Soft-Cascade Learning with Explicit Computation Time Considerations

Francisco Rodolfo Barbosa-Anda, Frédéric Lerasle, Cyril Briand, Alhayat Ali Mekonnen

Soft-Cascade Learning with Explicit Computation Time Considerations

Francisco Rodolfo Barbosa-Anda Frédéric Lerasle Cyril Briand Alhayat Ali Mekonnen
LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France
{frbarbos,lerasle,briand,aamekonn}@laas.fr

Abstract

This paper presents a novel framework for learning a soft-cascade detector with explicit computation time considerations. Classically, training techniques for soft-cascade detectors select a set of weak classifiers and their respective thresholds, solely to achieve the desired detection performance without any regard to the detector response time. Nevertheless, since computation time performance is of utmost importance in many time-constrained applications, this work divulges an optimization approach that aims to minimize the mean cascade response time, given a desired detection performance, fixed beforehand. The resulting problem is NP-Hard, therefore finding an optimal threshold vector can be very time-consuming, especially when building a soft-cascade detector of long length. An efficient local search procedure is presented that deals with long-length detectors. Our evaluations on two challenging public datasets confirm that a faster cascade detector can be learned while maintaining similar detection performances.

1. Introduction

Person detection is one of the most important challenges in the literature with a wide application range including video surveillance [4], image indexing and retrieval [34], intelligent robotic systems [14], and Advanced Driver Assistance Systems [17]. These issues are also illustrated by well-known challenges, e.g., the PASCAL challenge [15] and the MOT Challenge [21], which are very competitive in the Computer Vision community.

Recent trends on visual person detection have moved towards deep learning paradigms [19, 28, 24, 22]. Though these paradigms are showing detection performance gains, they induce specific, cumbersome, and expensive GPU architectures. This limits drastically their use for embedded applications. Such applications must favor conventional (cheap) architectures and the CPU cost could then represent a bottleneck for real time embedded applications. These insights motivate numerous investigations on image feature selection (thanks to boosting variants), and hard or soft cascade arrangements initiated by [30, 13]. Soft-cascade based detectors, especially, appear to be still competitive and even slightly better both in terms of detection performance and computation time than deep learning ones [35].

For both hard/soft cascade arrangement, the strategy is to inspect the image exhaustively by the sliding window technique [2]. The key idea is to cover all positions in the image and all sizes of the target person.

Cascade detectors computation time is mainly determined by two factors: the weak classifiers that compose the cascade and the threshold values of each stage of the cascade. For the first factor, selection methods have been proposed that put faster weak classifiers first and let more efficient ones for the end of the cascade, e.g., [23, 20]. For the second factor, threshold tuning approaches search to detect negative samples as fast as possible in the cascade, e.g., [32, 3, 1].

The cascade arrangement can be a *hard-cascade*, in which each cascade stage aggregates the weighted score of the several weak classifiers, and thresholds the resulting value to label each sample as positive (which is passed to the next stage) or as negative (which is rejected). It can also be a *soft-cascade* – instead of having separated cascade stages, it has one stage, and then it thresholds each sample response after each weighted weak classifier evaluation [32, 3]. In the literature, these types of cascades are predominantly constructed using AdaBoost, e.g., [30, 37, 32, 10].

Recently, soft-cascade variants have demonstrated much superior detection performance [35, 36, 10, 13] and are used in several applications, e.g., [27, 29, 18]. The soft-cascade thresholds are learned after the complete training of the strong classifier in a kind of calibration phase. They are solely learned to fulfill detection performance requirements without any computation time consideration. However, in real-time systems, when using a sliding window detection approach, it is worth to consider computation time aspects explicitly. Hence, this work focuses on incorporating computation time considerations in the calibration of a soft-cascade detector by determining the optimal thresholds that minimize the expected time response.

This work relies on the Mean-Cascade Response-time Minimization Problem (MSCRMP) presented by Barbosa-Anda *et al.* [1] and the Binary Integer Programming (BIP) optimization framework proposed to determine optimal rejection threshold values therein. The rejection threshold values minimize the overall computation time while keeping the detection performance unchanged. Their approach is actually quite efficient to train medium-length soft-cascades with medium training sets with a maximal size of 256 stages, but cannot deal with real-size instances that have at least 2048 stages. In this paper, we propose a novel local-search optimization procedure for tuning the cascade thresholds which is able to cope with large-problem instances efficiently. Finally, we demonstrate experimentally and comparatively the viability of the framework on a relevant application, namely pedestrian detection, using publicly available real life datasets.

The rest of this paper is structured as follows: Section 2 briefly presents related work in threshold tuning strategies (already mentioned in this introduction). Then, the proposed novel local search approach is presented in Section 3. Relevant experimental evaluations, results, and associated discussions are detailed in Section 4. Finally, concluding remarks are provided in Section 5.

2. Existing threshold tuning strategies

In a soft-cascade, the main interest is to reject as many of the negative windows at the earliest possible level, thereby (1) decreasing the computation time, and (2) decreasing the False Positive Rate (FPR). Let us consider a strong classifier in the form $\mathcal{H}(x) = \sum_{l=1}^{L} \alpha_l h_l(x)$. x corresponds to a test sample which can be positive ($y = 1$) or negative ($y = -1$), $h_l(x)$ is the l^{th} weak classifier, and $sign(\mathcal{H}(x) - \theta_L)$ determines the final class label. The intermediate cumulative score of a sample at the l^{th} weak classifier is defined by $S_l = \sum_{u=1}^{l} \alpha_u h_u(x)$. A soft-cascade algorithm deals with determining the rejection thresholds θ_l that reject the sample whenever $S_l < \theta_l$ at each weak classifier evaluation (or otherwise pass it to the next stage). The most notably works that propose different strategies to set these thresholds are the BIP-based approach of Barbosa-Anda *et al.* [1], the Direct Backward Pruning (DBP) of Zhang and Viola [32], the WaldBoost of Sochman and Matas [26], the "soft-cascade" of Bourdev and Brandt [3], and the boosting chain of Xiao *et al.* [31]. Even though it is not very important here, the specific weak classifier to use at each iteration of a soft-cascade, h_l, and the associated weighting coefficients, α_l, can be derived by minimizing the exponential loss (as in the case of AdaBoost), which provides an upper bound on the actual 1/0 loss classification [25].

Bourdev and Brandt [3] set the rejection thresholds based on a rejection distribution vector – an ad hoc detection rate target, which is arbitrary and non-optimal [32]. Sochman and Matas [26] use a ratio test to determine rejection thresholds. On the other hand, DBP [32] – which outperforms WaldBoost, "soft-cascade", and boosting chain – sets the thresholds to the minimum score S_l registered by any of the positive samples that have a final score S_L above the final threshold θ_L. Although DBP has been successfully used in several detection applications, e.g., [32, 13], it does not explicitly consider the computation time incurred when applying the detector on an image. To address this issue, Barbosa-Anda *et al.* [1] proposed a BIP-based approach that leads to a faster cascade over DBP under similar detection performances.

2.1. Direct Backward Pruning

Zhang and Viola [32] proposed the so-called Direct Backward Pruning (DBP) procedure that uses the intermediate cumulative score of a sample x_n at each l^{th} weak classifier that equals $S_{n,l} = \sum_{u=1}^{l} \alpha_u h_{n,u}$, where α_l is the positive weight associated to the weak classifier and $h_{n,l}$ is its known response ($h_{n,l} = 1$ if x_n is a positive sample, $h_{n,l} = -1$ otherwise). At each level l, a threshold θ_l is defined, according to (1), such that any sample having its score $S_{n,l} < \theta_l$ is rejected. As it can be observed, the definition of θ_l itself depends on θ_L, which is the final desired threshold that allows to provide the desired minimum number of true positive samples TPR.

$$\theta_l = \min_{\{n|S_{n,L} > \theta_L, y_n = 1\}} S_{n,l} \tag{1}$$

2.2. Mean Soft-Cascade Response-time Minimization Problem (MSCRMP)

We recall now the definition of MSCRMP as given in [1]. A soft-cascade with L weak classifiers trained on a configuration set of N samples is considered, composed of J positive and K negative samples (i.e., $\mathbf{N} = \mathbf{J} \cup \mathbf{K}$). The indexes n, j and k are further used for designating a sample (either positive or negative), a positive one and a negative one, respectively. Each weak classifier at level l has a positive cost c_l, which represents the time it requires to check a single sample. $S_{n,l}$ and θ_l still refers to the intermediate cumulative score and the threshold at level l, respectively (see Section 2.1) such that any sample x_n with $S_{n,l} < \theta_l$ is rejected. Note that, as a consequence of a rejection at level l, a computational time saving is made that equals $\sum_{u=l+1}^{L} c_u$.

In MSCRMP, unlike for DBP procedure, vector $\Theta = \{\theta_1, \ldots, \theta_L\}$ results from an optimization stage that aims at minimizing the total computation time required to analyze the configuration set or, equivalently, maximizing the saved computation time. Indeed, as this configuration set is assumed to be representative of any real-world sample set, this objective function can be seen equivalent to minimizing the mean computation time response of the detector.

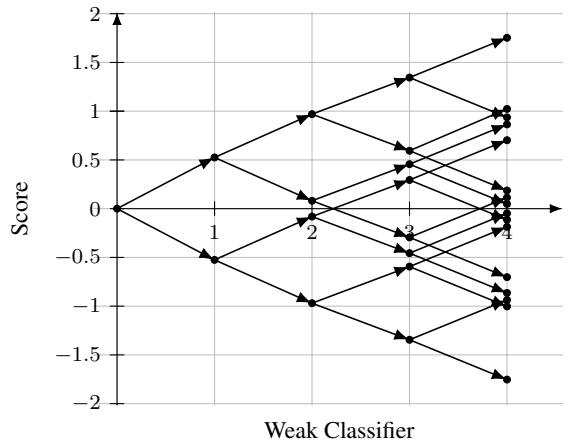

Figure 1. A score tree.

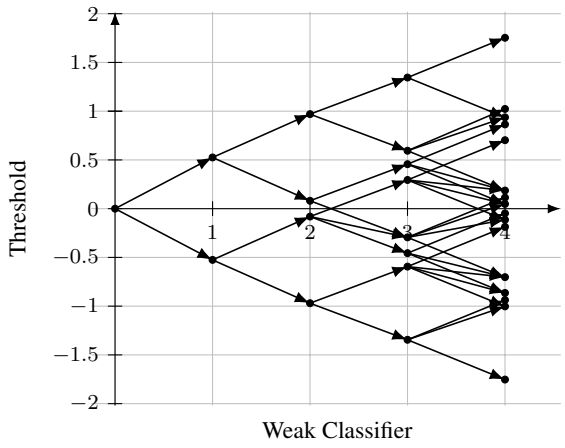

Figure 2. A threshold graph for the example of Figure 1.

MSCRMP computational complexity is proved to be NP-Hard in [1], the proof being based on a reduction from Subset-Sum Problem (SSP) [16].

Barbosa-Anda *et al.* [1] define a score tree such that each node (l, s) represents a specific possible score s that can be reached at level l: (l, s) has two children $(l + 1, s - \alpha_{l+1})$ and $(l + 1, s + \alpha_{l+1})$. Figure 1 shows the score tree obtained for an instance having four weak classifiers (trained by AdaBoost) with weights $A = \{0.5253, 0.4443, 0.3753, 0.4081\}$. This score tree allows the definition of a threshold graph $\mathcal{T}(V, E)$ with the same node set V such that an arc $e \in E$ is set between (l, s) and $(l + 1, s')$ if and only if $s' \geq s - \alpha_{l+1}$. Such a threshold graph is displayed in Figure 2 for the example of Figure 1.

Barbosa-Anda *et al.* [1] proved that any optimal solution of MSCRMP (i.e., a threshold vector Θ) can be associated with a specific path from virtual node $(0, 0)$ to a node of level L in $\mathcal{T}$ such that if node (l, s) is traversed then $\theta_l = s$. The number of different paths being clearly exponential, the authors also put dominance conditions into evidence such that $\mathcal{T}(V, E)$ can be pruned, which drastically reduces the solution space. On the basis of this property, a binary integer program (BIP) is proposed that finds an optimal path in $\mathcal{T}$. This formulation is detailed below as it is integrated as a basic component of our method. It involves binary variables $x_{n,l}$ and $\psi_{s,t,l}$. $x_{n,l} = 1$ implies that sample n is such that $S_{n,l} < \theta_l$ for the first time. $\psi_{s,t,l} = 1$ if the arc $e \in E$ linking nodes $(l, s) \in V$ and $(l+1, t) \in V$ is selected in the solution path. Objective function (2) measures the computational time saving. Constraints (3) enforce the detector to reach the true positive rate TPR (provided that $x_{n,L+1} = 1$ when the sample n has never been rejected before). Flow constraints (4)-(5) make sure that the selected arcs form a (unique) path in $\mathcal{T}$ ($\sigma^{-1}_{(l,s)}$ and $\sigma_{(l,s)}$ being the set of incoming and out coming arcs of node (l, s), respectively). Constraints (6) and (7) couple $x_{n,l}$ and $\psi_{s,t,l}$ variables together enforcing the solution path to pass at a score greater than $S_{n,l}$ at level l whenever $x_{n,l} = 1$, on the one hand, and imposing that sample n is rejected at a level lower or equal to l whenever the path passes through a score greater than $S_{n,l}$, on the other hand.

Maximize

$$\sum_{l=1}^{L} \sum_{n=1}^{N} \left[x_{n,l} \left(\sum_{u=l+1}^{L} c_u \right) \right] \tag{2}$$

Subject to

$$\sum_{n \in \mathbf{J}} x_{n,L+1} \geq TPR \tag{3}$$

$$\sum_{(l-1,t) \in \sigma^{-1}_{(l,s)}} \psi_{t,s,l-1} = \sum_{(l+1,t) \in \sigma_{(l,s)}} \psi_{s,t,l} \; \forall (s, l) \tag{4}$$

$$\psi_{0,0,1} + \psi_{0,1,1} = 1 \tag{5}$$

$$x_{n,l} \leq \sum_{\{t | S_{n,l} < t\}} \sum_{\{s | (l+1,t) \in \sigma_{(l,s)}\}} \psi_{s,t,l} \; \forall (n, l) \tag{6}$$

$$\sum_{\{t | S_{n,l} < t\}} \sum_{\{s | (l+1,t) \in \sigma_{(l,s)}\}} \psi_{s,t,l} \leq \sum_{i=1}^{l} x_{n,i} \; \forall (n, l) \tag{7}$$

$$x_{n,l} \in \{0, 1\} \; \forall (n, l) \tag{8}$$

$$\psi_{s,t,l} \in \{0, 1\} \; \forall (s, t, l) \in \mathcal{T} \tag{9}$$

3. A novel iterative search procedure for large scale problems

The previous approach has proved to be effective on small and medium size instances. Nevertheless, as the number of constraints (6) and (7) depends on the product of the

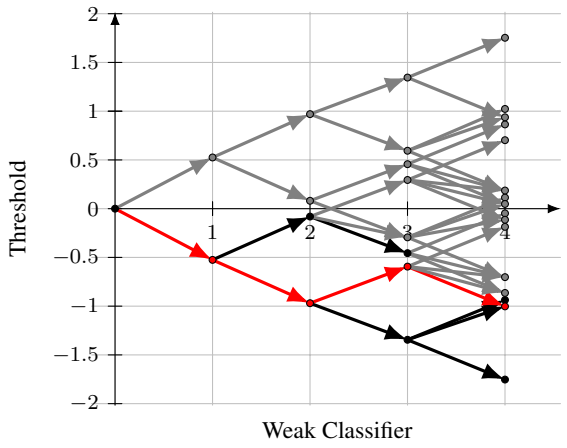

Figure 3. A neighborhood in a threshold graph.

cascade size L and the training set size N, it cannot be applied directly for large scale soft-cascades as the model becomes too huge to be solved in finite time even using the best outstanding commercial BIP solvers.

Hereafter, an iterative search procedure that allows overcoming the previous drawbacks is detailed. Even though it does not guarantee optimality anymore, it does allow to solve real size instances and provides good-quality solutions regarding both the classification performance and computation cost criteria. Two complementary techniques are used: (1) an iterative local search procedure which considers a specific neighborhood structure defined as an envelope of a path in $\mathcal{T}$, and (2) a cascade reduction procedure.

3.1. A Graph Local Search (GLS) method

Considering a solution path Θ_α of $\mathcal{T}$ found at iteration α, we define an envelope as a sub graph $\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta}) \subset \mathcal{T}(V,E)$ using a frontier vector $\Delta = \{\delta_1,\ldots,\delta_L\}$. We refer to (l,s^0) as the node of $\mathcal{T}$ selected in Θ_α at level l. The nodes of $V_{\Theta_\alpha,\Delta}$ are such that $V_{\Theta_\alpha,\Delta} = \bigcup_{l=0}^{L}\{(l,s^{-\delta_l}),(l,s^{-\delta_l+1}),\ldots,(l,s^{-1}),(l,s^0),(l,s^1),\ldots,(l,s^{\delta_l-1}),(l,s^{\delta_l})\}\}$, where (l,s^u) ((l,s^{-u}), resp.) is the u^{th} score value greater (lower, resp.) than (l,s^0). $E_{\Theta_\alpha,\Delta} \subseteq E$ are all the arcs in E that connect nodes in $V_{\Theta_\alpha,\Delta}$, so that $\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta}) \subseteq \mathcal{T}(V,E)$, as illustrated in Figure 3 where the initial solution $\Theta_\alpha = \{-0.5253, -0.9693, -0.5943, -1.0023\}$ is presented in red, the neighborhood $\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta})$ with $\delta_l = 1$ is presented in black and the non-explored threshold space $\mathcal{T}(V,E)\backslash\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta})$ is in gray. Note that Δ allows a fine control of the shape of the envelope. Obviously, $\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta}) = \mathcal{T}(V,E)$ for large δ_l values.

Barbosa-Anda *et al.* [1]'s model can be applied efficiently on sub graph $\mathcal{T}_{\Theta_\alpha,\Delta}(V_{\Theta_\alpha,\Delta}, E_{\Theta_\alpha,\Delta})$ finding another path $\Theta_{\alpha+1}$ strictly better than Θ_α. In that case, a new envelope can be defined from $\Theta_{\alpha+1}$ and the solving process can be repeated until there is no more improvement. Otherwise, the envelope can be expanded progressively around Θ_α until a better solution is found or a given maximum value is reached.

Algorithm 1 Graph local search

Require: $\text{tpr}(\Theta_0) \geq TPR$ and $\delta_{\max} \geq 1$
  $\alpha \leftarrow 1$
  $better1 \leftarrow$ **true**
  while better1 **do**
    $\delta \leftarrow 1$
    $better2 \leftarrow$ **false**
    $\Theta_{\alpha,0} \leftarrow \Theta_{\alpha-1}$
    while not $better2$ **and** $\delta \leq \delta_{\max}$ **do**
      $\Theta_{\alpha,\delta} \leftarrow \text{BIP}(\mathcal{T}_{\Theta_\alpha,\delta-1}, TPR)$
      $better2 \leftarrow \text{objfunc}(\Theta_{\alpha,\delta}) < \text{objfunc}(\Theta_{\alpha,\delta-1})$
      $\delta \leftarrow \delta + 1$
    end while
    $\Theta_\alpha \leftarrow \Theta_{\alpha,\delta-1}$
    $better2 \leftarrow \text{objfunc}(\Theta_\alpha) < \text{objfunc}(\Theta_{\alpha-1})$
    $\alpha \leftarrow \alpha + 1$
  end while
  return $\Theta_{\alpha-1}$

3.2. A cascade reduction procedure

Given a feasible solution Θ, if the mandatory TPR target is first reached at stage $l_{TPR} < L$, then threshold values for stages $l_{TPR}+1$ to L are imposed: they equal the minimum score of the true positive samples in stage l_{TPR}. Indeed, as the mandatory TPR target is reached at stage l_{TPR}, the detector is not allowed to classify as negative any of the remaining true positives samples until the end of the cascade.

On the basis of this property and given an initial solution Θ_0, we only focus on determining a partial solution $\Theta^{L'}$ for the first $L' = l_{TPR}$ stages. We set the complete solution Θ according to Equation (10) – i.e, for all stages $l \leq L'$ the threshold values θ_l are those of the partial solution $\Theta^{L'}$, while for all other stages, the threshold values θ_l are set to the minimum score $S_{n,l}$ registered by the positive samples that have a final partial score $S_{n,L'}$ greater than $\theta^{L'}_{L'}$.

$$\theta_l = \begin{cases} \theta^{L'}_l & l \leq L' \\ \min\limits_{\{n|S_{n,L'}>\theta^{L'}_{L'}, y_n=1\}} S_{n,l} & \text{otherwise} \end{cases} \tag{10}$$

The complete solution Θ could be used as a new initial solution Θ_β repeating the procedure until the new l_{TPR} is equal to the actual cascade length L', as presented in

Algorithm 2. This procedure shortens the cascade length, thereby reducing the threshold graph to be explored and hence the computational effort.

Algorithm 2 Cascade reduction procedure

Require: $\text{tpr}\,(\Theta_0) \geq TPR$
  $better \leftarrow$ **true**
  $\beta \leftarrow 1$
  $L' \leftarrow L$
  while $better$ **and** $l_{TPR} < L'$ **do**
    $L' \leftarrow l_{TPR}$
    $\Theta^{L'}_{\beta-1} \leftarrow \{\theta_1, \ldots, \theta_{L'}\} \in \Theta_{\beta-1}$
    $\Theta^{L'}_{\beta} \leftarrow \text{BIP}\left(\mathcal{T}_{\Theta^{L'}_{\beta-1}}, TPR\right)$
    for all $l | 1 \leq l \leq L$ **do**
      if $l \leq L'$ **then**
        $\theta_l \in \Theta_\beta \leftarrow \theta^{L'}_l \in \Theta^{L'}_\beta$
      else
        $\theta_l \in \Theta_\beta \leftarrow \min_{\{n | S_{n,L'} > \theta^{L'}_{L'} \in \Theta^{L'}_\beta, y_n = 1\}} S_{n,l}$
      end if
    end for
    $better \leftarrow \text{objfunc}\,(\Theta_\beta) < \text{objfunc}\,(\Theta_{\beta-1})$
    $\beta \leftarrow \beta + 1$
  end while
  return $\Theta_{\beta-1}$

3.3. Mixed method

We mix both previous procedures. The soft cascade is truncated as explained in Section 3.2. Instead of using the BIP as in Algorithm 2, the GLS is applied with respect to Algorithm 1, using as input the initial solution Θ_β (with performance TPR). The final procedure is as in Algorithm 3.

Algorithm 3 Iterative search procedure

Require: $\text{tpr}\,(\Theta_0) \geq TPR$ and $\delta_{\max} \geq 1$
  $better \leftarrow$ **true**
  $\beta \leftarrow 1$
  $L' \leftarrow L$
  while $better$ **and** $l_{TPR} < L'$ **do**
    $L' \leftarrow l_{TPR}$
    $\Theta^{L'}_{\beta-1} \leftarrow \{\theta_1, \ldots, \theta_{L'}\} \in \Theta_{\beta-1}$
    $\Theta^{L'}_{\beta} \leftarrow \text{GLS}\left(\Theta^{L'}_{\beta-1}, TPR, \delta_{\max}\right)$
    for all $l | 1 \leq l \leq L$ **do**
      if $l \leq L'$ **then**
        $\theta_l \in \Theta_\beta \leftarrow \theta^{L'}_l \in \Theta^{L'}_\beta$
      else
        $\theta_l \in \Theta_\beta \leftarrow \min_{\{n | S_{n,L'} > \theta^{L'}_{L'} \in \Theta^{L'}_\beta, y_n = 1\}} S_{n,l}$
      end if
    end for
    $better \leftarrow \text{objfunc}\,(\Theta_\beta) < \text{objfunc}\,(\Theta_{\beta-1})$
    $\beta \leftarrow \beta + 1$
  end while
  return $\Theta_{\beta-1}$

4. Evaluations and discussion

To validate our contributions, two soft-cascades were trained and tested: (1) using images taken from the public INRIA Person dataset [8]; and (2) another one using images taken from the public Caltech Pedestrian dataset [12, 13]. The training is carried out using Piotr's Computer Vision Matlab Toolbox [9] that provides an implementation of the Aggregate Channel Features (ACF) soft-cascade from [10]. Here we chose to use ACF as a benchmark as it is one of the best detectors in the literature taking both speed and detection performance into account. ACF based on the notion of channel features that has outperformed several detectors on various benchmarking datasets [10]. It is based on aggregates of features represented as channels. A channel is a per-pixel feature map computed from a corresponding patch of input pixels. It can, for example, be the L component of the LUV color transformed input image, or even a histogram of each quantified gradient orientation (one channel per orientation) of the input image. ACF uses ten channels – gradient magnitude, HOG (6 channels), and LUV color channels. Each channel is aggregated over blocks to create lower resolution channels. The final classifier is learned by using AdaBoost and depth two decision trees over these channel features.

The implementation of ACF soft-cascade from [10], sets the threshold vector Θ to a fixed linear set of values, i.e., to -1 plus l times a calibration parameter fixed for each dataset. We set c_l to 1 at any cascade level l and refer to c_l as the processing unit for the remaining of this paper. Using the DBP algorithm [32, 13] and the ACF cascade, another cascade variant has been built. We built up two additional soft-cascade variants by using our GLS procedure applied on the two previous variants, using $\delta_{\max} = 4$. This gives a total of four soft-cascade variants for each dataset. We summarize each method below with used labels:

- ACF: The ACF detector as described in [10].
- ACF+GLS: The ACF detector tuned with GLS method with the default threshold vector as initial solution.
- ACF+DBP: The ACF detector tuned with the DBP algorithm.
- ACF+DBP+GLS: The ACF detector tuned with GLS method with the DBP output as initial solution.

All the experimental tests are carried out on an Intel® Core™ i5-4670 CPU 3.4 GHz processor machine with 16GB DDR3 1600MHz RAM memory. No GPU was used.

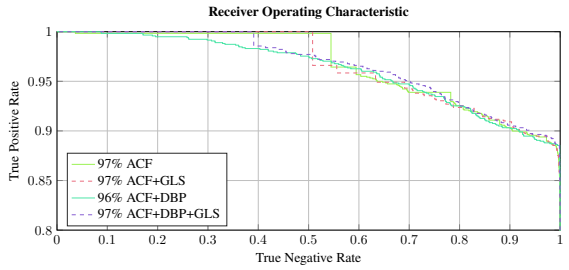

Figure 4. INRIA Dataset Evaluations: Receiver Operating Characteristic Curve in configuration set.

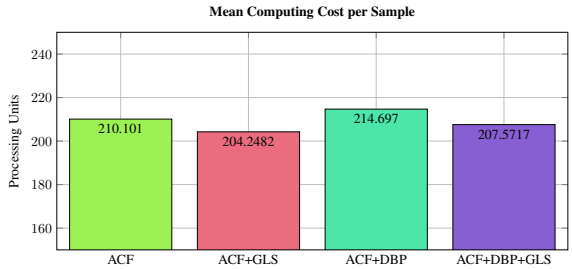

Figure 5. INRIA Dataset Evaluations: Objective Function: Mean Response Time by sample in configuration set.

4.1. INRIA Dataset Evaluations

For the INRIA Person Dataset, the soft-cascades have 2048 stages. The training set is composed of 2474 positive samples and 15710 negative samples extracted from the training images. The configuration set used is composed of 1178 positive samples and 10947 negative samples extracted from test images. This configuration set was used to run the DBP algorithm and the GLS method to generate the four soft-cascade variants: ACF, ACF+GLS, ACF+DBP and ACF+DBP+GLS. In our experiments, we experienced a mean computation time of GLS that equals 1.5 hours during the post-training tuning of the thresholds.

Once the detectors built, per samples evaluations were made in the configuration set. The Receiver Operating Characteristic (ROC) curve in Figure 4 shows the performances of the four soft-cascade variants in the configuration set, which is almost similar for all variants with slightly better performance for ACF detector. Figure 5 details the mean response time evaluations. We observe that, in both cases, GLS variants have a better mean response time than their original counterparts (ACF and ACF+DBP), about 3.06%.

Likewise, per image evaluations [13] were made in the 288 test images from the INRIA Person dataset. The Miss Rate vs. False Positives per Image (FPPI) curve in Figure 6 shows the performances of the four soft-cascade variants in the test set images. These evaluations are consistent with those of Figure 4, showing that our cascade variants have similar classification performance as the ACF cascade

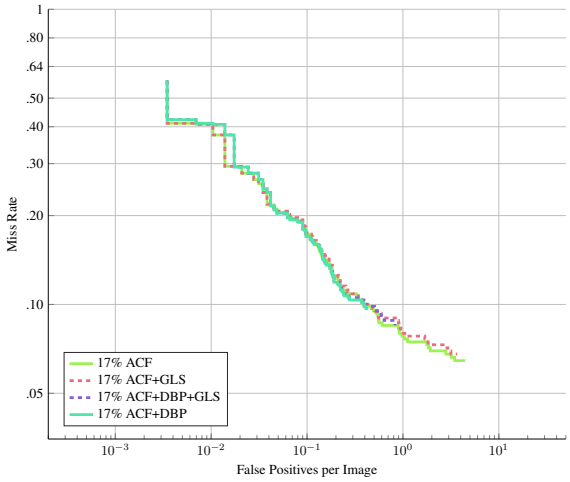

Figure 6. INRIA Dataset Evaluations: Miss Rate vs. FPPI Curve in test set.

Detector	Frames per Second	Log-average Miss Rate
ACF	30.71	**16.83**%
ACF+GLS	**31.86**	17.03%
ACF+DBP	27.10	17.28%
ACF+DBP+GLS	29.42	17.06%

Table 1. INRIA Dataset Evaluations: Log-average Miss Rate vs. Frames per Second in test set.

which is slightly better. Table 1 details the frames per second and per variant. These results are also consistent with those of Figure 5. We observe that ACF+GLS processes 1.15 more frames per second than the original ACF cascade.

We compare our results with the one reported by Cao *et al.* [5] in Figure 7. It is necessary to remark that the two set of evaluations have been made in different computer configurations, which can be seen in the differences between their and ours ACF detector results. We can observe that our detectors are comparable with the results reported by Cao *et al.* [5]. Crosstalk detector has better computation time of 45.40 frames per second followed by our ACF+GLS with 31.86 fps and ACF with a mean of 31.305 fps. The detectors with the best compromise between computation time and detection performance are: Crosstalk, ACF+GLS, ACF, NNNF and SpatialPooling.

4.2. Caltech Dataset Evaluations

For the Caltech Pedestrian Dataset, the soft-cascades have 4096 stages. The ACF soft-cascade variant for Caltech dataset use continuous Adaboost in its learning, which results in weak classifiers that do not return a binary response. Consequently, as the BIP model of Barbosa-Anda *et al.* [1] requires a binary weak classifier response. It cannot be applied directly, neither our procedure that uses that BIP. Therefore, we train a variant of Caltech soft-

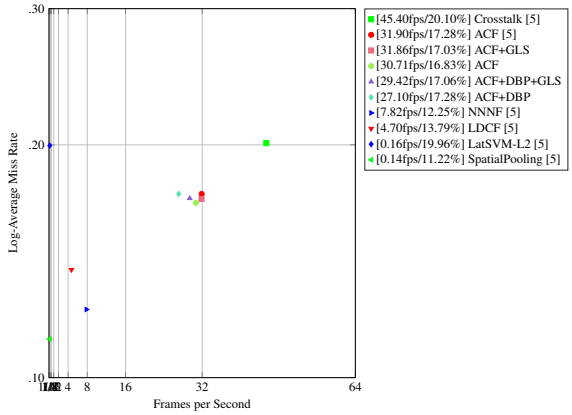

Figure 7. INRIA Dataset Evaluations: Log-average Miss Rate vs Frames per Second comparison with results from [5].

cascade using discrete AdaBoost instead, further referred to as ACF_DISC. The training set for the weak classifiers is composed of 24498 positive samples and 100000 negative samples extracted from the training images. The configuration set for thresholds tuning is composed of 1996 positive samples and 54413 negative samples extracted from the test images. The number of samples being too big for applying directly our proposed approach, we perform beforehand a k-mean clustering of the negative samples with $k = 16120$. This allows us to remove redundant negative samples. The DBP algorithm has been run on the initial configuration set, while the clustered configuration set is used for our GLS approach. This way, four soft cascade variants were generated, further referred to as ACF_DISC, ACF_DISC+CLUS+GLS, ACF_DISC+DBP and ACF_DISC+CLUS+DBP+GLS. For this case, we experienced a GLS mean computation time that approximately equals 10 hours during the post-training tuning of the thresholds.

Given these detectors, per samples evaluations were made on the full configuration set. The Receiver Operating Characteristic (ROC) curve in Figure 8 shows the performances of the four soft-cascade variants in the configuration set, which is again similar for all four variants. Figure 9 details the mean response time evaluations. We observe that the GLS variants have a much better mean response time than their respective original variants (ACF_DISC and ACF_DISC+DBP), about 13.54%.

Likewise, per image evaluations were made in the 4024 test images from the Caltech Pedestrian dataset. The Miss Rate vs. False Positives per Image (FPPI) curve in Figure 10 shows the performances of the four soft-cascade variants in the test set images. These evaluations are consistent with those of Figure 8, showing that our cascade variants have the same performance that the ACF_DISC cascade. It is known that original ACF cascade for Caltech

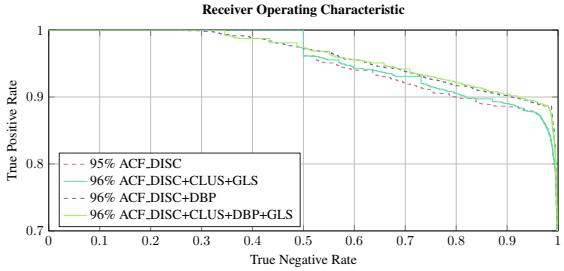

Figure 8. Caltech Dataset Evaluations: Receiver Operating Characteristic Curve in configuration set.

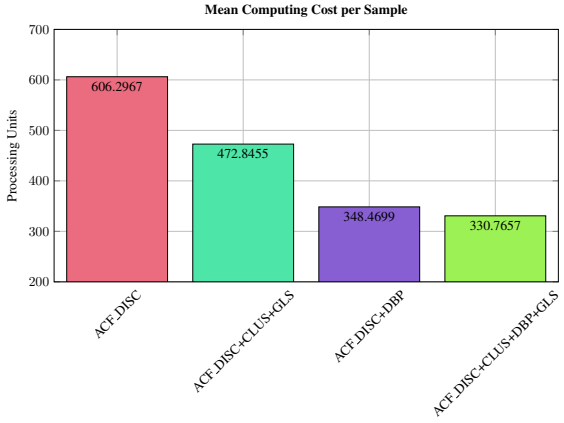

Figure 9. Caltech Dataset Evaluations: Objective Function: Mean Response Time by sample in configuration set.

Detector	Frames per Second	Log-average Miss Rate
ACF_DISC	6.85	**32.50**%
ACF_DISC+CLUS+GLS	**9.88**	32.52%
ACF_DISC+DBP	7.92	32.99%
ACF_DISC+CLUS+DBP+GLS	9.85	32.84%

Table 2. Caltech Dataset Evaluations: Log-average Miss Rate vs Frames per Second in test set.

Dataset, which is trained using continuous AdaBoost, has a slightly better performance. Table 2 details the frames per second per variant. One more time, we observe that ACF_DISC+CLUS+GLS process 3.03 more frames per second that the ACF_DISC cascade.

We compare our results with the ones reported by Cao *et al.* [5] and Zhang *et al.* [33] in Figure 11. Let us highlight that these evaluations were conducted in more powerful computer architectures than our, some of them (CCF, CompACT-Deep and RPN+BF [33]) even using GPU architecture. Nonetheless, we observe that our detectors are competitive with the most powerful ones (according to the literature). Crosstalk detector has better computation time of 14.10 frames per second followed by our ACF_DISC+CLUS+GLS with 9.88 fps and ACF with a mean of 9.49 fps. The detectors with the best compromise

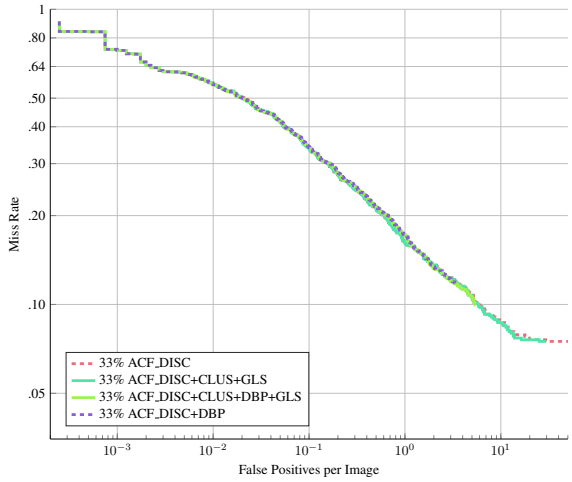

Figure 10. Caltech Dataset Evaluations: Miss Rate vs FPPI in test set.

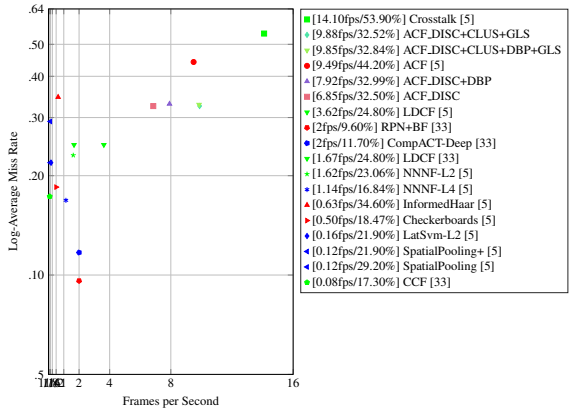

Figure 11. Caltech Dataset Evaluations: Log-average Miss Rate vs Frames per Second comparison with results from [5] and [33].

between computation time and detection performance are: Crosstalk, ACF_DISC+CLUS+GLS, LDCF and RPN+BF.

Evaluations in Caltech Dataset also show that the clustering performed on the configuration set doesn't reduce the final detection performances and allows to apply our algorithm without loss of information. As expected, the clustering concentrates the diversity of the sample set.

The evaluation presented in Sections 4.1 and 4.2 shows that we can improve the computation time of a soft-cascade without decreasing the detection performance. An average computation time of several hours remains acceptable during the post-training tuning stage. The mean improvement in frames per second is 6.15% for INRIA dataset and 34.32% for Caltech dataset. We attribute the variations between the results of INRIA and Caltech datasets to the following differences. Sliding windows in INRIA dataset has a 128×64 size and the cascade length is 2048. In Caltech dataset the window size equals 64×32 and the soft-cascade length is 4096. Consequently, sliding windows technique provides much more samples for the Caltech dataset case than for the INRIA one, which, in combination with the cascade length, tends to decrease the FPS. Therefore, in the case of long size soft-cascades, the soft cascade has a richer search space allowing exploring for best solutions. Our detectors are a good option to keep the computation time acceptable, especially because we overtake classical ACF cascade that has one of the best computation times [5, 6]. Even though we do not use any GPU, we are competitive with some deep-learning based detectors of the literature that are reported to offer an average of 2 frames per second in Caltech dataset [33] with a Tesla K40 GPU architecture.

5. Conclusion

In this paper, a soft-cascade detector approach which explicitly considers the computation time is presented. The new training method uses a novel local search procedure for threshold tuning that considers a neighborhood structure defines as an envelope around a path. This approach allows to consider long length detector and to deal with large-size training set contrarily to [1]. It can also be used as a post processing phase, when designing traditional soft-cascade detectors, intending to improve their CPU performance. Moreover, it is independent from the feature type or the application context, the only restriction being to use a soft-cascade classifier with binary weak classifiers. On the considered data sets, our detector offers better performances in comparison with other classic soft-cascade detectors, from the point of view of the mean response time criterion. In this work, separate analysis of the computational complexity of the features is not necessary as they all the considered feature sets have equal computation cost. Our approach can be extended to heterogeneous features and so heterogeneous computation time should be considered. An approach similarly to Dollár *et al.* [11] can be adopted to determine the computational complexity of the features. With that information, it is still possible the proposed feature selection algorithm, but for improved performance the ordering of the features will have to also be taken into account like [7].

Acknowledgements

We thank the Mexican National Council of Science and Technology (CONACYT) and the French National Center for Scientific Research (CNRS) for their support.

References

[1] F. Barbosa-Anda, C. Briand, F. Lerasle, and A. Mekonnen. Mean response-time minimization of a soft-cascade detector, In *Int. Conf. on Operations Research and Enterprise*

Systems (ICORES'16), Rome, Italy, Feb. 2016, pages 252–260.

[2] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what we have learnt? In *Workshop of Europ. Conf. on Computer Vision (ECCV'14)*, Zurich, Switzerland, Sep. 2014.

[3] L. Bourdev and J. Brandt. Robust object detection via soft cascade, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, San Diego, USA, June 2005, pages 236–243.

[4] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, Online multi-person tracking-by-detection from a single, uncalibrated camera, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI'11)*, 33(9):1820–1833, 2011.

[5] J. Cao, Y. Pang, and X. Li. Pedestrian detection inspired by appearance constancy and shape symmetry, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, USA, June 2016, pages 1316–1324.

[6] J. Cao, Y. Pang, and X. Li, Pedestrian detection inspired by appearance constancy and shape symmetry, *IEEE Trans. on Image Processing*, 25(12):5538–5551, Dec 2016.

[7] C.-H. Chen, T.-Y. Chen, D.-J. Wang, and T.-J. Chen, A cost-effective people-counter for a crowd of moving people based on two-stage segmentation, *Journal of Information Hiding and Multimedia Signal Processing*, 3(1):12–23, Jan. 2012.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, USA, June 2005, pages 886–893.

[9] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT), `https://github.com/pdollar/toolbox`.

[10] P. Dollár, R. Appel, S. Belongie, and P. Perona, Fast feature pyramids for object detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI'14)*, 36(8):1532–1545, 2014.

[11] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification, In *Computer Vision and Pattern Recognition (CVPR'07)*, Minneapolis, USA, June 2007.

[12] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'09)*, Miami, USA, June 2009, pages 304–311.

[13] P. Dollár, C. Wojek, B. Schiele, and P. Perona, Pedestrian detection: An evaluation of the state of the art, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI'12)*, 34(4):743–761, 2012.

[14] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, Object detection and tracking for autonomous navigation in dynamic environments, *Int. Journal of Robotics Research (IJRR'10)*, 29(14):1707–1725, 2010.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, The Pascal visual object classes (VOC) challenge, *Int. Journal of Computer Vision (IJCV'10)*, 88(2):303–338, 2010.

[16] M. Garey and D. Johnson. Computers and intractability: A guide to the theory of np-completeness. New York, NY, USA: W. H. Freeman & Co., 1979.

[17] D. Gerónimo, A. López, A. Sappa, and T. Graf, Survey of pedestrian detection for advanced driver assistance systems, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI'10)*, 32(7):1239–1258, 2010.

[18] Han, Bing and Wang, Xiaoyu, Detection for power line inspection, *MATEC Web Conf.*, 100:03010, 2017.

[19] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'15)*, Boston, USA, June 2015, pages 4073–4082.

[20] L. Jourdheuil, N. Allezard, T. Chateau, and T. Chesnais. Heterogeneous Adaboost with real-time constraints - application to the detection of pedestrians by stereovision, In *Int. Conf. on Computer Vision Theory and Applications*, Rome, Italy, Feb. 2012, pages 539–546.

[21] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, MOTChallenge 2015: Towards a benchmark for multi-target tracking, *ArXiv:1504.01942 [cs]*, April 2015, arXiv: 1504.01942.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. Ssd: Single shot multibox detector, In *Europ. Conf. on Computer Vision (ECCV'16)*, Amsterdam, The Netherlands, Oct. 2016.

[23] A. A. Mekonnen, F. Lerasle, A. Herbulot, and C. Briand. People detection with heterogeneous features and explicit optimization on computation time, In *Int. Conf. on Pattern Recognition (ICPR'14)*, Stockholm, Sweden, August 2014.

[24] S. Ren, K. He, R. Girshick, and S. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI'17)*, 39(6):1137–1149, June 2017.

[25] R. E. Schapire, The boosting approach to machine learning: An overview, *Lecture Notes in Statistics*:149–172, 2003.

[26] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection, In *Int Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, Boston, USA, June 2005.

[27] M. Teutsch and W. Krüger. Robust and fast detection of moving vehicles in aerial videos using sliding windows, In *Workshop in Computer Vision and Pattern Recognition (CVPR'15)*, June 2015, pages 26–34.

[28] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks, In *Int. Conf on Computer Vision and Pattern Recognition (CVPR'15)*, Boston, USA, June 2015, pages 5079–5087.

[29] R. Varga and S. Nedevschi. Robust pallet detection for automated logistics operations, In *Int. Conf. on Computer Vision Theory and Applications (VISAPP'16)*, Rome, Italy, Feb. 2016, pages 470–477.

[30] P. Viola and M. Jones, Robust real-time face detection, *Int. Journal of Computer Vision (IJCV'04)*, 57(2):137–154, 2004.

[31] R. Xiao, L. Zhu, and H. Zhang. Boosting chain learning for object detection, In *Int. Conf. on Computer Vision (ICCV'03)*, Nice, France, Oct. 2003.

[32] C. Zhang and P. Viola. Multiple-instance pruning for learning efficient cascade detectors, In *Neural Information Processing Systems (NIPS'08)*, Columbia, Canada, Dec 2008, pages 1681–1688.

[33] L. Zhang, L. Lin, X. Liang, and K. He. Is faster R-CNN doing well for pedestrian detection? In *European Conf. on Computer Vision (ECCV'16)*, Amsterdam, The Netherlands, Oct. 2016, pages 443–457.

[34] M. Zhang and R. Alhajj. Content-based image retrieval: From the object detection/recognition point of view, In *Artificial Intelligence for Maximizing Content Based Image Retrieval*, ser. PA: Information Science Reference, Z. Ma, Ed., Hershey, 2009, pages 115–144.

[35] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, USA, June 2016, pages 1259–1267.

[36] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'15)*, Boston, USA, June 2015, pages 1751–1760.

[37] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients, In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'06)*, New York, USA, June 2006.