l'évolution de l'erreur d'apprentissage sans coût algorithmique (adoucissement maximum, identique à un adaboost classique). Les autres courbes sont liées à des valeurs du coefficient $\alpha \in [0; 1]$. La valeur de ce coefficient a un effet très important sur les performances de l'algorithme. La courbe $\alpha_{0.12}$ (verte) semble être un bon compromis entre la décroissance de l'erreur d'apprentissage et le temps de calcul.
Pour vérifier l'efficacité de la courbe $\alpha_{0.12}$, nous avons utilisé les données des aires des courbes ROC du protocole de la section 2.2 (page 3) que nous avons affiché en fonction du coût algorithmique Λ_t. La figure 9 représente 1-l'aire des courbes ROC en fonction du temps pour $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$.
Elle confirme les performances de la courbe $\alpha_{0.12}$ (verte).

3.4 Temps de calcul fixe

Cette partie présente les expérimentations menées sur la base test, afin de comparer les performances de la méthode ADH à celles d'Adaboost classique sur les trois classifieurs

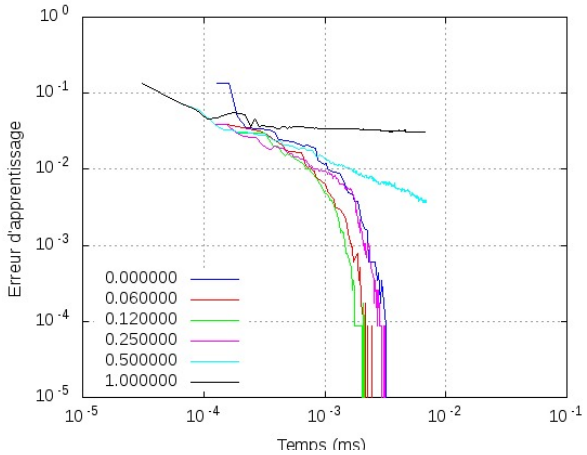


FIGURE 8 – Erreur d'apprentissage de la méthode HOG+Moyenne en fonction du temps de calcul (ms) pour $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$

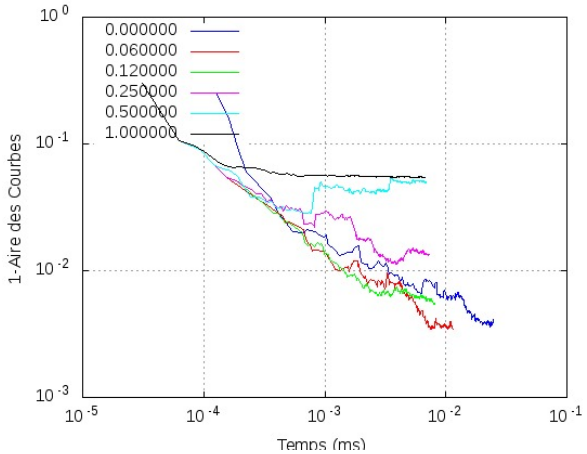
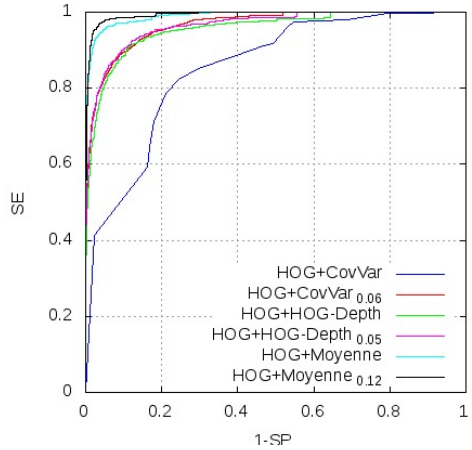


FIGURE 9 – 1-l'aire des courbes ROC de la méthode HOG+Moyenne en fonction du temps de calcul (ms) pour $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$.

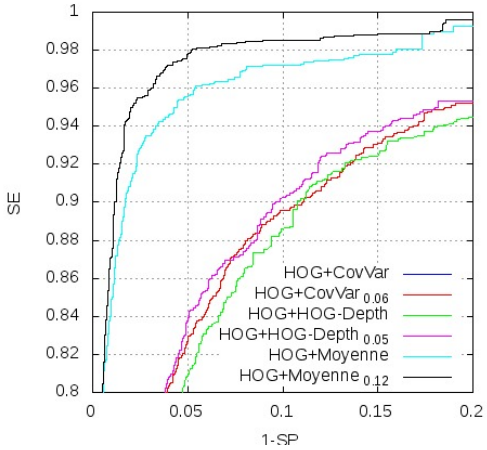
faibles HOG+Moyenne, HOG+CovVar et HOG+HOG-Depth. Le protocole utilisé est identique à celui présenté dans la section 2.2 (page 3). Une limite de temps maximum lim à la variable temps Λ_t a été définie pour tenir compte des contraintes temps réel.

$$Best = \arg\max_{t=1}^{300} \begin{cases} Aire(Courbe_t) & \Lambda_t \leqslant lim \\ 0 & \Lambda_t > lim \end{cases} \quad (12)$$

Les limites lim ont été fixées à $2.5\mu s$ et à $1.25\mu s$ correspondant respectivement à 10 et 20 images par seconde. Ce temps limite a été estimé pour prendre en compte un nombre de 40 000 tests de sous-fenêtre pour évaluer une image. La figure 10 (respectivement 11) montre les six courbes ROC correspondant à $lim = 2.5\mu s$ (respectivement $lim = 1.25\mu s$). Les parties (a) représentent les courbes complètes dans l'intervalle $[0; 1] \times [0; 1]$, tandis que les parties (b) sont les zooms sur l'intervalle $[0; 0.2] \times [0.8; 1]$.



(a) Intervalle $[0; 1] \times [0; 1]$



(b) Intervalle $[0; 0.2] \times [0.8; 1]$

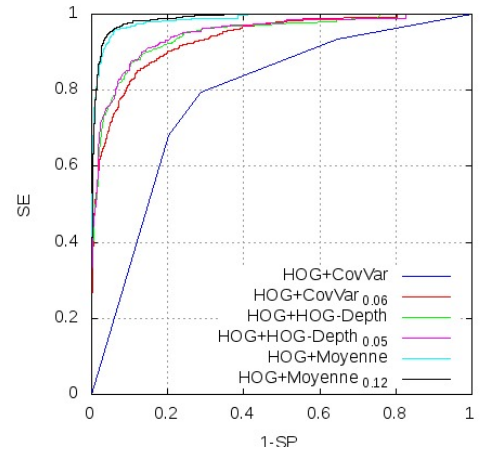
FIGURE 10 – Courbes ROC de comparaison entre un adaboost et ADH pour un temps de calcul $\Lambda_t \leqslant 2.5\mu s$.

Dans les deux figures, les courbes HOG+Moyenne (bleu clair), HOG+CovVar (bleu foncé) et HOG+HOG-Depth (verte) sont apprises avec un Adaboost classique. Les courbes HOG+Moyenne (noir), HOG+CovVar (rouge) et HOG+HOG-Depth (violette) sont apprises avec ADH et un coefficient α respectivement de valeur 0.12, 0.06 et 0.05 déterminée suivant le protocole présenté précédemment.

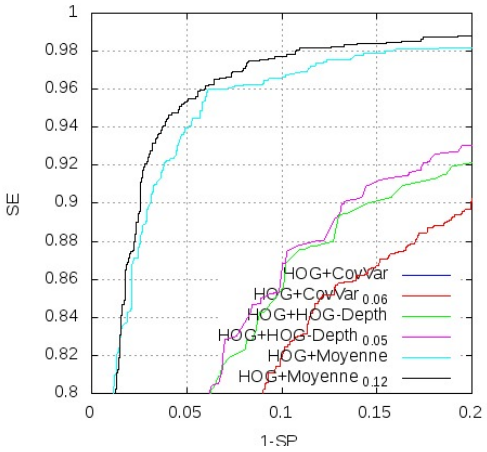
Sur les deux figures, les courbes générées par ADH présentent un taux de détection (SE) supérieur à leurs homologues générées par Adaboost classique quelque soit le taux de faux positifs (1-SP).

D'autre part, quelque soit le temps sélectionné, le meilleur classifieur fort est donné par l'apprentissage HOG+Moyenne (noir).

La figure 4 (page 4), faisait apparaître que la courbe CovVar était plus performante que HOG-Depth. Or, le coût al-



(a) Intervalle $[0; 1] \times [0; 1]$



(b) Intervalle $[0; 0.2] \times [0.8; 1]$

FIGURE 11 – Courbes ROC de comparaison entre un adaboost et ADH pour un temps de calcul $\Lambda_t \leqslant 1.25mus$.

gorithmique de CovVar est dix fois supérieur à celui de HOG-Depth (cf. figure 5). Les figures 10 et 11 montrent que la fusion par ADH conduit à une combinaison HOG+HOG-Depth plus performante que HOG+CovVar.

4 Conclusion et perspectives

Dans cet article, nous avons proposé et présenté différents descripteurs faibles issus des descripteurs de la carte de profondeur. Les résultats comparatifs ont mis en évidence que le descripteur simple que nous avons proposé (Moyenne), basé sur la moyenne locale de la profondeur, donne des performances équivalente à HOG, choisi comme descripteur de référence. Dans une seconde phase, nous avons proposé une modification de l'algorithme Adaboost en assignant un coût algorithmique λ_c à chaque classifieur en fonction de son temp de calcul.

Ce nouvel algorithme d'apprentissage a été évalué sur la fusion d'un descripteur d'apparence (HOG) avec des descripteurs de la carte de profondeur (CovVar, HOG-depth et Moyenne). Nous avons montré que, pour un temps de traitement fixe, la courbe ROC de l'algorithme proposé est supérieure à cette d'une approche Adaboost classique. L'algorithme ADH permet également de fournir, en sortie de traitement, une évaluation du coût algorithmique du classifieur fort Λ_t qui peut être fixé.

L'algorithme ADH pourrait être adapté à l'utilisation d'une cascade [10] en fixant un temps de traitement à chaque étage. De plus, le coût algorithmique de chaque classifieur peut être redéfini à chaque étage en fonction du temps de traitement choisi afin de favoriser les classifieurs lents et performants en fin de cascade.

Références

[1] J. Begard, N. Allezard, and P. Sayd. Real-time human detection in urban scenes : Local descriptors and classifiers selection with adaboost-like algorithms. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8, june 2008.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893, june 2005.

[3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *In European Conference on Computer Vision*. Springer, 2006.

[4] M. Enzweiler and D. Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *Image Processing*, 20(10) :2967–2979, 2011.

[5] A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press, June 2008.

[6] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5) :771–780, September 1999.

[7] L. Jourdheuil, N. Allezard, and T. Chateau. Vers une détection de piétons temps réel par apprentissage de forme dans l'image de profondeur. In *ORASIS - Congrès des jeunes chercheurs en vision par ordinateur*, Praz-sur-Arly, France, 2011. INRIA Grenoble Rhône-Alpes.

[8] M.J. Saberian and N. Vasconcelos. Boosting classifier cascades. NIPS, 2010.

[9] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *Transactions on Pattern Analysis and Machine Intelligence*, 30 :1713–1727, 2008.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Vattern Recognition*, 2001.

[11] S. Walk, K. Schindler, and B. Schiele. Disparity statistics for pedestrian detection : Combining appearance, motion and stereo. *ECCV*, pages 182–195, 2010.

[12] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 794–801, june 2009.

[13] Rong Xiao, Long Zhu, and Hong-Jiang Zhang. Boosting chain learning for object detection. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 709 –715 vol.1, oct. 2003.

[14] J. Yoa and J.M. Odobez. Fast human detection from videos using covariance features. In *8th European Conference on Computer Vision Visual Surveillance workshop*, Marseille, October 2008.

Algorithme 1 : Adaboost Discret Hétérogène

Entrées :

- Un ensemble de N exemples labellisés $\{(\mathbf{x}_n, y_n)\}_{n=1,..,N}$
- Une distribution de probabilité D associée aux N exemples.
- Un ensemble de C algorithmes d'apprentissage faibles avec leur coût algorithmique associé $\{(\texttt{WeakLearn}_c, \lambda_c)\}_{c=1,..,C}$.
- Le coût algorithmique de vide λ_0
- Un coefficient d'adoucissement $\alpha \in [0; 1]$
- Un nombre T d'itérations.

Initialise :

- Le vecteur de poids $w_i^1 = D(i)$ pour $i = 1, .., N$
- Un vecteur de coûts normalisés $\tilde{\lambda}_c = \dfrac{\lambda_c^\alpha}{\sum_{c=1,..,C} \lambda_c^\alpha}$
- L'erreur initiale $\varepsilon_0 = 0,5$

Pour chaque $t = 1, 2, ..., T$ **Faire**

(1)
$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

(2) **Pour chaque** $c = 1, 2, ..., C$ **Faire**

(2.1) Appeler le classifieur faible $\texttt{WeakLearn}_c$ avec la distribution $\mathbf{p}^t$ qui retourne un classifieur faible $\tilde{h}_t^c : \mathbf{x} \to \{0; 1\}$ de coût algorithmique λ_c.

(2.2) Calculer la pseudo erreur pénalisée
$$\tilde{\epsilon}_t^c = \tilde{\lambda}_c \sum_{i=1}^N p_i^t \left| \tilde{h}_t^c(\mathbf{x}_i) - y_i \right|$$

(3) Sélectionner le classifieur faible qui minimise la pseudo-erreur pénalisée :
$$h_t = \tilde{h}_t^{\hat{c}} \mid \hat{c} = \arg \min_{c=1,..,C} \tilde{\epsilon}_t^c$$

(4) Calculer l'erreur : $\epsilon_t = \sum_{i=1}^N p_i^t \, |h_t(\mathbf{x}_i) - y_i|$ et le coefficient $\beta_t = \dfrac{\epsilon_t}{1 - \epsilon_t}$

(5) Mettre à jour le vecteur des poids
$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(\mathbf{x}_i) - y_i|}$$

(6) Mise à jour du nouveau coût algorithmiques $\lambda_c = \lambda_0$ pour tout c calculé en même temps de $\hat{c}$.

(7) Normalisation du coût $\tilde{\lambda}_c = \dfrac{\lambda_c^\alpha}{\sum_{c=1,..,C} \lambda_c^\alpha}$ avec $\alpha \in [0; 1]$

Sortie :

le classifieur fort :
$$H(\mathbf{x}) = \begin{cases} 1 \text{ si } \sum_{t=1}^T \left(\log \dfrac{1}{\beta_t} \right) h_t(\mathbf{x}) \geq \frac{1}{2} \sum_{t=1}^T \log \dfrac{1}{\beta_t} \\ 0 \text{ sinon} \end{cases}$$