Person Detection with a Computation Time Weighted AdaBoost

Alhayat Ali Mekonnen, Frédéric Lerasle, Ariane Herbulot

Person Detection with a Computation Time Weighted AdaBoost

A. A. Mekonnen, F. Lerasle, and A. Herbulot

CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France
{alhayat-ali.mekonnen, frederic.lerasle, ariane.herbulot}@laas.fr

Abstract. In this paper, a boosted cascade person detection framework with heterogeneous pool of features is presented. The boosted cascade construction and feature selection is carried out using a modified AdaBoost that takes computation time of features into consideration. The final detector achieves a low Miss Rate of 0.06 at 10^{-3} False Positive Per Window on the INRIA public dataset while achieving an average speed up of 1.8× on the classical variant.

Keywords: Person Detection, AdaBoost, Feature Selection.

1 Introduction

Person detection is one of the prominent problems considered in Computer Vision. It has a vast pool of applications spanning surveillance systems, human-robot interaction, biometric data acquisition, and pedestrian protection systems in the automotive industry, to name a few. At the same time, it is a very challenging task owing to the physical variation of persons, variable appearances, occlusion, background clutter, and many more. Designing a detection system that is capable of overcoming these challenges while fulfilling real time detection requirements of many applications is still an open problem [3].

Depending on the type of sensor(s) utilized, different approaches could be employed for person detection. In this work, we focus only on monocular images which could be from a classical camera either fixed or mounted on a moving vehicle (no static camera assumption). We present a person detection system that makes use of heterogeneous pool of features–five of the most commonly used features: Haar like features [14], Edge Orientation Histograms (EOH) [4], Local Binary Patterns (LBP) [17], Histogram of Oriented Gradients (HOG) [1]–with a boosted cascade detector configuration [14]. Contrary to classical approaches which only take detection performance into consideration, our boosted cascade detector is constructed taking detection performance and feature computation time into consideration simultaneously. Our implementation shows very promising detection performance on the INRIA public dataset [1] with only a 6% Miss Rate at 10^{-3} FPPW while on average taking 1.8× less time than that of a similar detector constructed without computation time consideration.

1.1 Related Works

The problem of person detection has been studied for more than a decade by researchers. Through these times, a large number of different approaches have been proposed; it is

practically impossible to mention all due to space constraints. We will restrict this section by highlighting works on monocular images that employ heterogeneous features in a sliding window candidate generation scheme (see recent survey papers [3, 5] for a broader review).

The interesting pioneering works on person detection were first reported using Haar like features [11,15]. Since then, person detection has improved a lot. The work of Dalal and Triggs [1] which introduced and used HOG features was next in line to set the bar high. To date, HOG is the most discriminant feature, and in fact, a majority of detectors proposed hence-after make use of HOG or its variant one way or another [3]. Most of the works that have improved over [1] combine HOG with multiple other features (e.g. with LBP [17], with edgelets and covariance descriptors [18]), *i.e.*, they consider heterogeneous features.

Different features try to capture different facets of a given image: edge distribution, intensity differences, appearance variations, *etc*. Using heterogeneous features, thus, helps acquire complementary information that could be useful to handle challenging detection tasks. This has clearly been demonstrated in the literature. To mention a few: Walk et al. [16] considered HOG, Histogram Of Flow, and CSS features and carried out different experiments by combining them combinatorially. The best results were reported when considering a feature set made up by concatenating the three sets. Schwartz et al. [13] presented a detection system composed of HOG, color frequency features, and co-occurrence features, asserting similar conclusions. The same goes to Hussain and Triggs [7] whom considered feature sets made up of HOG, LBP, and Local Ternary Patterns (LTP).

The main issue to consider with heterogeneous features is how to combine them. The trivial approach is to concatenate all features to make a very high dimensional vector and use SVM as a classifier [16]. The downside with this is the high computation time required to extract the complex features and to apply the SVM weights on each candidate window which inevitably leads to low frame rates. Applying a dimensional reduction scheme [7, 13] might help performance and speed up training period, but, it still suffers during detection because of the data projection involved. In addition, different features could possibly be best dealt with different classifiers–they could for example lie in linear or non-linear spaces which may require different classification techniques [18].

A second approach is to gather all heterogeneous features in one big pool and learn an ensemble classifier using a boosting technique. Employing an attentional cascade configuration [14] is natural with this as it speeds up detection drastically. This has the added advantage that different classifier types well suited to a specific feature could be used as weak learners in the boosting framework. Representative works include the works of Dollár et al. [2], which used heterogeneous integral channel features in a boosting framework, and Geronimo et al. [4], which combined EOH and Haar like features via AdaBoost. At each iteration of the boosting learning cycle, the feature which reflects the best detection performance on the training set, measured by the weighted classification error, is added to the ensemble. Evidently, this tends to favor complex features amongst different candidates irrespective of the associate computation time. The preferred way would be to weigh detection performance against computation time and

privilege features that make a compromise between the two. In this vein, Jourdheuil et al. [8] proposed to add a multiplicative factor to penalize the criterion to minimize in AdaBoost with a normalized computation time corresponding to each feature. This way features are selected only if their combined detection performance and normalized computation time are better (have the minimum value) than the other candidates. The authors used this to detect persons in a stereo camera using features extracted from depth maps and images. Their implementation led to a detector with an acceptable detection performance and reduced computation time. Inspired by this, we use a similar formalization to learn a cascaded person detector on heterogeneous features extracted from monocular images. This work differs from [8] in three main aspects: First, it is intended for monocular images and hence employs five commonly used heterogeneous features that have never been considered all together. Second, unlike [8] which use a single node, it uses a cascaded configuration to allow early rejection for improved detection speed. Third, evaluations and results are presented on the INRIA public dataset [1]. These points conglomerated make the gist of the contribution this paper tries to make.

This paper begins by highlighting the different heterogeneous pool of features properly categorized in section 2. In section 3, it describes the classifier learning algorithm with emphasis on computation time consideration. Finally, experimental results are detailed in section 4 finishing off with concluding remarks in section 5.

2 Heterogeneous Pool of Features

The heterogeneous pool of features considered are a mix of both scalar and multi-dimensional features. Five different families of features are used. Scalar features: Haar like and Edge Orientation Histogram features; multi-dimensional features: Color Self Similarity, Local Binary Patterns, and Histogram of Oriented Gradients. Each feature family is extracted within a given fixed size image candidate window of 64x128 pixels. To generate the overcomplete set of features, the position and scale (width and height) of the region the features are computed is exhaustively varied within the candidate window. The computation time of each feature is determined irrespective of any implementation optimization that can be done during detection, e.g. use of caches to buffer some features. This helps establish an upper bound on it. For each feature considered, the computation time is made up of two components. A part associated with image preprocessing (including rudimentary feature preparation) that is mostly shared by features of the same family, and a second part pertaining to the feature extraction and necessary computation during detection (e.g., multi-dimensional feature projection). For a feature indexed by j, these are represented as $\tau_{p,j}$ and $\tau_{e,j}$ consecutively; the combined computation time of that feature becomes $\tau_j = \tau_{p,j} + \tau_{e,j}$. These values are determined by averaging over $1,000$ times repeated computation iterations.

2.1 Scalar Features

Haar like Features Haar like features represent a fast and simple way to compute region differences. These features have been extensively used for face, person, and various object detections, e.g. [4, 9, 11, 14]. For a given feature, the response is obtained by subtracting the sum of pixels spanned by the black region from the sum of pixels

spanned by the white region. In this work, we have used the extended Haar like features from Viola and Jones [14] and Lienhart and Maydt [9], which contains upright and tilted filters of various configurations as shown in fig. 1. In our implementation, a horizontal and vertical stride of 2 pixels is used to generate the overcomplete set, $\mathcal{F}_{haar}$. As these features furnish scalar feature values, a decision tree is used as a weak classifier. Computationally, these features are very cheap to extract. The pre-processing stage

Fig. 1: Set of extended Haar like features used.

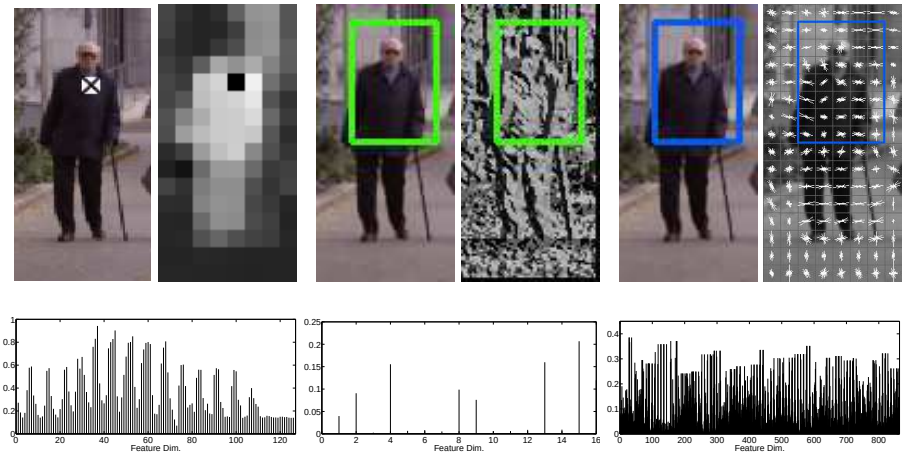
for this family of features corresponds to the integral image computation. The feature extraction is carried out efficiently with a few additions and subtractions on the integral image; feature evaluation (prediction) during detection is done via a table lookup operation (using a table expanded from the decision tree).

Edge Orientation Histogram (EOH) EOH is another feature set that has been used for person detection [4]. These features represent ratios of gradients computed from edge orientations histograms. Within a given overlaid region, first gradients are computed. Then, a gradient histogram is built by quantizing the gradient orientations. Finally, the ratios of each histogram bin with one another makes up individual features. The overcomplete EOH feature pool set, denoted $\mathcal{F}_{EOH}$, is constructed by extracting feature values for all possible combinations of overlaid region location and size within the candidate window.In this work, gradient orientation quantization levels of 4 (shown to give best results in [4]) and horizontal and vertical strides of 4 pixels are used. Owing to the scalar feature values, decision trees are used as a weak classifier.

The pre-processing step, here, corresponds to the gradient computation (this is also shared by HOG features). The feature extraction relates to the gradient histogram construction and bin ratio computation. Similar to Haar like features, the feature evaluation during detection is merely a memory lookup operation.

2.2 Multi-dimentional Features

Color Self Similarity (CSS) CSS features, proposed by Walk et al. [16], encode color similarities in different sub-regions. To compute the features, first the image window is subdivided into non-overlapping blocks of 8x8 pixels. Then within each block, a 3x3x3 color histogram in HSV space is built with interpolation. Then, similarities are computed by intersecting individual histograms. In [16], all histogram intersections values are used to define one feature vector. But, here, we define the intersection of one histogram block with the rest of the blocks as a single feature. With an 8x8 block size and 64x128 candidate window, there are 128 different blocks. The intersection of one block with the rest gives 127 scalar values (excluding intersection with itself). These scalar values all together make the feature vector computed for the block location. This is repeated for each block resulting in 128 different features, the CSS feature pool set ($\mathcal{F}_{CSS}$), of 127 dimensions each. Fig. 2a shows an exemplar feature computed at the crossed block position. The figure at the bottom is an unrolled representation of the feature shown at the top right; observe how neighboring blocks are similar. These features are few in number and multi-dimensional, hence, SVM is used as a weak classifier.



(a) CSS (b) CS-LBP (c) HOG

Fig. 2: Sample extracted multi-dimensional features.

The complete set of features extracted from this family rely on a histogram computed once for each block. If one feature is extracted, extracting another feature involves using the same histogram with changed reference block for the histogram intersection. Keeping this in mind, the pre-processing stage encompasses the histogram computation. The rest of the computation time is consumed by the extraction (histogram intersection) and the multiplication by the SVM hyperplane during detection.

Local Binary Pattern (LBP) Local Binary Patterns were initially proposed as a texture characterization features [10]. Since then, they have been used in many applications– primarily facial analysis, e.g. [19], and person detection, e.g. [17]. To date, many variants of LBP have been proposed. In this work, we adhere to Center-Symmetric Local Binary Pattern (CS-LBP) as feature because of the short histograms it furnishes and its demonstrated good performance on person datasets [6]. In our implementation, CS-LBP is computed over a 3x3 pixel region (best results reported in [6]) by comparing the opposite pixels and adding a modulated term accordingly (see [6] for detail). This gives a scalar value less than 16 which is assigned to the center pixel. This is done for all the pixels in the candidate window, top right image in fig. 2b (values are scaled to aid visibility). Finally, the actual feature is constructed by constructing a CS-LBP histogram over a given overlaid region. Fig. 2b bottom shows an exemplar histogram obtained from the overlaid rectangular region shown on top. Similarly, by varying the position and scale of the overlaid region exhaustively within the candidate window gives the LBP feature pool, denoted $\mathcal{F}_{CLBP}$. The histograms have 16 bins corresponding to CS-LBP quantization levels. Since the feature vectors are no more scalar, Fisher's Linear Discriminant Analysis (LDA) with decision tree is used as a weak classifier. Fisher LDA is preferred over SVM because of its comparatively short training duration. Given the large number of features in this feature pool family (table 1), employing SVM would lead to an overwhelming training period.

The pre-processing stage for this feature family corresponds to the raw CS-LBP feature computation. The rest is made up of histogram construction (extraction phase), and LDA projection during detection.

Histogram of Oriented Gradients (HOG) HOG features are extracted first by computing the gradient, then by constructing a histogram weighted by the gradient in an atomic region called a cell. Histograms of neighboring cells are grouped into a single block, cross-normalized and concatenated to give a feature vector per block. The final extracted feature within a given detection window is the concatenation of the vectors from each feature block (details in [1]). In this work, we use the original HOG features as proposed by Dalal and Triggs [1] with a cell size of 8x8 pixels, a feature block size of 2x2 cells and an 8 pixel horizontal and vertical stride. For a given overlaid region, the feature vector corresponds to a concatenation of all block features within it. The over-complete set, $\mathcal{F}_{HOG}$, is then generated by varying the location and scale of the overlaid region. Fig. 2c demonstrates a sample feature computed in this manner. Computation time wise, the gradient computation falls in the pre-processing step and the rest in the extraction part.

Table 1: Feature pool summary with minimum and maximum feature computation time in each feature family. Time is reported as a multiple of the cheapest feature computation time of 0.0535 μs.

Feature Type	No of features	τ_{min}		τ_{max}		Weak Classifier
		$(t_p)_{min}$	$(t_e)_{min}$	$(t_p)_{max}$	$(t_e)_{max}$	
Haar like	672,406	0.6	0.4	1.88	1.6	Decision Tree
EOH	712,960	2.72	2.11	315.65	2.1	Decision Tree
LBP	59,520	1.24	14.26	111.6	282.04	LDA + Decision Tree
CSS	128	560.75	457.19	560.75	457.19	SVM
HOG	3,360	10.59	479.12	315.75	51103.8	SVM

Table 1 summarizes the characteristics of the heterogeneous pool of features considered. The total number of features in each family, the minimum and maximum feature computation time (both pre-processing, t_p, and extraction, t_e)–scaled with 0.0535 μs[1] which corresponds to the combined computation time of cheapest feature, a two boxed horizontal Haar feature–along with the weak classifier used are listed. Finally, the complete feature pool is determined by merging all heterogeneous feature pool sets, *i.e.* $\mathcal{F} = \{\mathcal{F}_{Haar}, \mathcal{F}_{EOH}, \mathcal{F}_{CLBP}, \mathcal{F}_{CSS}, \mathcal{F}_{HOG}\}$. The computation time of each individual feature is denoted as τ_j, where $j \in \{1, 2, ..., |\mathcal{F}|\}$. Each feature is also associated with a weak learner h_j that maps each instance of the training set to a discrete label, $h_j : X \rightarrow \{-1, +1\}$.

3 Classifier Learning Algorithm

3.1 Feature Computation Time

Equation 1 shows the smoothed normalized computation time, $\tilde{\tau}_j$, for each feature which will be used within AdaBoost. $\beta \in [0, 1]$ is an exponential smoothing coefficient. $\tau_{max,i}$ denotes the maximum computation time registered within each distinct feature pool family, *i.e.*, $i \in \{Haar, EOH, CLBP, CSS, HOG\}$.

$$\tilde{\tau}_j = \frac{\tau_j^{\beta}}{\sum_i \tau_{max,i}^{\beta}} \tag{1}$$

[1] Computed on a core i7 machine running at 2.4*Ghz*

The computation time associated with each feature, $\tau_j = \tau_{p,j} + \tau_{e,j}$, is not constant (consequentially $\tilde{\tau}_j$ changes too). The exact value evolves during the classifier learning stage. It changes in two cases. The first is when a feature that has already been selected is considered in future cascade nodes, and the second is when a feature from the same family gets selected. In the prior case, the computation time of the selected feature is replaced by a constant time, τ_0, in future references which accounts for only memory access. In the latter case, the computation time for all of the features in the same family gets affected, specifically, the time associated with the pre-processing stage, $\tau_{p,j}$, is set to zero for all the features in that family. This is logical and is done to favor features of the same family. For example, if a Haar feature is selected, it will better to consider another Haar feature so the integral image computation can be done once for the area spanned by the two features, rather than considering another feature from a different family which will require a different pre-processing step. This way the computation time of the features within the same family will be levied significantly speeding up detection. Accordingly, the normalized computation time of all affected features is updated. The computation time, $\mathcal{T}_k$, of a trained cascade node k is determined straightforward by adding the computation time of each selected component features (associated weak learners), *i.e.*, $\mathcal{T}_k = \sum_{t=1}^{T} \tau_t$. Here, an index t is used to signify reference of a selected feature and T represents the total number of features in this cascade node.

3.2 Modified Discrete AdaBoost

Given the complete heterogeneous feature pool, $\mathcal{F}$, and associated computation time, $\{\tau_j\}_{\{j=1,2,\ldots,|\mathcal{F}|\}}$, a modified version of Viola and Jones Discrete AdaBoost [14] is used to learn a strong classifier for each node of the cascade. As discussed in section 2, each feature is associated with a unique weak learner, h_j, that maps the given training set X to a discrete label, *i.e.*, $h_j : X \rightarrow \{-1, +1\}$ [2]. The original Discrete AdaBoost algorithm constructs a strong classifier by iteratively selecting the best weak classifier, h_t, based on the error distribution on the training set, ϵ_j, weights it, with α_t, and adds it to the ensemble. Each subsequent addition tries to correct the errors made by previously added weak classifiers. The modification here is to select the best weak classifier that minimizes the error weighted with a normalized computation time of the features, equation 2. This modification enables AdaBoost to select the feature (weak learner) that offers the best compromise between computation time and detection error. This is detailed in algorithm 1 (main modifications on the classical one are shown in bold typeface).

$$h_t = \arg\min_{h_j \in \mathcal{F}} \tilde{\tau}_j * \epsilon_j \tag{2}$$

Given the vast number of features involved, looping through each feature set at each iteration of AdaBoost is infeasible. Hence, as is commonly done, e.g. in [18,20], at each AdaBoost iteration 3000 features are randomly sampled, proportional to the number of features from each family, to build the strong per node classifier incrementally. 3000 is well way above the suggested number of trials, ≈ 299, required to obtain amongst the best 0.05 estimates of random variables with a probability of 0.99 [12](pp. 180). Once,

[2] Because of this we can use a weak learner and a feature interchangeably.

the required detection performance, specified a priori via a Node Miss Rate (NMR) and Node False Positive Rate (NFPR), is achieved on a separate validation set, the feature addition stops and the node is retrained using both the training and validation set up until the previously validated number of features have been added (see [14] for details). The final ensemble obtained is the strong classifier for this node and construction proceeds to the next node.

Algorithm 1 Modified Discrete AdaBoost

Given: A set of labeled examples $\{(x_m, y_m)\}\{m = 1, ..., (n_+ + n_-)\}$ where $x_m \in X$, $y_m \in Y = \{-1, +1\}$

Extract Features: $\mathcal{F} = \{\mathcal{F}_{Haar}, \mathcal{F}_{EOH}, \mathcal{F}_{CLBP}, \mathcal{F}_{CSS}, \mathcal{F}_{HOG}\}$

Initialize: $D_1(m) = \dfrac{1}{(n_+ + n_-)}$ *distribution over training samples*

For $t = 1, ..., T$

- Select 3000 features randomly, $\mathcal{F}^* \subset \mathcal{F}$
- Find the best weak learner $h_t : X \rightarrow \{-1, +1\}$
  - **Compute** $\tilde{\tau}_j = \dfrac{\tau_j^{\beta}}{\sum_i \tau_{max,i}^{\beta}}$
  - $h_t = \underset{h_j \in \mathcal{F}^*}{\arg\min}\, \tilde{\tau}_j * \epsilon_j$ where $\epsilon_j = \sum_{m=1}^{n_+ + n_-} D_t(m)[y_m \neq h_j(x_m)]$
- **Update the computation time of the selected feature** τ_j to τ_0
- **Update the computation time of the features in the same family as the selected feature by setting their pre-processing time to 0**, ***i.e.***, $\tau_j \leftarrow \tau_{e,j}$
- $\alpha_t = \dfrac{1}{2}\ln\dfrac{1 - \epsilon_t}{\epsilon_t}$
- $D_{t+1}(m) = \dfrac{D_t(m)\exp(-\alpha_t y_m h_t(x_m))}{Z_t}$ *Z_t is a normalization factor used to make D_{t+1} a distribution*

Strong classifier: $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$ **Computation time of cascade node:** $\mathcal{T} = \sum_{t=1}^{T} \tau_t$

3.3 Cascade Construction

The cascade detector construction is inspired by the works of Viola and Jones [14]. The detection performance requirement of each cascade node are specified by two parameters, the NMR and NFPR. Given a total of N_+ and N_- cropped positive and negative training windows respectively (bear in mind $N_- >> N_+$), the construction begins by randomly selecting a subset of negative windows, n_-, and using all positive training windows, $n_+ = N_+$. This is divided into a training and validation set, in our case 60% and 40% respectively. Following, the modified AdaBoost is used to learn the strong classifier for this node using the heterogeneous features set.

This learning is monitored via the validation set; at each boosting iteration the zero threshold is lowered to see if the strong classifier meets the NMR and NFPR criteria.

When that is achieved, the AdaBoost is retrained using the whole training and validation set to the point determined during validation. This completes the construction of the first node. Henceforth, all N_- are tested with the trained node and all those that get misclassified (harder examples) are used in consecutive stages of the cascade. The process continues until such a point where the number of negative windows is less than the positive windows, in which case the construction terminates furnishing the whole detector cascade. The MR and FPR of the entire cascade are products of the nodes NMR and NFPR consecutively.

4 Experiments and Results

4.1 Evaluation metrics

To evaluate the detection performance, Detection Error Tradeoff (DET) curves with Miss Rate versus False Positive Per Window (FPPW) on a log-log scale are used [1]. To determine these values the True Positives, False Positives, True Negatives, and False Negatives of the test set are determined via a per-window approach [3]. The per-window approach relies on cropped labeled positive and negative train and test set. The training is performed using these cropped images and the test likewise (please refer [3] for details).

4.2 Dataset

Experiments are carried out using the public INRIA person detection dataset [1]. The training set for this dataset consists of $2,416$ cropped positive instances and $1,218$ images free of persons (out of which many negative train cropped windows are generated). The test set contains $1,132$ positive instances and 453 person free images for testing purposes. This is the most widely used dataset for person detector validation and comparative performance analysis. For constructing the cascade the complete positive instances and randomly sampled, at different scales and locations, 2.55×10^6 negative cropped instances from the training set are used. During cascade construction, the number of negative windows per each node, n_-, has been kept equal to the positive windows n_+, *i.e.,* 2,416. For testing, the $1,132$ positive instances and uniformly sampled 2×10^6 cropped windows from the test set are used.

4.3 Results

Validation: In our framework the parameters that need to be specified are per node NMR and NFPR and the depth of the decision tree to use. Since we want the cascade to detect all possible positive instances, a 0.0 NMR is used for each node. The NFPR on the other hand affects the number of cascade nodes built. Higher values result in more number of nodes to exhaust the training set. As a compromise, NFPR of 0.5 is used so each node is required to discard at least 50% of the incoming candidate windows.

The best depth of the decision trees for Haar like features, EOH, and LBP features is determined empirically using a 1 fold cross validation. A decision tree depths of 2, 3, and 3 are used for Haar like features, EOH features, and LBP features respectively. The weak classifiers learned using these depths offer a better trade off between detection performance and overfitting on the validation set. Computing Fisher LDA weights, for

LBP features, per each node makes the classifier overfit on the training set with deteriorated performance on the validation set. Hence, the LDA weights for LBP computed at the first node are used throughout the cascade by learning only new decision trees.

Similarly, the exact value of β, the computation time smoothing exponential factor, to use in the modified AdaBoost is determined empirically through a validation step. The modified AdaBoost is used to learn a single nodal cascade using different β values on a subset of the training set. Then the classification errors on a validation set and the conglomerated computation time of the trained node is determined to select the best value that offers a good trade-off. Fig. 3 shows the validation result plots for different values of β. Clearly, higher β reduces smoothing, in effect, features with low computation time dominate improving speed but with poor detection performance. Lower values favor complex features. As a compromise, a β value of 0.2 is used to train the final cascade classifier.

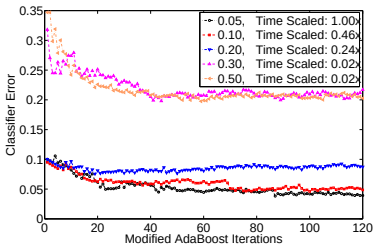


Fig. 3: β parameter tuning in the modified algorithm.

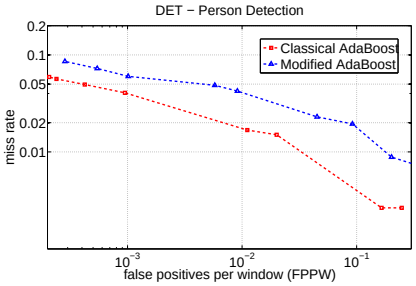


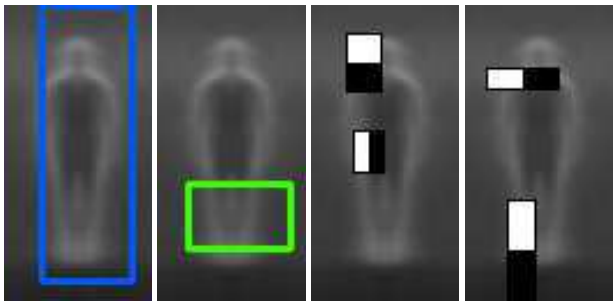
Fig. 4: Comparative performance evaluation of the cascade detector on the INRIA dataset.

Table 2: Proportion of features in the final cascaded detector.

	Haar like	EOH	LBP	CSS	HOG	Average Time Improvement
Classical AdaBoost	25.36%	53.79%	10.16%	4.00%	6.67%	1.0x
Modified AdaBoost	86.72%	0.00%	9.14%	2.40%	3.90%	1.8x

Test results: A complete boosted cascade detector is learned using the framework presented, *i.e.*, with a modified computation time weighted AdaBoost, using the combined training and validation dataset (referred as modified AdaBoost hereafter). As a benchmark, a second complete cascade detector is also learned using the classical Discrete AdaBoost (referred as classical AdaBoost hereafter). The modified AdaBoost cascade has ten nodes with a combined total of 821 features. The minimum and maximum number of features per node are 6 in the first and 350 on the 7^{th} node, respectively. The proportion of selected features from each family is shown in table 2. The most used features are Haar like features, 86.72%, and the least are CSS features, 2.40%. Compared to the classical cascade, there is smaller number of the complex multi-dimensional features. No EOH features are selected. This is because, Haar like and EOH features exhibit comparable detection performance but Haar like features have less computation time thus are privileged. The selected features for the first node of the cascade are shown in fig. 5. For HOG and LBP, the illustrations show the support region of the selected

features. The four selected Haar like features are also shown in fig. 5c and 5d. Indeed, four out of six selected features are from the computationally cheapest feature category.



(a) HOG (b) LBP (c) Haar like (d) Haar like

Fig. 5: The features selected and used in the first node of the cascade superimposed on an average human gradient image.

The performance of this modified detector on the test set is depicted as an MR vs FPPW plot in fig. 4. It achieves a 0.06 MR at 10^{-3} FPPW. This is a 2% loss compared to the detection performance achieved by the classical AdaBoost at the same FPPW. But, this reduction results on an average 1.8× accelerated speed. The final detector misses only 97 persons out of the total 1132 tested instances. A majority of these mistakes correspond to persons with non-upright/non-conventional poses. Some of the detection mistakes are shown in fig. 6 for both positive and negative samples.[3]

(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 6: Sample misclassified positive (a to d) and negative (e to h) instances.

5 Conclusion

In conclusion, in this work, a computation time weighted AdaBoost has been presented to learn a cascaded person detector using heterogeneous pool of features. The final detector achieves 1.8× average speed up compared to a classical AdaBoost based cascade with only a 2% detection performance loss at 10^{-3} FPPW. At this point, it is capable of detecting persons with a very low Miss Rate of 6% as demonstrated on the INRIA public dataset.

[3] Sample detections are not shown here due to space constraints. Please visit homepages.laas.fr/aamekonn/acivs/ for illustration.

Currently, investigations are on the way to use this detector in a tracking-by-detection framework in the context of Robotic navigation in crowds.

Acknowledgment

This work was supported by a grant from the French National Research Agency under grant number ANR-12-CORD-0003.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. CVPR. pp. 886–893 (2005)
2. Dollar, P., Tu, Z., Perona, P., Belongie, S.: Integral channel features. In: Proc. BMVC. pp. 1–11 (2009)
3. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. IEEE T-PAMI 34(4), 743–761 (2012)
4. Gerónimo, D., López, A.M., Ponsa, D., Sappa, A.D.: Haar wavelets and edge orientation histograms for on-board pedestrian detection. In: Proc. IbPRIA. pp. 418–425 (2007)
5. Geronimo, D., Lopez, A., Sappa, A., Graf, T.: Survey of pedestrian detection for advanced driver assistance systems. IEEE T-PAMI 32(7), 1239–1258 (2010)
6. Heikkil, M., Pietikinen, M., Schmid, C.: Description of interest regions with local binary patterns. Pattern Recognition 42(3), 425 – 436 (2009)
7. Hussain, S., Triggs, B.: Feature sets and dimensionality reduction for visual object detection. In: Proc. BMVC. pp. 1–10 (2010)
8. Jourdheuil, L., Allezard, N., Chateau, T., Chesnais, T.: Heterogeneous adaboost with real-time constraints - application to the detection of pedestrians by stereovision. In: Proc. VISAPP. pp. 539–546 (2012)
9. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Proc. ICIP. pp. 900–903 (2002)
10. Ojala, T., Pietikinen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern Recognition 29(1), 51 – 59 (1996)
11. Papageorgiou, C., Poggio, T.: A trainable system for object detection. IJCV 38(1), 15–33 (2000)
12. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
13. Schwartz, W.R., Kembhavi, A., Harwood, D., Davis, L.S.: Human detection using partial least squares analysis. In: Proc. ICCV. pp. 24–31 (2009)
14. Viola, P.A., Jones, M.J.: Robust real-time face detection. IJCV 57(2), 137–154 (2004)
15. Viola, P.A., Jones, M.J., Snow, D.: Detecting pedestrians using patterns of motion and appearance. In: Proc. ICCV. pp. 734–741 (2003)
16. Walk, S., Majer, N., Schindler, K., Schiele, B.: New features and insights for pedestrian detection. In: Proc. CVPR. pp. 1030–1037 (2010)
17. Wang, X., Han, T., Yan, S.: An HOG-LBP human detector with partial occlusion handling. In: Proc. ICCV. pp. 32–39 (2009)
18. Wu, B., Nevatia, R.: Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. In: Proc. CVPR. pp. 1–8 (2008)
19. Zhao, G., Pietikainen, M.: Dynamic texture recognition using local binary patterns with an application to facial expressions. IEEE T-PAMI 29(6), 915–928 (2007)
20. Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: Proc. CVPR. pp. 1491–1498 (2006)