Vers une détection de piétons temps réel par apprentissage de forme dans l'image de profondeur

Loïc Jourdheuil, Nicolas Allezard, Thierry Chateau

Vers une détection de piétons temps réel par apprentissage de forme dans l'image de profondeur

Towards a real time detection of pedestrians by learning form in the depth image

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 pour la détection des piétons dans une image de profondeur, acquise à partir d'une paire de caméras calibrées (tête stéréoscopique). Nous proposons d'apprendre les caractéristiques d'un piéton à l'aide d'un algorithme de type boosting, à partir de classifieurs faibles issus de caractéristiques statistiques simples des distances à l'intérieur d'une sous-fenêtre de la zone d'analyse. Ces hypothèses seront ensuite confirmées par un second détecteur, classique, basé sur l'analyse de l'apparence visuel des piétons. La méthode ainsi obtenue permet une détection des piétons, à une cadence proche du temps réel, grâce à l'utilisation de la notion d'image intégrale, appliquée au calcul des descripteurs 3D. La méthode proposée est comparée à une méthode de détection d'obstacles verticaux classique sur une séquence vidéo réelle annotée. Elle permet de réduire le nombre de faux positifs d'approximativement 60% par rapport à une méthode de détection d'obstacles, tout en diminuant le temps de calcul.

Mots Clef

Détection, piétons, vision stéréo, disparité, apprentissage.

Abstract

This paper presents a method for detecting pedestrians in a depth image, established from a pair of calibrated cameras (stereoscopic head). We propose to learn the characteristics of a pedestrian thanks to a boosting type algorithm, using weak classifiers created from simple statistical characteristics of distance within a sub-window of the area of analysis. These assumptions will then be confirmed by a second, classic detector based on analysis of the visual appearance of pedestrians. Our resulting method allows for the detection of pedestrians at a rate close to real time, using the concept of integral image, applied to the calculation of 3D descriptors. The proposed method is compared with a traditional method for detecting vertical obstacles on a real annotated video sequence. Our method reduces the number of false positives by approximately 60% compared to an obstacle-detection method, while reducing the computation time.

Keywords

Detection, pedestrian, stereo vision, disparity, learning.

1 Introduction

La détection d'objets et en particulier celle des piétons fait partie des outils de la protection et de la sécurité d'aujourd'hui. Les problématiques actuelles restent la fiabilité et le temps de traitement des informations. Néanmoins, la plupart des méthodes de l'état de l'art présentent, soit des temps d'exécution importants, soit un nombre de faux positifs détecté trop grand. Ainsi, pour diminuer ces deux critères, les recherches ont été menées suivant deux voies :

– La première approche consiste en l'optimisation des performances de la classification à l'aide d'une cascade [10], constituée d'un ensemble de classifieurs de complexité croissante. Ainsi, les zones les plus simples à classifier sont rejetées en début de cascade, les calculs les plus complexes étant réservés aux zones de l'image les plus ambiguës. Pour qu'une boîte soit validée comme piéton, il faut donc que chaque étage de la cascade affirme qu'il s'agisse d'un piéton. Si un seul étage échoue, alors la boîte sera rejetée.
– La seconde approche utilise un pré-calcul pour détecter des obstacles qui sont ensuite évalués par le classifieur. Différentes solutions existent en fonction du matériel utilisé. Certaines méthodes utilisent un télémètre laser [4] afin d'obtenir la distance de chaque point du passage du laser. Ces méthodes ont l'avantage d'être précises sur les distances données sur une ou plusieurs lignes. Mais l'information est difficilement utilisable pour caractériser un piéton et il faut alors les coupler à un capteur plus riche comme une caméra.
  D'autres solutions utilisent plusieurs caméras afin de re-

monter aux informations de profondeur. Dans l'image de disparité, les méthodes comme celles proposées dans [1, 5, 2], recherchent suivant le plan du sol, les obstacles potentiels. Ces obstacles sont caractérisés par une zone dont le nombre de pixels à une distance donnée dépasse un certain seuil. Si ce seuil est atteint, la boîte est considérée comme un obstacle. En présence d'un environnement encombré par des murs ou des voitures, ces méthodes ont l'inconvénient de générer beaucoup de zones d'intérêts.

La méthode que nous proposons peut se classer dans la deuxième catégorie. C'est une méthode basée apprentissage qui utilise les informations fournies par une carte de disparité pour classifier une fenêtre 3D candidate comme un objet de type piéton. Nous considérons, comme hypothèse principale, que la forme d'un piéton est suffisamment caractéristique pour permettre un apprentissage statistique, par des techniques de *Machine Learning*. L'algorithme d'apprentissage utilisé est le Boosting de classifieurs faibles. Nous proposons alors de définir des classifieurs faibles à partir de statistiques géométriques effectuées sur des sous-fenêtres de la zone candidate. De plus, le calcul de ces classifieurs s'effectue rapidement grâce à l'utilisation de la notion d'image intégrale .

Cet article est structuré de la manière suivante. Nous présentons la méthode de détection de piéton développée, à partir de données issues d'une tête stéréoscopique, dans une première partie. La deuxième partie décrit les différentes expérimentations effectuées pour comparer l'algorithme proposé à l'état de l'art. Nous concluons dans une dernière partie.

2 Méthode proposée

Cette section décrit la méthode de détection de piéton proposée.

2.1 Principe

La scène est observée à l'aide d'une tête stéréo calibrée, et la carte de disparité (voir Figure 1) est ensuite construite, à partir de la paire d'images rectifiées, par un algorithme de type *Sum of Absolute Differences (SAD)*. la carte de disparité est ensuite filtrée afin d'éliminer les zones qui ne sont pas assez texturées pour être correctement appareillées.

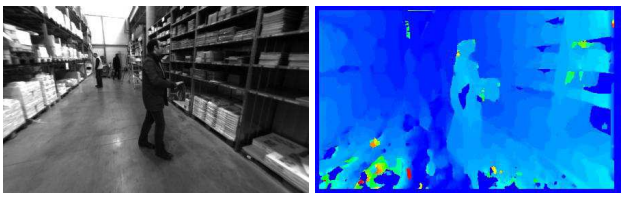


FIGURE 1 – Exemple d'image traitée (à gauche) et de la carte de disparité correspondante.

La génération de régions d'intérêts s'effectue par une stratégie de fenêtres glissantes, parcourant le plan du sol (2.2) dans l'image de disparité (voir figure 2) . Pour chaque fenêtre, un vecteur de caractéristiques est extrait (2.3), regroupant un ensemble de descripteurs locaux issus d'une carte de disparité. Les vecteurs sont ensuite évalués selon un modèle qui a été créé par un apprentissage hors ligne de type Boosting (2.5). Les boîtes positives décrivant le même piéton sont finalement fusionnées.

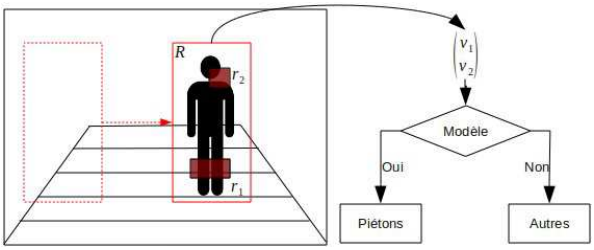


FIGURE 2 – Principe de la méthode proposée. Une stratégie de fenêtres glissantes 3D est utilisée pour générer des régions candidates. Un vecteur de caractéristiques est extrait, regroupant un ensemble de descripteurs locaux issus d'une carte de disparité. Les vecteurs sont ensuite évalués selon un modèle qui a été créé par un apprentissage hors ligne de type Boosting.

2.2 Parcours suivant le plan du sol

En partant de l'hypothèse qu'un piéton se trouve très probablement sur le plan du sol, nous proposons une stratégie de fenêtres glissantes, dans l'espace 3D. Soit R une zone de test de dimension $W*H$ dans la carte de disparité, avec une ratio constant $H = 2 * W$. On définit $\mathcal{R} \doteq \{r_i\}_{i=1,..,I}$ un ensemble de sous-fenêtres de R. De plus, on note z la distance à laquelle se trouve la fenêtre R du repère monde [1]. Pour chaque triplé distance z, taille de piéton t et coordonnée x (en pixels) de la ligne de la carte de disparité associée à z, une zone de test R est définie. un vecteur de paramètres (descripteur) $\vec{M} \doteq (M_1, ..., M_I)^T$ est construit. l'algorithme 1 présente cette stratégie de fenêtres glissantes 3D.

Les boîtes qui sont validées comme piéton sont ensuite fusionnées par un algorithme de type *Quality Threshold Clustering (QTclust)* classique [6] suivant la position de la boîte sur le plan du sol. Les boîtes résultantes de l'algorithme QTclust sont ensuite filtrées afin d'éliminer les boîtes occultées par d'autres boîtes.

2.3 Calcul des descripteurs

Nous postulons que l'observation de la géométrie d'un piéton à travers la carte de disparité permet de caractériser ce dernier. Nous proposons une approche de description basée sur un découpage de la fenêtre candidate en sous-fenêtres de tailles et de position variables. Viola et Jones dans [10] proposent un découpage similaire, dans le cas d'images de luminance, et utilisent des ondelettes de Haar

1. Le repère monde est associé au repère d'une des deux caméras de la base stéréo

Algorithme 1 : Parcours de l'image selon une stratégie de fenêtres glissantes 3D

Pour chaque *Distance* z **Faire**
  Pour chaque *Taille de piéton* t **Faire**
    Calcul de la position de la boîte dans l'image
    Pour chaque *Pixel de la ligne* x **Faire**
      Décalage de la boîte à la position x
      Calcul des descripteurs et du vecteur caractéristique
      Calcul du score de classification du vecteur

Fusion des boîtes positives

pour coder l'apparence d'un piéton. Dans [9], Tuzel propose également un découpage en sous-fenêtres, mais avec un descripteur basé sur le calcul de matrices de covariance dans chaques sous-fenêtres. Cette stratégie conduit à un vecteur de caractéristiques de taille importante, pour lequel un classifieur faible sera associé à chaque élément du vecteur. Une sélection des caractéristiques (classifieurs faibles) les plus pertinentes est ensuite réalisée par un algorithme de type *Adaptive Boosting (Adaboost)* [3].

1. Le descripteur proposé est basé sur le calcul d'une moyenne normée des distances dans une sous-fenêtre de la zone de test.

On associe un descripteur M_{r_i} à chaques sous-fenêtre, défini par :

$$M_{r_i} = \frac{1}{n} * \sum_{d \in r_i} \begin{cases} S(d,z) & \text{si } d \neq 0 \\ 0 & \text{si } d = 0 \end{cases}$$

$$\text{avec : } S(d,z) = \min(\max(\frac{f*b}{d} - z, -\Delta z_{max}), \Delta z_{max}) \quad (1)$$

Où n est le nombre de pixels dans r_i dont la valeur est différente de 0, f est la focale des caméras et b est l'écartement entre caméras. Dans l'équation précédente, la disparité d est convertie en distance par la relation :

$$l = \frac{f*b}{d} \quad (2)$$

M_{r_i} définit la moyenne des écarts de profondeur, pour tous les point de la sous-fenêtre entre la mesure de disparité et la distance z de test ; cette mesure d'écart étant calculée par la fonction $S(d,z)$.
Les pixels qui ont une disparité égale à 0 sont les pixels filtrés ou à l'infini. Dans le cas où toute la zone testée est à l'infini ou filtrée, c'est à dire si $n = 0$, alors $M_{r_i} = z_{max}$.
La détection d'obstacles sans saturation ($\Delta z_{max} = \infty$) nécessite moins de temps de calcul car dans l'équation 1 la distance z peut être factorisée.

$$M_{r_i} = \left(\frac{1}{n} * \sum_{d \in r_i} \begin{cases} \frac{f*b}{d} & \text{si } d \neq 0 \\ 0 & \text{si } d = 0 \end{cases} \right) - z \quad (3)$$

2.4 Accélération du calcul par Image intégrale

Le descripteur proposé peut se calculer de manière rapide à partir d'une image intégrale. Un tel principe a déjà été utilisé dans [10] pour le calcul rapide d'ondelettes de Haar ou plus récemment dans [9] pour le calcul rapide de matrices de covariances ou dans [8] pour le calcul d'histogrammes.
La valeur de l'image intégrale, nommée $ii(x,y)$ à la position (x,y) est la somme de toutes les valeurs des pixels de l'image originale, au-dessus et à la gauche de (x,y) (voir Figure 3(1)) :

$$ii(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x',y') \quad (4)$$

Avec $i(x,y)$ la valeur du pixel dans l'image de disparité à la position (x,y). L'image intégrale permet de calculer en temps constant la somme des valeurs d'une zone de l'image. Comme par exemple dans la Figure 3(2), où la somme des valeurs de la zone r_i est égale à : $ii(L_4) - ii(L_2) - ii(L_3) + ii(L_1)$.

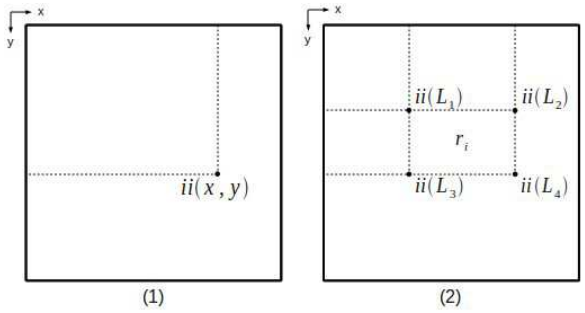


FIGURE 3 – Représentation d'une image intégrale. (1) Image intégrale. (2) La région r_i peut être calculée en utilisant la valeur des ses quatre coins : $ii(L_4) - ii(L_2) - ii(L_3) + ii(L_1)$

Dans le cas de l'équation 1, il faut une image intégrale pour chaque distance z et une image intégrale pour le nombre de points tel que $d \neq 0$. On a donc :

$$ii_z(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} \begin{cases} S(i(x',y'),z) & \text{si } i(x',y') \neq 0 \\ 0 & \text{si } i(x',y') = 0 \end{cases}$$

$$ii_n(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} \begin{cases} 1 & \text{si } i(x',y') \neq 0 \\ 0 & \text{si } i(x',y') = 0 \end{cases} \quad (5)$$

Soient L_1, L_2, L_3, L_4 les coordonnées (x,y) des quatre coins de la zone r_i, alors :

$$M_{r_i} = \frac{ii_z(L_4) - ii_z(L_2) - ii_z(L_3) + ii_z(L_1)}{ii_n(L_4) - ii_n(L_2) - ii_n(L_3) + ii_n(L_1)} \quad (6)$$

Dans le cas $\Delta z_{max} = \infty$, seulement deux images intégrales sont nécessaires.

$$\begin{aligned} ii(x,y) &= \sum_{\substack{x' \leq x \\ y' \leq y}} \begin{cases} \frac{f*b}{i(x',y')} & \text{si } i(x',y') \neq 0 \\ 0 & \text{si } i(x',y') = 0 \end{cases} \\ ii_n(x,y) &= \sum_{\substack{x' \leq x \\ y' \leq y}} \begin{cases} 1 & \text{si } i(x',y') \neq 0 \\ 0 & \text{si } i(x',y') = 0 \end{cases} \end{aligned} \quad (7)$$

Alors l'équation 3 donne :

$$M_{r_i} = \frac{ii(L_4) - ii(L_2) - ii(L_3) + ii(L_1)}{ii_n(L_4) - ii_n(L_2) - ii_n(L_3) + ii_n(L_1)} - z \quad (8)$$

2.5 Apprentissage statistique

Le nombre de sous-fenêtres possibles pour une zone candidate est très important. Il faut donc appliquer un algorithme de sélection de sous-fenêtres pour retenir uniquement les plus pertinentes. Ce dernier consiste à déterminer l'ensemble $\mathcal{R} \doteq \{r_i\}_{i=1,..,I}$. Nous proposons d'utiliser l'algorithme Adaboost pour sélectionner cet ensemble.
Un classifieur faible de type *Decision stump* [7] est utilisé par Adaboost. Ce classifieur est un arbre de décision binaire à un noeud et deux feuilles. La valeur d'entrée du noeud est comparée à un seuil. Le seuil est déterminé par l'algorithme Adaboost durant l'apprentissage.
Adaboost est entraîné à l'aide d'une base d'apprentissage constituée d'un ensemble d'exemples positifs et négatifs (Figure 4) d'images de disparité. Ces exemples permettent à l'algorithme de choisir parmi toutes les sous-fenêtres $r_{1,2,...}$ de R un ensemble de descripteurs. Nous considérons toutes les sous-fenêtres r à partir d'une taille de $1/8$ de la largeur et $1/16$ de la hauteur de la zone R. La taille de r est multipliée par 2 en horizontal, en vertical ou les deux, jusqu'à que $r = R$.

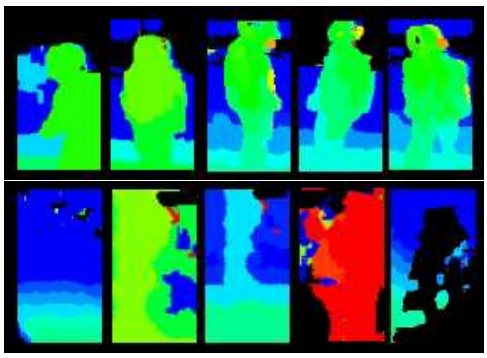
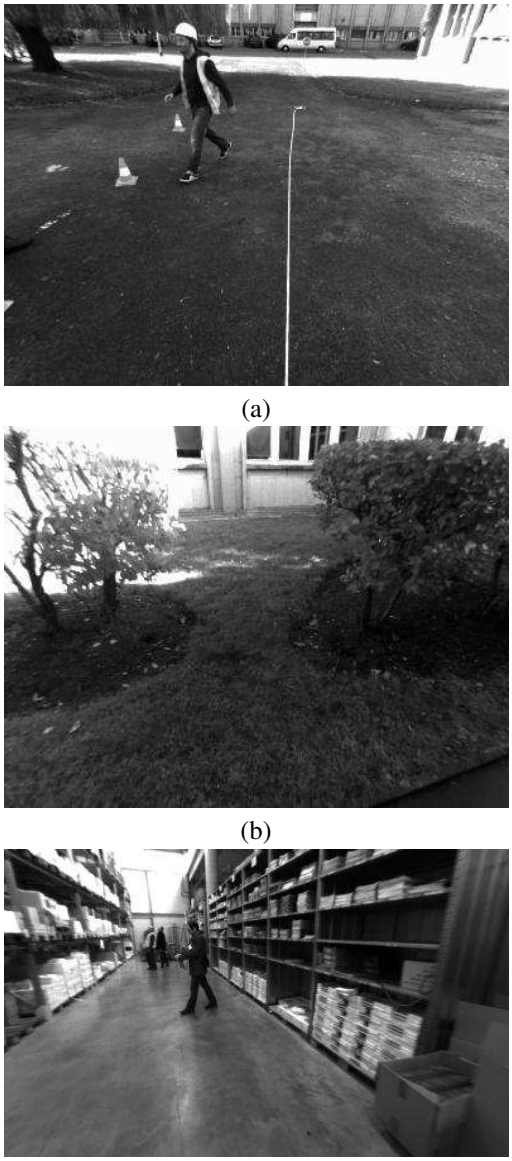


FIGURE 4 – Exemples positifs (haut) et exemples négatifs (bas). Rouge : $z = \Delta z_{max}$, Vert : $z = 0$, Bleu : $z = \Delta z_{max}$, Noir : $z = \infty$ ou filtré. Ces exemples sont utilisés par l'algorithme Adaboost pour sélectionner les sous-fenêtres les plus pertinentes

3 Expérimentations

Cette section présente les données utilisées et les différents tests effectués pour évaluer l'algorithme de détection de piétons proposé.
Un apprentissage par Adaboost a été réalisé à partir d'exemples labellisés issus de deux séquences. Une séquence positive (voir Figure 5(a)) qui contient approximativement 4000 exemples de piétons. Une séquence négative (voir Figure 5(b)), sans piétons, dans laquelle nous avons extrait 10000 imagettes négatives aléatoirement. On peut noter que les deux vidéos utilisées pour l'apprentissage représentent des scènes en extérieur. Pour la vidéo de test (voir Figure 5(c)), nous avons utilisé une vidéo de 1950 images en intérieur dans les allées d'un entrepôt.



(a)

(b)

(c)

FIGURE 5 – Exemples d'image des vidéos : (a) vidéo positive. (b) vidéo négative. (c) vidéo de test.

Les calculs ont été réalisés sur une machine de type intel Xeon 5160 à $3GHz$. Les objets sont recherchés dans l'image suivant une distance comprise entre $1m$ et $5m$ pour un pas de $10cm$. Leur taille supposée est comprise entre $1.5m$ et $1.9m$. Le nombre total de boîtes testées par les détecteurs d'obstacles est de 29737. Nous avons imposé à l'algorithme d'apprentissage de caractériser le modèle piéton avec 30 descripteurs (voir Figure 6). Pour la fusion des boîtes, nous avons imposé le regroupement des boîtes qui ont une distance de moins de $50cm$.

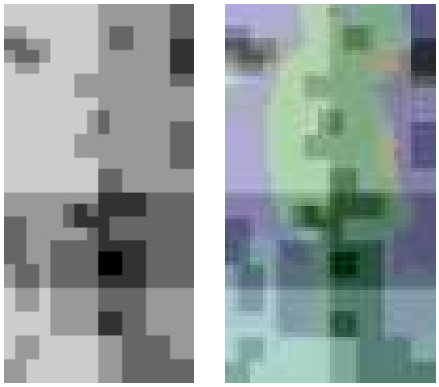


FIGURE 6 – Exemple de la répartition spatiale des 30 descripteurs issus de la phase d'apprentissage caractérisant la forme d'un piéton (à gauche) et son application sur un exemple de piéton (à droite). Les zones sombres représentent les zones sélectionnées par plusieurs descripteurs.

Dans tous les tests, la méthode proposée (nommée AdaboostDisparité par la suite) est comparée à un algorithme classique de détection d'obstacles verticaux (nommé DetectObs par la suite). Ce dernier consiste à évaluer, suivant le plan du sol, le pourcentage de pixels d'une zone verticale cohérente (voir Figure 7). Si le taux de remplissage est supérieur à un seuil alors la boîte est considérée comme un obstacle à classifier. Les boîtes contenant un obstacle sont ensuite fusionnées par le même algorithme et avec les mêmes paramètres que pour la méthode proposée. Enfin, un algorithme basé sur l'apparence est appliqué à la suite de la détection utilisant la carte de disparité.

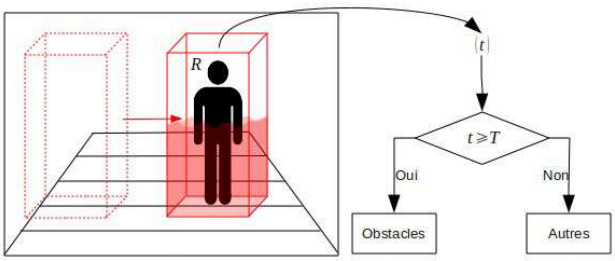


FIGURE 7 – Principe de la méthode de recherche d'obstacles suivant le plan du sol avec seuillage du taux de remplissage.

3.1 Courbes ROC

La figure 8 montre les courbes *Receiver Operating Characteristics* (ROC) comparant la méthode AdaboostDisparité avec la méthode DetectObs sur la vidéo de test. Elles représentent le pourcentage de piétons correctement détectés en fonction du pourcentage de fausses détections dans la vidéo test. Les courbes ont été calculées sur un échantillon de $29737 * 1950 = 57987150$ boîtes testées alors que le nombre de piétons présents dans la vidéo a été de 1570. La courbe ROC rouge représente la méthode DetectObs où 70% du nombre de piétons est correctement détecté, mais avec une erreur de $0,00024\%$ soit une moyenne de 6 fausses détections par images. La méthode AdaboostDisparité (avec $\Delta z_{max} = \infty$) représentée par la courbe ROC bleue détecte 80% des piétons avec une erreur de $0,00007\%$ soit une moyenne de 2 fausses détections par images. La courbe verte représente la méthode AdaboostDisparité avec $\Delta z_{max} = 2m$, les résultats obtenus sont similaires au cas $\Delta z_{max} = \infty$. Ce résultat confirme que la méthode AdaboostDisparité est plus performante que la méthode DetectObs.

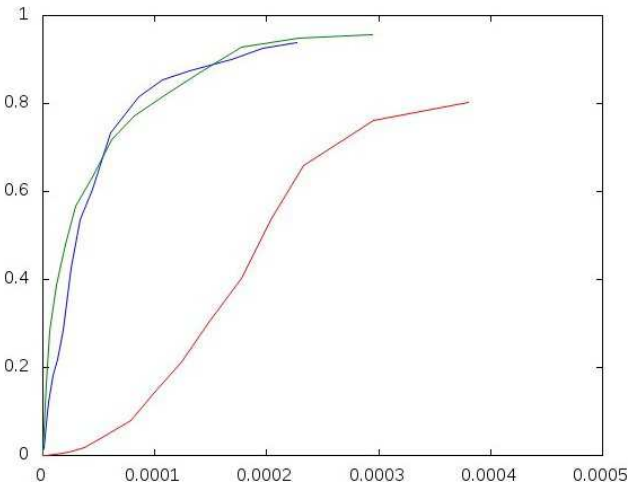


FIGURE 8 – Courbes ROC de la détection de forme. En vert, AdaboostDisparité avec $\Delta z_{max} = 2m$. En bleu, AdaboostDisparité avec $\Delta z_{max} = \infty$. En rouge, DetectObs.

La figure 9 montre quelques exemples de détection réalisées à l'aide de la méthode AdaboostDisparité et de la méthode DetectObs. Les rectangles verts et jaunes représentent les détections de la méthode DetectObs. Les boîtes rouges représentent les détections de la méthode AdaboostDisparité.

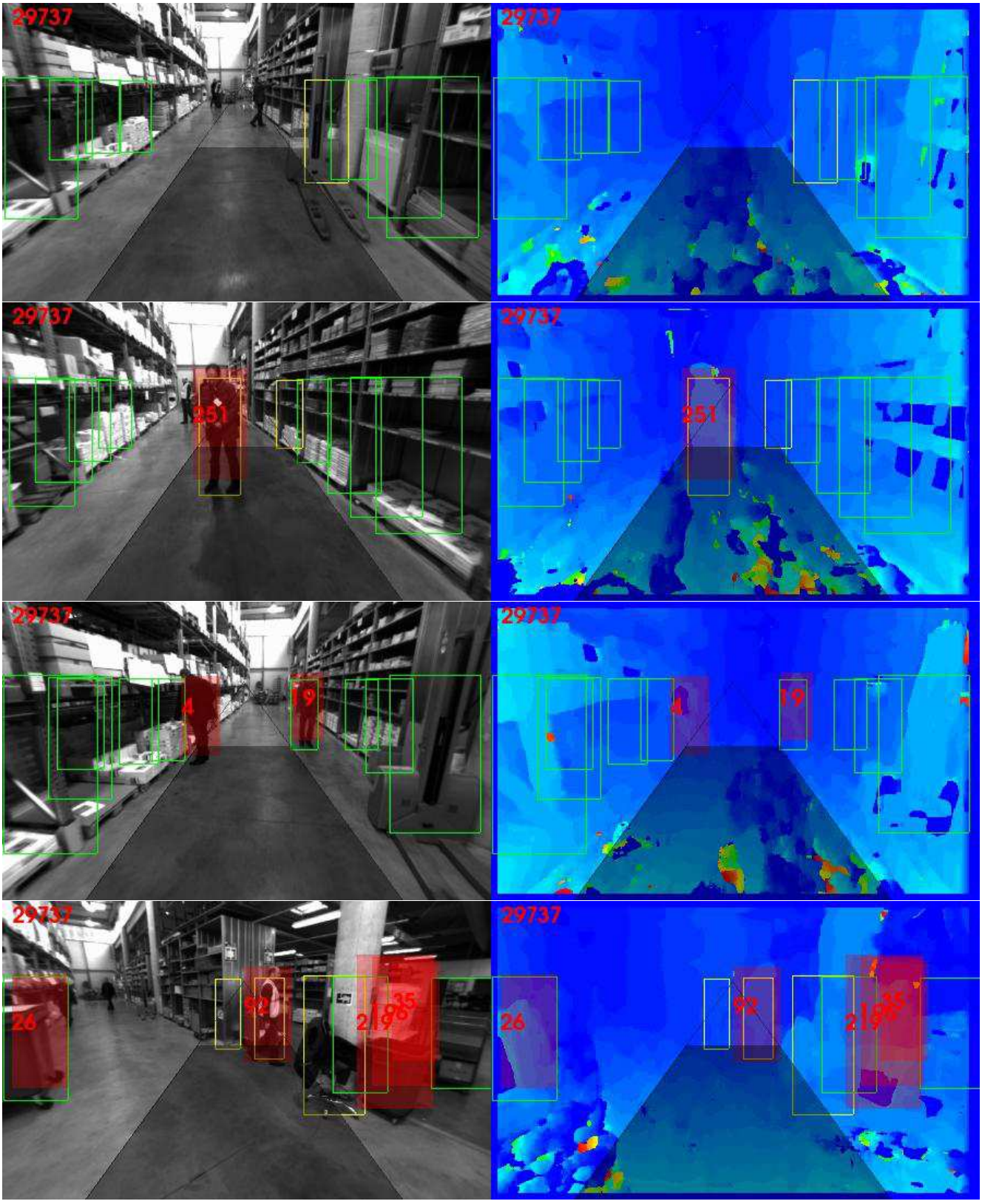


FIGURE 9 – Exemples de détection de la méthode DetectObs (rectangles verts et jaunes) et de la méthode AdaboostDisparité (boîtes rouges).

3.2 Temps de calcul

Le tableau 1 compare les temps de calcul de la méthode AdaboostDisparité avec la méthode DetectObs. Dans le cas de $\Delta z_{max} = 2m$, le calcul des 42 images intégrales limite les performances et permet à notre méthode de détecter les personnes en $130ms$. A l'inverse, la méthode n'ayant besoin que de seulement 2 images intégrales pour détecter les personnes dans le cas où $\Delta z_{max} = \infty$, son temps de calcul est $60ms$. La détection d'obstacles classique (DetectObs) utilisée est plus optimisée et effectue sa détection en seulement $10ms$. Cette différence de temps de calcul est compensée par la différence de performance de détection, c'est à dire le nombre de boîte à classifier.

TABLE 1 – Temps moyen de calcul par image.

Algorithme	Temps de calcul moyen par image
AdaboostDisparité avec $\Delta z_{max} = 2m$	130 ms
AdaboostDisparité avec $\Delta z_{max} = \infty$	60 ms
DetectObs	10 ms

4 Conclusion

Dans cet article, nous avons proposé et présenté une méthode de détection de piétons, dans la carte de disparité, grâce à un apprentissage par Adaboost. Nous avons proposé un descripteur simple et robuste, utilisant les données de la carte de disparité. Les performances de détection ont été caractérisées et comparées à une méthode de détection d'obstacles sur une carte de disparité. Nous avons travaillé pour améliorer la détection d'obstacles en caractérisant la forme d'un piéton grâce à un algorithme de boosting et des descriteurs sur la carte de disparité. Ce type de détecteur permet une amélioration du temps de classification dans des environnements encombrés. Ces environnements génèrent en effet une grande quantité de zones à tester par des algorithmes de détection d'obstacles classique.

L'emploi d'une cascade de classification doit permettre de réduire le temps de calcul. D'autre part, la stéréo vision limite la qualité de la carte de profondeur. L'utilisation de capteur 3D plus précis du genre Kinect rend possible l'utilisation de descripteurs plus performants que la simple moyenne de la profondeur (par exemple ceux de [9]).

Références

[1] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, L. H. Matthies, *A Fast Stereo-based System for Detecting and Tracking Pedestrians from a Moving Vehicle*, The International Journal of Robotics, 2009

[2] A. Ess, K. Schindler, B. Leibe, L. van Gool, *Improved Multi-Person Tracking with Active Occlusion Handling*, Proceedings of the IEEE ICRA 2009, 2009

[3] Y. Freund, R. E. Schapire, *A Short Introduction to Boosting*, Journal of Japanese Society for Artificial Intelligence, Vol. 14, pp. 771-780, 1999

[4] S. Gidel, P. Checcin, C. Blanc, T. Chateau, L. Trassoudaine, *Pedestrian Detection Method using a Multilayer Laserscanner : Application in Urban Environment*, IEEE Transactions on Intelligent Transportation Systems, 2010

[5] H. Hattori, A. Seki, M. Nishiyama, T. Watanabe, *Stereo-based Pedestrian Detection using Multiple Patterns*, 2009

[6] L. J. Heyer, S. Kruglyak, S. Yooseph, *Exploring expression data : Identification and analysis of coexpressed genes*, Genome Research, Vol. 9, pp. 1106–1115, 1999

[7] W. Iba, P. Langley, *Induction of One-Level Decision Trees*, Proceedings of the Ninth International Conference on Machine Learning, 1992

[8] F. Porikli, *Integral Histogram : A Fast Way to Extract Histograms in Cartesian Spaces* IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 1, pp. 829-836, June 2005

[9] O. Tuzel, F. Porikli, P. Meer, *Pedestrian Detection via Classification on Riemannian Manifolds*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, pp. 1713-1727, 2008

[10] P. Viola, M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features*, Computer Vision and Vattern Recognition, 2001