HETEROGENEOUS ADABOOST WITH REAL-TIME CONSTRAINTS

Application to the Detection of Pedestrians by stereovision

Loïc Jourdheuil[1], Nicolas Allezard[1], Thierry Chateau[2] and Thierry Chesnais[1]

[1]*CEA, LIST, Laboratoire Vision et Ingénierie des Contenus, Gif-sur-Yvette, France*

[2]*LASMEA, UMR UBP-CNRS 6602, 24 Avenue des Landais, AUBIERE, France*

{*loic.jourdheuil, nicolas.allezard, thierry.chesnais*}*@cea.fr, thierry.chateau@lasmea.univ-bpclermont.fr*

Keywords: Adaboost. stereovision. real time.

Abstract: This paper presents a learning based method for pedestrians detection, combining appearance and depth map descriptors. Recent works have presented the added value of this combination. We propose two contributions: 1) a comparative study of various depth descriptors including a fast descriptor based on average depth in a sub-window of the tested area and 2) an adaptation of the Adaboost algorithm in order to handle heterogeneous descriptors in terms of computational cost. Our goal is to build a detector balancing detection rate and execution time. We show the relevance of the proposed algorithm on real video data.

1 Introduction

The pedestrian classification is one of the most requested tools in the video surveillance field. Some specific solutions exist in the context of video acquired with stationary cameras. In this case, image features from the spatial and temporal domains are fused in order to jointly learn the correlation between appearance and foreground information based on background subtraction (Yoa and Odobez, 2008). Some other works (Dalal et al., 2006), (Wojek et al., 2009) have used the fusion of the appearance and movement in the image in order to improve results.

The method described in this article can not use these combinations because its implementation field is linked with the use of cameras embedded on moving vehicles. Issues linked with this class of application are the reliability and the computing time of the detector. During the last years, the community has paid attention to depth map information coming from sensors using stereovision (Walk et al., 2010), (Enzweiler and Gavrila, 2011), (Ess et al., 2008), in order to have an efficient classification. We will present some of these descriptors and their performances when they are used by an Adaboost learning algorithm.

Works (Walk et al., 2010) combining both appearance and disparity descriptors have been also published recently. However, the fusion of several descriptors often increases the processing time. In addition, this difference of computation time is never included in the learning phase of the detector while the objectives of the application are to achieve a real-time solution. We propose a modification of Adaboost algorithm by introducing a penalty linked with algorithmic complexity of each descriptor.

In a first part, we present different weak classifiers based on depth information. Then we compare them with the Histogram of Oriented Gradient (HOG) method based on the image luminance. In a second part, we will present an innovative learning method, called Heterogeneous Adaboost (HAB), designed to merge several detectors, by taking into account the computing time of each algorithm.

2 Depth descriptors

In this first section four descriptors of the depth map are presented. The first three are based on local deviations of depth. The last one is calculated on the depth itself.

In the Figure 1, the left image shows a pedestrian moving in an industrial environment. The pseudo image on the right of the figure shows the representation of the associated depth map.

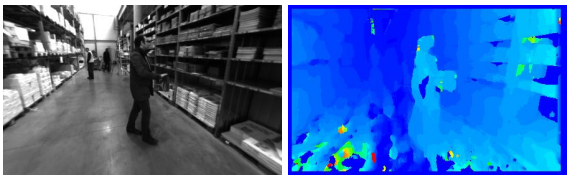


Figure 1: Example image (left) and the associated depth map (right).

The Figure 2 shows, in its upper part, a set of depth maps for pedestrians (positive examples)and its lower part examples of depth maps of the environment (negative examples). The creation of this map is made by a conventional method for dense depth map in real time, not presented in this article.

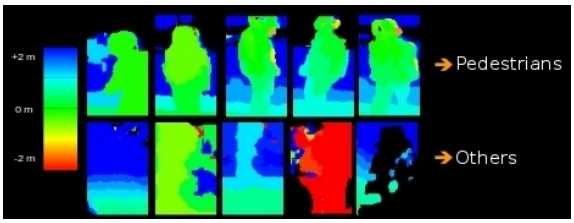


Figure 2: Positive examples (upper) and negative examples (lower) in the depth map.

A sliding window strategy (with a ratio of width to height equals $\frac{1}{2}$) covering the ground level, is used to scan the depth image (Figure 3). For each window, a feature vector composed a of set of local descriptors from the depth map, is computed. Then vectors are evaluated following a model created during off-line learning stage.

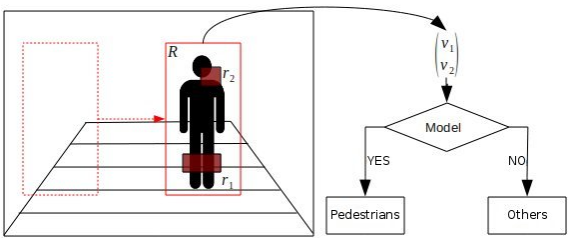


Figure 3: Evaluation scheme of pedestrian classification.

2.1 Comparison of descriptors

The use of depth map-based descriptors is recent. We have adapted well-known efficient descriptors of appearance to be used on the depth map. We have also created a specific descriptor of the depth map. We present four depth map descriptors with HOG as a reference method on real images.

2.1.1 HOG-depth

This descriptor describes the depth map with as a Histogram of Oriented Gradient (HOG) of depth. This descriptor type is typically used to represent images in appearance (Dalal and Triggs, 2005) generally. The version used for the study, has been enriched with the gradient magnitude, (Begard et al., 2008): a normalized histogram of nine magnitudes of orientation (from 0° to 180°) is built from the map of depth gradient. Then this histogram is enriched with a tenth value: the gradient magnitude.

2.1.2 Covariance matrix (MatCov)

This descriptor, derived from the work of Tuzel (Tuzel et al., 2008) offers a coding pedestrians from the covariance matrices of appearance and its spatial derivative, calculated on sub-windows of the scan. Our proposition is the same as the previous one but using information from the depth map with the data $[x\ y\ d]$ where x and y are the coordinates of pixel and d the depth value of this pixel. Then the covariance matrix is in a connected Riemannian manifold $\mathcal{M}_{3\times3}$.

$$f_{cov} : \chi \rightarrow \mathcal{M}_{3\times3} \quad (1)$$

Where $\mathcal{M}_{3\times3}$ is in $\mathbb{R}^{3\times3}$. A function $\psi : \mathcal{M}_{3\times3} \rightarrow \mathbb{R}^6$ projects the manifold to the tangent space using the formula:

$$\psi(X) = vec_\mu(log_\mu(X)) \quad (2)$$

Where X and μ are two positive, symmetric matrices. With:

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

And :

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

μ is the point of the weighted average of covariance matrices, coming from the formula:

$$\mu = arg \min_{Y \in \mathcal{M}_{3\times3}} \sum_{i=1}^{N} d^2(X_i, Y) \quad (5)$$

So an essential metric for the comparison of two covariance matrices is defined.

2.1.3 Covariance, Depth Variance (CovVar)

The previous descriptor (MatCov) has the drawback of being quite heavy in terms of computation time. So one simplified version, consisting of the definition of a distance which does not take any more account of properties of the Riemannian manifold, has been used. In the covariance matrix $3*3$ created with the data $[x\ y\ d]$, the first two columns are constant. Thus, the matrix is reduced at its last column and we gets the vector $[cov(x,d)\ cov(y,d)\ var(d)]$. So the norm $L2$ to define the distance between two descriptors can be used by this reduction.

2.1.4 Average

We propose a simple descriptor based on the average of the differences between the test depth d and measured depth z in a sub-area of the window of analysis. This descriptor called M_{r_i} is associated with the region r_i, and it is defined by the following equation:

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

Where n is the number of defined depth points z. r_i regions sizes and positions are defined during the learning phase.

This average depth-based descriptor may remember the descriptor *DispStat* (Walk et al., 2010) using a fixed cutting up of the disparity map.

2.2 Performance comparison

The performance comparison of different descriptors was done by using a learning machine type Adaptive Boosting algorithm (Freund and Schapire, 1999) and a decision stump classifier type for HOG-depth, CovVar, Average, and HOG on appearance. In the case of a learning phase with MatCov, the weak classifier is a linear regression done on the components of the symmetric matrix.

2.2.1 Assessment protocol

Learning has been done on a video shot inside a warehouse where $10,000$ images of negative examples and $1,500$ images of positive examples have been extracted. Each tested area R is divided into $1,482$ rectangular regions r_i.

In boosting methods, a new weak classifier is chosen at each round of learning. Then n chosen weak classifiers are combined in a new strong classifier. Each strong classifier is evaluated on $10,000$ images of negative examples and $1,500$ images of positive examples taken from a test video shot in a different warehouse from the learning one. Following tests done, the maximum of performance on the test database can be achieved through a strong classifier, created after the one which has reduced the learning error to 0 in the learning phase. As about 100 rounds of Adaboost are required by classifiers to reduce the learning error to 0 on the learning database, we have chosen to execute 300 rounds on the learning database to maximize the probability of regulation of the maximum performance on the test database. The 300 Receiver Operating Characteristic (ROC) associated with each classifier are compared and the best curve is selected, according to the criterion:

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

Thus, each classifiers type can be evaluated at its maximum performance regardless of the number of rounds completed. A similar test of the HOG descriptor on the appearance will be the reference curve.

2.2.2 Results

The figure 4 shows ROC curves obtained by using the previous protocol. The curves have been compared at a detection rate of 90% to the reference HOG curve (red). HOG-Depth curve (green) has a false positive rate of 15% or 11% above the baseline (4%). Strong classifiers selected of curves HOG-Depth and HOG have been learned with a respective number of rounds of 270 and 290. The false positive rate of the curve MatCov (purple) is 8% (120 rounds), the curve CovVar (dark blue) is 10% (208 rounds), respectively 4% and 6% higher than the reference. The percentage of false positives of the Average curve (light blue) is close to 4% (212 rounds) similar to the reference.

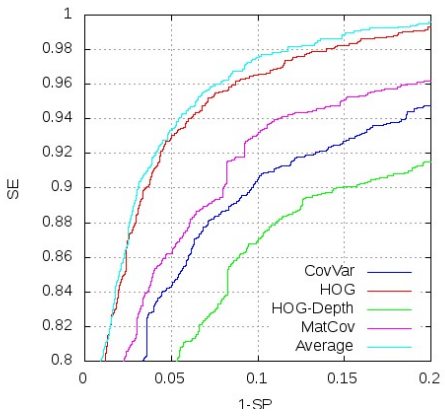


Figure 4: Comparison of detection performance of weak classifiers on the depth map (HOG is the reference).

The results obtained using our Average classifier created specifically to work on the depth map are as good as the ones obtained using HOG appearance. That confirms our initial hypothesis: the depth map is sufficient to classify pedestrians. These encouraging results on the use of the depth map for the pedestrian classification, open a new way of merger with a appearance descriptor.

3 Heterogeneous Adaboost

The work (Walk et al., 2010) of Walker et al. has showed that the fusion of descriptors of appearance, motion and disparity improves significantly the classification results. However, this one generates an increase in computing time which can be a limited factor not acceptable for real-time application.

In this second section, we propose to take account of a constraint of algorithms cost, softened by $\alpha \in [0,1]$, in the Adaboost algorithm (Friedman et al., 1998) (Heterogeneous Adaboost , HAB). It will be selection criteria of heterogeneous descriptors to create a strong and fast classifier. Also, the depth descriptors presented in the previous section, will be combined with HOG appearance in a global learning phase.

Following the results of the previous study, the method CovVar will be preferred to MatCov method because the change of the method MatCov by CovVar method damages a little bit results while reducing the computation time.

3.1 Algorithm HAB

Let C be the set of weak learning algorithms to Adaboost. We define an algorithm as a component descriptors. We associate to each algorithm, its algorithmic cost λ_c. The algorithmic cost can be estimated from the complexity of the descriptor, or measured statistically on a target machine.

One algorithmic cost softened normalized $\tilde{\lambda}_c$, enabling management of the influence of algorithmic cost in the calculation of the pseudo error penalized, is calculated using the equation:

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

where $\alpha \in [0;1]$ is a softening coefficient chosen empirically.

In the classical AdaBoost, the criterion to minimize is calculated as follows:

$$\varepsilon_t^c = E[e^{-y\tilde{h}_t^c}] \tag{9}$$

With $\tilde{h}_t^c : \mathbf{x} \rightarrow \{-1;1\}$ the response of the weak classifier. We propose to penalize the criterion to minimize ε_t^c by the algorithmic cost $\tilde{\lambda}_c$ in HAB. So the new equation of the criterion to minimize (penalized) becomes:

$$\tilde{\varepsilon}_t^c = \tilde{\lambda}_c \, E[e^{-y\tilde{h}_t^c}] \tag{10}$$

When one component of a descriptor is chosen by the algorithm, other related components of the same descriptor are calculated simultaneously. Therefore, the algorithmic cost λ_c of all components of descriptors already calculated change and should be updated. So we introduced the algorithmic cost λ_0 of classification without calculation.

For each component c already calculated, λ_c is replaced by the algorithmic cost *Empty* λ_0. The new normalized softened algorithmic cost $\tilde{\lambda}_c$ is recalculated.

Algorithm 1 on page 8 describes the different stages of Heterogeneous Adaboost.

3.2 Algorithmic cost

The evaluation of the complexity of the descriptor and the statistical measure of processing time on a target machine are among the possibilities to estimate the cost algorithm λ_c of weak classifiers $c \in C$. We chose to measure the average time (ns) of calculation required to calculate a weak classifier. The figure 5 represents algorithmic costs[1] for methods CovVar, HOG, HOG-depth and Average, and the algorithmic cost (called Empty) cited in the HAB algorithm, evaluated by a classifier that returns an empty vector.

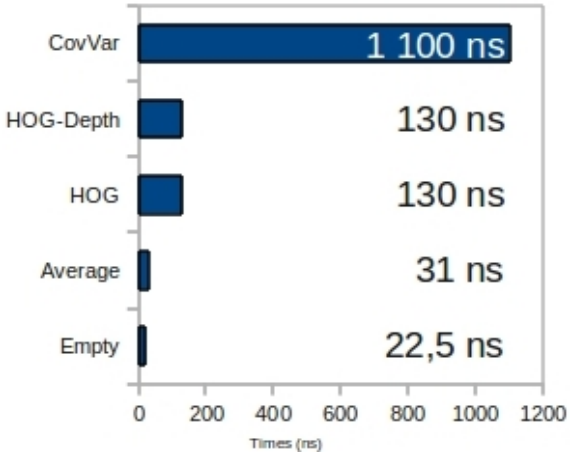


Figure 5: Algorithmic cost λ_c for methods CovVar, HOG, HOG-depth and Average and empty cases in nanoseconds (ns) per weak classifier.

The algorithmic cost varies from $1,100ns$ for CovVar to $31ns$ for the Average classifier which is close to $22.5ns$ of the classifier empty. This classifier is also four times faster than the reference HOG and its derivative HOG-Depth ($130ns$).

We define the algorithmic cost Λ_t of the strong classifier output of the rounds t of HAB following equation:

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

where λ_c is the algorithmic cost of the weak classifier selected round t.

[1] All tests were performed on a PC computer equipped with a $2.8GHz$ processor

3.3 Choice of the softening coefficient α

The magnitude of the algorithmic cost $\tilde{\lambda}_c$ greatly influenced the choice of the weak classifier in the HAB algorithm. To manage this influence, a softening coefficient α is applied (equation 8). The latter is chosen accordingly to the learning error and the computing time Λ_t.

In this section, we will present the influence and the choice of a coefficient α for HAB with HOG+Average descriptor.

3.3.1 Learning error

The AdaBoost algorithm minimizes the learning error for each of his rounds. The figure 6 shows learning errors ε_t based on the number of rounds t for classical Adaboost (red) and the HAB algorithm (blue). As we can see, the learning error of Adaboost is less the HAB error.

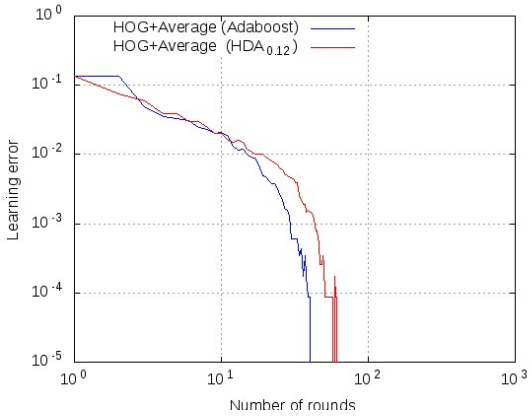


Figure 6: Learning errors based on the number of rounds for HOG+Average learning with classical Adaboost and HAB.

In the figure 7, we represented learning errors based on the algorithmic cost of strong classifiers Λ_t (equation 11) to account for the algorithmic cost. In this case, algorithm HAB (blue) reduces the learning error faster than classical Adaboost (red) regardless of the algorithmic cost (Time).

3.3.2 Determination of the coefficient α

The experiments have showed that the algorithmic cost influences strongly the selection of weak classifiers. We have looked for to manage its influence by assigning a softening coefficient α (equation 8) to it. The figure 8 shows learning errors based on the algorithmic cost (time) for a representative sample of possible values $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$. In

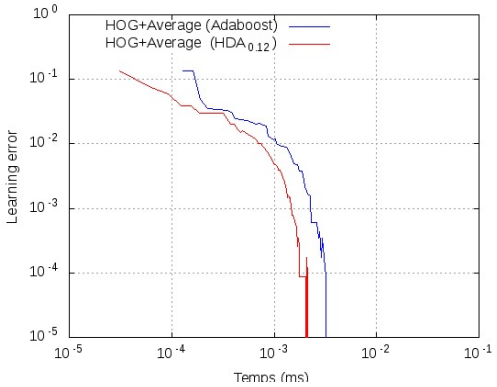


Figure 7: Learning errors based on the number of computing time (ms) for learning HOG+Average with classical Adaboost and HAB.

this figure, the curve α_1 (black) represents the learning error without softening coefficient. The percentage error of the learning curve has reached only 5% error at the end of 300 rounds of learning. The curve α_0 (dark blue) represents the evolution of the learning error without algorithmic cost (softening maximum, similar to a classical AdaBoost). The other curves are related to values of the coefficient $\alpha \in [0; 1]$. The value of this coefficient has a very significant effect on the performance of the algorithm. The curve $\alpha_{0.12}$ (green) appears to be a good compromise between the decrease of the learning error and computing time.

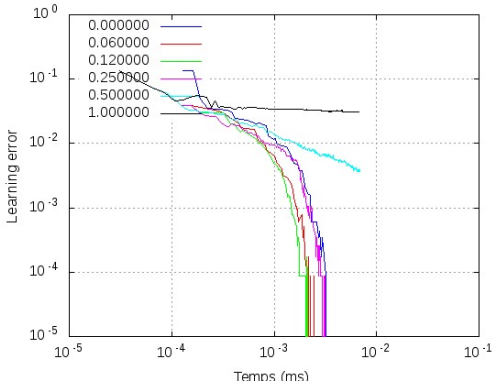


Figure 8: Learning errors for HOG+Average method based computation time (ms) for $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$

To verify the effectiveness of the curve $\alpha_{0.12}$, we used the data of the areas of ROC curves of the protocol section 2.2.1 (page 3) that we posted on a algorithmic cost Λ_t. The figure 9 shows the 1-surface of ROC curves as a function of time for $\alpha \in \{0; 0.06; 0.12; 0.25; 0.50; 1\}$.

It confirms the performance of the curve $\alpha_{0.12}$ (green).

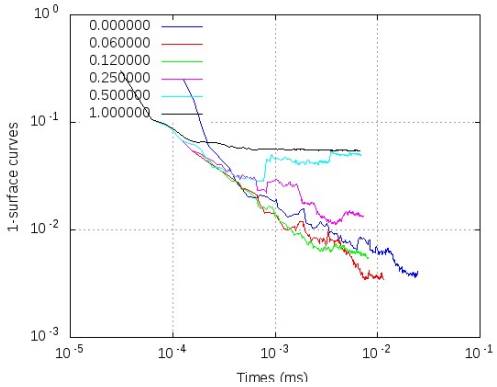


Figure 9: 1-surface of ROC curves for HOG+Average method based computation time (ms) for $\alpha \in \{0;0.06;0.12;0.25;0.50;1\}$.

3.4 Fixed computation time

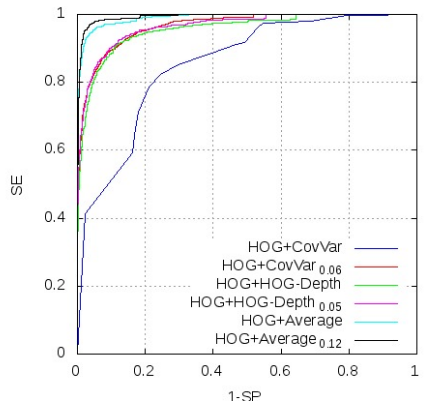
This section presents the experiments conducted on the test dataset, to compare method performance with those of HAB on the three classical Adaboost weak classifiers HOG+Average, HOG+CovVar et HOG+HOG-Depth. The protocol used is identical to the one presented in section 2.2.1 (page 3). A maximum time limit *lim* for the time variable Λ_t is defined to reflect the real-time constraint.

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

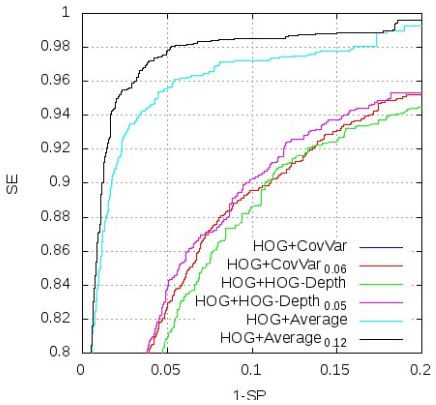
The time limit *lim* is set to $2.5\mu s$ and $1.25\mu s$ corresponding respectively to 10 and 20 frames per second. This time limit has been estimated to account for a number of $40,000$ tests sub-window to evaluate an image. Figure 10 (respectively 11) shows six ROC curves corresponding to $lim = 2.5\mu s$ (respectively $lim = 1.25\mu s$). Part (a) represent the full curves in the interval $[0;1] \times [0;1]$, while part (b) are zooms in the interval $[0;0.2] \times [0.8;1]$.

In both figures, curves HOG+Average (light blue), HOG+CovVar (dark blue) and HOG+HOG-Depth (green) are learned with a classical Adaboost. Others curves are learned with HAB and a coefficient α determined accordingly to the protocol presented above. The coefficient α is equal 0.12 for HOG+Average (black), 0.06 for HOG+CovVar (red) and 0.05 for HOG+HOG-Depth (purple). In both figures, the curves generated by HAB present a detection rate (SE) better than their counterparts produced by classical Adaboost regardless of the false positive rate (1-SP).

On the other hand, whatever the selected time, the best strong classifier is given by HOG+Average learning (black).



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



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

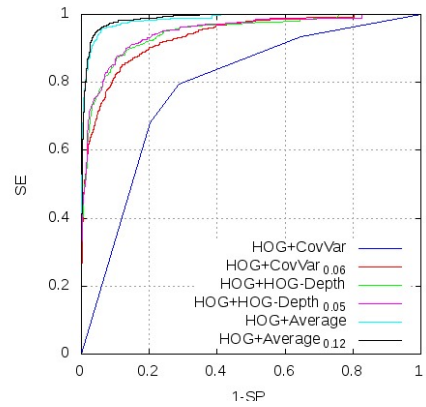
Figure 10: ROC curves for comparison between a AdaBoost and HAB for a computing time $\Lambda_t \leqslant 2.5\mu s$.

The figure 4 (page 3), showed that the CovVar method was more efficient than HOG-Depth. However, the CovVar algorithmic cost is ten times higher than the HOG-Depth (figure 5). Figures 10 and 11 show that the merger by HAB leads to a combination HOG+HOG-Depth that outperforms HOG+CovVar.

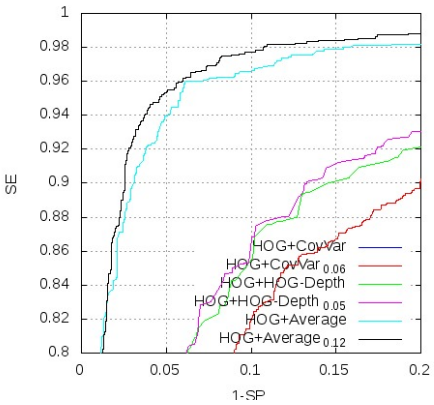
4 Conclusion and prospects

In this paper, in a first part, we have proposed and presented different descriptors of the depth map. Following of the tests results the Average descriptor provides performance equivalent to HOG descriptor which is our reference.

In a second part, we have proposed a change of the Adaboost algorithm taking account of the algorithmic cost λ_c for the selection of each weak classifier. This new algorithm (HAB) has been evaluated on the fusion of a appearance descriptor (HOG) with de-



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



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

Figure 11: ROC curves for comparison between a AdaBoost and HAB for a computing time $\Lambda_t \leqslant 1.25\mu s$.

scriptors of the depth map (CovVar, HOG-depth and Average). The ROC curve which has a fixed processing time, is superior to the conventional Adaboost approach. The output of HAB algorithm evaluates Algorithms cost of strong classifier Λ_t also.

The HAB algorithm could be adapted to the use of a cascade (Viola and Jones, 2001) by setting a processing time of each floor. In addition, the computational cost of each classifier can be redefined at each stage in order to favour slow but efficient detectors at the end of the cascade.

REFERENCES

Begard, J., Allezard, N., and Sayd, P. (2008). 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.

Dalal, N. and Triggs, B. (2005). 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.

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

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

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

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

Friedman, J., Hastie, T., and Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Statistics, Stanford University Technical Report*.

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

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition*.

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

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

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

Algorithm 1 Heterogeneous Adaboost

Require: N a set of labelled examples $\{(\mathbf{x}_n, y_n)\}_{n=1,..,N}$, D a probability distribution associated with N examples, C a set of learning algorithms with low computational cost associated $\{(\texttt{WeakLearn}_c, \lambda_c)\}_{c=1,..,C}$, λ_0 algorithmic cost of empty, $\alpha \in [0;1]$ a coefficient of mitigations, T a number of iterations.

The weight vector $w_i^1 = D(i)$ pour $i = 1,..,N$

A vector of normalized costs $\tilde{\lambda}_c = \dfrac{\lambda_c^\alpha}{\sum_{c=1,..,C} \lambda_c^\alpha}$

The initial error $\varepsilon_0 = 0,5$

for $t = 1$ to T **do**

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

  for $c = 1$ to C **do**

    Call the weak classifier learner $\texttt{WeakLearn}_c$ with the distribution $\mathbf{w}^t$ that returns a weak classifier $\tilde{h}_t^c = \frac{1}{2} \frac{P_{\mathbf{w}^t}(y=1|x)}{P_{\mathbf{w}^t}(y=-1|x)}$ of algorithmic cost λ_c.

    Calculate the penalized criterion to minimize

$$\tilde{\varepsilon}_t^c = \tilde{\lambda}_c \, E[e^{-y\tilde{h}_t^c}]$$

  end for

  Select the weak classifier that minimizes the penalized criterion to minimize:

$$h_t = \tilde{h}_t^{\hat{c}} \mid \hat{c} = \arg \min_{c=1,..,C} \tilde{\varepsilon}_t^c$$

  Update the weight vector

$$w_i^{t+1} = w_i^t e^{-y_i h_t(\mathbf{x}_i)}$$

  Update of the new algorithmic cost $\lambda_c = \lambda_0$ for all c calculated at the same time $\hat{c}$.

  Normalization of the cost $\tilde{\lambda}_c = \dfrac{\lambda_c^\alpha}{\sum_{c=1,..,C} \lambda_c^\alpha}$ with $\alpha \in [0;1]$

end for

return the strong classifier:

$$H(\mathbf{x}) = sign\left[\sum_{t=1}^{T} h_t(\mathbf{x})\right]$$