Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Ingrid Visentini · Lauro Snidaro · Gian Luca Foresti

Cascaded Online Boosting

Received: date / Revised: date

Abstract In this paper, we propose a cascaded version of the online boosting algorithm to speed-up the execution time and guarantee real-time performance even when employing a large number of classifiers. This is the case for target tracking purposes in computer vision applications. We thus revise the on-line boosting framework by building on-the-fly a cascade of classifiers dynamically for each new frame. The procedure takes into account both the error and the computational requirements of the available features and populates the levels of the cascade accordingly to optimize the detection rate while retaining real-time performance. We demonstrate the effectiveness of our approach on standard datasets.

Keywords Online Boosting, Multiple classifiers systems, Object detection, Tracking

1 Introduction

Classifier ensembles are proved to benefit from the diverse decision capabilities of their members, thus improving classification performance [13]. Boosting [6], bagging and other forms of classifiers combination [26] aim to form an ensemble of weak classifiers that will perform as a strong one [11,28,24,12].

Usually the weak classifiers are trained on the entire training set and then greedily added to the ensemble with respect to a certain criterion; therefore, the iterative process builds the set of weak classifiers that performs best on the training set. This batch procedure is particularly demanding in terms of computational resources and is dedicated to systems that can afford a time-consuming training phase [15,32]. A new stimulus has been given by the intuition of Oza [21] that conceived online versions of the Bagging and Boosting algorithms. In particular, the Online Boosting algorithm has found many applications in pattern recognition. For example, Grabner and Bischof [10] showed how to apply this framework to computer vision tasks. Pham and Cham [25] proposed an asymmetric version of the original algorithm, to cope with unbalanced classes. Another interesting approach exploits the Wald's sequential decision theory within the Boosting algorithm [31]; in a later work, this idea has been applied to online learning to achieve a better compromise between complexity and accuracy with respect to Online Boosting [9]. The online WaldBoost modifies the cardinality of the ensemble depending on the classification task; when the cardinality is reduced a speed-up is achieved. As will be discussed later, the proposed method has variable accuracy but always maintains the real-time constraint.

Many other recent works point that the Online Boosting is an effective framework for combining multiple classifiers for computer vision applications [17,39,35,18].

The most prominent use of Online Boosting for vision is probably object tracking: a combination of classifiers is used to learn the appearance of a target in the current frame and the ensemble is then employed to detect it in the following frame, considering the detection step as a discrimination between the object and the background [23,2]. The location of the target is therefore given by the region that triggers the strongest response of the ensemble.

Since object detection and tracking in video sequences are known to benefit from the employment of multiple (heterogeneous) features (e.g. color, orientation histograms, etc.) as shown in [5,8,36], the combination of different features within the boosting framework can be a winning strategy in terms of robustness and accuracy [42]. However, this can prohibitively increase the computational burden to the point of preventing real-time performance even in the case of Online Boosting. In fact, the computational complexity increases with the number of features/classifiers employed.

To overcome a similar problem in off-line classification, Viola and Jones [33] proposed the idea of building a cascaded classifier to speed-up the application of the

Dept. Mathematics and Computer Science
University of Udine
Udine, Italy
E-mail: {visentin, snidaro, foresti}@dimi.uniud.it

ensemble to many sub-regions of an image during the detection step. They exploited the fact described in [1] that few accurate weak classifiers are sufficient to narrow the focus of attention on the regions of the image where the object is likely to be present. These classifiers would constitute the first level of the cascade while the other levels would comprise the remaining classifiers in descending order of accuracy. This approach has been successfully applied to many tasks, including face detection [38,37], pedestrian detection [40,22], neural network frameworks for classification [7], and handwritten digit recognition [14,41]. From its first appearance in the literature until these recent works, including [3,38,4], the cascade was trained off-line through an extremely time consuming process. An exception is the work of Wu and Nevatia [35], where a cascaded version of the Online Boosting algorithm is proposed, but starting from general seed detectors learned off-line.

To speed up the application of this ensemble, here we propose a cascaded Online Boosting algorithm that builds on-the-fly a cascade from a heterogeneous set of online boosted classifiers. Since different features can have different computational costs, the cascade construction takes into account both the error and the computational requirements of the available features and populates the levels accordingly to optimize the detection rate while retaining real-time performance. We thus exploit the advantages given by the online boosting algorithm while guaranteeing the sought-after property of real-time performance as it is the case in most computer vision tasks.

The idea was initially conceived in [34] for a homogeneous set of features and then extended in [30] to the heterogeneous case. With respect to these previous papers, the present work introduces the following improvements:

- A refined cascade algorithm that doesn't require user pre-defined thresholds
- A novel strategy for hypotheses selection that dynamically takes into account the cost of the features and the current frame rate
- Thorough experimentation on standard datasets

We have compared the performance achieved by the proposed cascaded algorithm with those of the *monolithic* online boosting. In particular, we have focused on computational cost and thus measured the frame rate obtained in object tracking tasks. For completeness, we have also validated the approach by considering tracking scenarios in real-world video sequences.

The paper is organized as follows: Sect. 2 provides a brief recount of the key concepts of Boosting and classifiers ensembles and puts them in the context of object tracking in video sequences; Sect. 3 provides a step-by-step description of the proposed algorithm; Sect. 4 presents the experimental validation on standard datasets of real-world video sequences.

2 Ensembles of classifiers and Booosting

Ensemble methods are well-known techniques for combining multiple classifiers' outputs in order to improve classification performance (see [26] for a recent survey). Ensemble-based (meta) algorithms [26], such as Bagging and Boosting, follow the intuition that fusing multiple weak hypotheses (classifiers) yields a strong ensemble, that is with increased classification performance.

Given a set of (weak) hypotheses $\{h_1, h_2, \ldots, h_T\}$ so that $h_t : X \rightarrow Y$ where X is the set of vector-valued samples, $Y = \{-1, +1\}$ is the set of labels, and $\{(\mathtt{x}_i, y_i) : i = 1, \ldots, N \wedge \mathtt{x}_i \in X \wedge y_i \in Y\}$ is the training set, the Boosting algorithm builds a (strong) classifier ensemble H as follows

$$H(\mathtt{x}) = sign\left[\sum_{t=1}^{T} \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) h_t(\mathtt{x})\right] \tag{1}$$

where t is the current training epoch and T is the total number of epochs (and, at the same time, the total number of weak classifiers). For each t, a hypothesis h_t is greedily added to the ensemble with respect to the probability distribution D_t on all the learning samples in X. The probability distribution D_t, updated at each epoch t, assigns a weight to each training example according to its "difficulty": the Boosting algorithm adjusts the weights in order to focus on the "hard" samples in the training set. The objective is to minimize the weak classifier error ϵ_t, defined as

$$\epsilon_t = P(h_t(\mathtt{x}_i) \neq y_i) = \sum_{i:h_t(\mathtt{x}_i)\neq y_i} D_t(i) \tag{2}$$

In 2001, Oza [21] proposed an online version of the former boosting algorithm due to Schapire and Freund [6]. Two main improvements were introduced: the first one is that the ensemble can be at the same time used for classification and trained "on-the-fly" on every new sample. The second one refers to the way the learning samples are processed. In the off line version, all the samples are available at the same time, and a distribution on the training set is maintained. At each boosting round t the algorithm, thus, modifies the weights of each sample according to the error of the base learner h_t.

On the contrary, in the online version no distribution on the samples is available, and the information on the training set difficulty has to be embedded in the hypotheses. In particular, a value λ associated to each incoming sample reflects the difficulty of the entire hypotheses set to classify it [20]. The error of every hypothesis h_m is conditioned by the correctly recognized samples weight $\lambda_m^{sc} = \lambda_m^{sc} + \lambda$ and the misclassified samples weight $\lambda_m^{sw} = \lambda_m^{sw} + \lambda$, so that

$$\epsilon_m = \frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}} \tag{3}$$

Eventually, the ensemble output is given by:

$$H(\mathtt{x}) = \arg\max_{y \in Y} \left(\sum_{m=1}^{M} log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) h_m(\mathtt{x}) \right) \quad (4)$$

where M is the number of weak learners fixed a priori, and ϵ_m is the error of hypothesis h_m on the training set defined as above.

When ensemble classifiers are used online for target tracking [10,20], there is no training set in the traditional sense. Instead, samples arrive sequentially in time and they are processed (learned) one by one and then discarded. In particular, in our case at each time t the ensemble learns a positive and a negative sample. Let $\mathtt{x}$ be the subregion of the image I_t that has been classified by ensemble H as the target. Then $\mathtt{x}$ is considered as the new positive sample to be learned. The negative sample, on the contrary, is a background patch (negatively labelled) $\mathtt{x}_b$ chosen at a random position in the image I_t with a preference for the area surrounding the target.

Considering every hypothesis h_m as a Naive Bayes classifier that discriminates between the background and the target, the outputs of the ensemble over the positive (target) and negative (background) samples can be modelled by two normal distribution $\mathcal{N}(\mu_{m,y}, \sigma^2_{m,y})$ where $y \in Y = \{-1, 1\}$. The highest posterior probability given by the Bayesian classifier on a sample $\mathtt{x}$ determines the belonging class (Maximum a Posteriori)

$$h(\mathtt{x}) = P(y|\mathtt{x}) \propto \arg\max_{y \in Y} P(y)\mathcal{N}(\mathtt{x}, \mu_{m,y}, \sigma^2_{m,y}|y) \quad (5)$$

To learn a sample $\mathtt{x}$, the parameters of the distributions at time t are modified as follows

$$\mu_{m,y,t} = (1-\rho)\mu_{m,y,t-1} + \rho h(\mathtt{x}) \quad (6)$$

$$\sigma^2_{m,y,t} = (1-\rho)\sigma^2_{m,y,t-1} + \rho(h(\mathrm{x}) - \mu_{m,y,t})^2 \quad (7)$$

where ρ is a learning rate parameter.

3 Processing steps

This section provides a step-by-step description of the processing phases required to build on-the-fly the proposed online boosted cascaded ensemble of classifiers.

The principle behind a cascade of classifiers is quite simple: the hypotheses are organized in a pyramidal fashion, often called cascade of attention, that is composed of several levels that follow a coarse-to-fine strategy, as illustrated in Fig. 1. The upper levels, that comprise a small number of classifiers, are applied first. The job of the upper level is to provide a quick response with high sensitivity (few false negatives). In terms of object tracking this means that the levels are not likely to miss the object to be tracked but also a lot of background is probably going to be considered similar to the target. The deeper the level, the more populated is the ensemble, the more confident the classifier, and the more operations are required.

In [33] a single level of the cascade is built analysing the performances on the training set, and using a threshold (e.g. on false positives) to determine the stopping point. As already mentioned, while the classical Boosting algorithm maintains a distribution on the training data according to the "difficulty" of the samples, no distribution is available in the case of online Boosting since new data is continuously streaming in (i.e. new frames from a video sensor) [20].

We propose to exploit the error associated to each hypothesis (3) at time t to estimate the accuracy of the classifier at time $t+1$. In particular, a weak learner with a low error has, by definition, a low rate of false negatives and false positives. We can suppose then that a low error weak learner at at time t is likely to perform well at time $t+1$. We suppose also that the object we are searching for maintains a similar appearance in two subsequent images which is a reasonable assumption given the high input frame rate.

Moreover, when the ensemble is composed of different types of classifiers or features as weak base learners, every one of them can take a different amount of operations to be applied; for this reason, a cost factor γ_m that indicates the time for a hypothesis h_m to be computed has to be considered.

Following this reasoning, in the first levels of our cascade should be placed the hypotheses with low error and low computational cost, in order to reject the highest number of regions not containing the object (True Negatives) while preserving a low False Negatives rate and maintaining the real-time constraint.

3.1 Cost factor

An important consideration when structuring a framework that involves several types of hypotheses to be applied in real-time is the different amount of operations every single weak learner requires in practice to better distribute the classifiers among levels.

To each hypothesis h_m we assign a weight $0 \leq w_m \leq 1$ that denotes its "importance"

$$w_m = 1 - \frac{\epsilon_m + \gamma_m}{\epsilon_m \gamma_m + 1} \quad (8)$$

and comprises the error ϵ_m and the computational cost γ_m. By means of this value we can build the cascade including in the first levels the classifiers with a high weight, to provide a good mix of accuracy and speed. We then normalize all the w_m so that $\sum_m w_m = 1$

The cost γ_m related to hypothesis h_m in (8) can considerably vary depending on the kind of classifiers involved: Haar features are computationally less expensive, for example, than colour histograms. The costs are determined by an off-line procedure that times the process

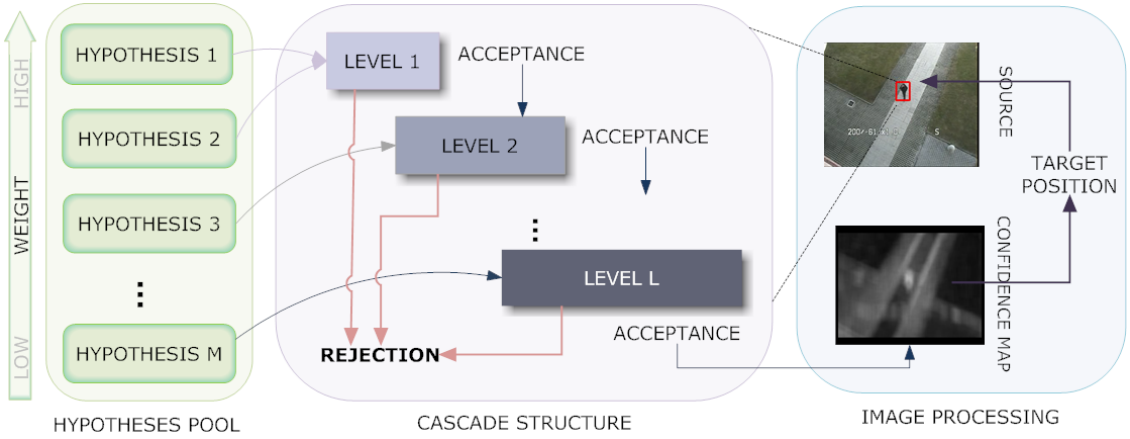


Fig. 1 Architecture of the proposed attention cascade algorithm.

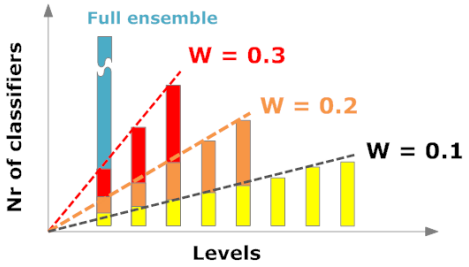


Fig. 2 Illustration of the distribution of the classifiers in the levels of the cascade when varying the parameter W.

of extracting and probing each feature, thus avoiding the need to set them manually. This set-up phase is independent of the video sequence used and also of the size of the target, as these features can be computed in constant time through fast data structures as integral images and integral histograms. After probing the different computational times required by each feature type, their cost is obtained by dividing by the slowest time. In this way, cost values will be normalized so that $0 < \gamma_m \leq 1$, where the slowest feature type will have a gamma value equal to 1.

Considering an upper limit $0 \leq W \leq 1$ for the summation of the weights of the classifiers for each level l, the strong classifier at level l is

$$H_l = \{h_1, \ldots, h_k\} : \sum_k w_k \leq W \tag{9}$$

We notice that the higher W the more classifiers are included in the first levels. The cascade in this case is shorter, as shown in Fig.2, and the time of computation is extended: to search the object of interest, more operations have to be done in the first levels on all the subregions of the image.

3.2 Forcing the Real-time

Since real-time execution is the goal of the proposed approach, we have decided to consider the frame rate as a stakeholder in the process of building the cascade. Exploiting the fact that W can modify the number of levels and thus the shape of the cascade (Fig. 2), we tune the level limit W to reflect the need for speed. As we know that W influences the number of features to be included in a level, if the cascade has been too slow at time t, the shape of the cascade should be revisited and stretched: W_{t+1} should be a more tolerant limit, therefore with a smaller value.

Said FR_{opt} the desired frame rate, and FR_t the current frame rate at time t, W_{t+1} becomes

$$W_{t+1} = W_t * \frac{FR_t}{FR_{opt}} \tag{10}$$

Note that values of $W > 1$ have no meaning since the upper limit is $W = 1$ which means that the entire set of classifiers is comprised in a single level (monolithic ensemble). With respect to our previously adopted solution, where two thresholds for the error rate and the cost rate were used to build the cascade levels, here we use only the frame rate to construct the cascade. To be more precise, the approach is now almost threshold-free since the current frame rate FR_t can be read from the system, the target frame rate FR_{opt} is generally set to 25 fps, and the initial value of W can be randomly initialized or hard-coded to any default value in $(0, 1)$ (e.g. 0.5) and then the algorithm will adjust it automatically via (10) to achieve the required frame rate FR_{opt}. That is, the only input required from the user is the required frame rate FR_{opt}. With this new solution, the real-time constraint is enforced, but at the same time the classifiers of each level provide a compromise between accuracy and maximum number of operations per frame, so that the total computational cost of the level can be limited and the error is kept as low as possible.

Algorithm 1: Cost Cascaded on-line boosting

Require: Strong classifier H_0 randomly initialized
Require: Cascaded classifier H_{out} initialized
Require: Cost value for each hypothesis type in H_{out}
for the new frame at time t **do**
  // Train H_t with the target (positive sample)
  $\mathtt{x}_t^+ \leftarrow \arg\max_{\mathtt{x}}(H_{out}(\mathtt{x})), \mathtt{x} \in I_t$
  $\lambda \leftarrow 1$
  OnlineBoosting$(H_t, (\mathtt{x}^+, 1), \lambda)$ [20]
  // Train H_t with the negative sample(s)
  for each negative training sample $\mathtt{x}_t^-$ **do**
    $\lambda \leftarrow 1$
    OnlineBoosting$(H_t, (\mathtt{x}_t^-, 0), \lambda)$ [20]
  end for
  // Calculate the weight w_m for every classifier
  for $h_m \in H_{aux}, m = 1, 2, \ldots, M$ **do**
    $w_m \leftarrow (1 - \frac{\epsilon_m + \gamma_m}{\epsilon_m \gamma_m + 1})$ as in (8)
  end for
  // Normalize the weights w_m
  for $m = 1, 2, \ldots, M$ **do**
    $w_m \leftarrow \frac{w_m}{\sum_m w_m}$
  end for
  // Sort the weak classifiers w.r.t. their weight (w)
  // in descending order
  $H_{aux} \leftarrow \text{Sort}(H_t)$
  // Build the cascade with the hypotheses in H_{aux}
  $l \leftarrow 0$
  while $H_{aux} \neq \emptyset$ **do**
    $weight \leftarrow 0$
    for $h_m \in H_{aux}, m = 1, 2, \ldots, M$ **do**
      if $weight \geq W$ **then**
        // Initialize a new level
        $weight \leftarrow 0$
        $l \leftarrow l + 1$
        break;
      else
        // Add a hypothesis to the current level
        $weight \leftarrow weight + w_m$
        $H_{aux} \leftarrow H_{aux} - h_m$
        $H_l \leftarrow H_l \cup h_m$
      end if
    end for
  end while
  $L \leftarrow l$
  $H_{out} \leftarrow \bigcup_{l=1}^{L} H_l$
end for

Algorithm 1 summarizes the steps to be taken to construct the cascade. First of all, at time t the hypotheses are trained both with a positive and a negative sample; the error of the boosted classifiers is multiplied with the cost coefficient, providing a weight for each hypothesis. The classifiers are sorted in descending order of weight, and organized by levels that are partitions of the original set. In particular, for every level they are included until the W limit is reached, considering all the weights of the classifiers in a level. The process is repeated by filling the next levels with the remaining base learners. After the levels are completed, their concatenation forms the cascade, which is applied on the image at time $t+1$ and the process is repeated.

Algorithm 2 describes the processing steps for the application of the cascade ensemble. The cascade obtained

Algorithm 2: Application of the cascade.

Require: Cascaded classifier H_{out}
Require: Subregion $\mathtt{x}$
  // Sum of the weights of the (previous) levels
  $wHist \leftarrow 0$
  // Sum of the weights of the current level
  $wCurr \leftarrow 0$
  // Cascade output
  $output \leftarrow 0$
  for $l = 1, 2, \ldots, L$ **do**
    // Probe hypotheses of level l
    for $k = 1, 2, \ldots, K_l$ **do**
      // Update the classifier output
      $output \leftarrow output + \beta_{k,l} * h_{k,l}(\mathtt{x})$
      $wCurr \leftarrow wCurr + \beta_{k,l}$
    end for
    // Get the weights of the hypotheses
    $wHist \leftarrow wHist + wCurr$
    if $((output < 0) \vee (l = L))$ **then**
      // Return the normalized confidence
      // with which to fill the confidence map
      $output \leftarrow \frac{output}{wHist}$
      break;
    end if
  end for

at time $t-1$ processes every subregion in a frame at time t, assigning them also a confidence value. Every sample can be rejected at any level or sent further to the next ones; if the bottom of the cascade is reached and the final output is positive, the object is flagged with the positive label. If the object is rejected at any level, the normalized weighted outputs of the classifiers determines the confidence of the cascade on the subregion; a confidence map keeps record of the decisions of the ensemble.

3.3 Cascade confidence

For what concerns the output of the cascade, we consider both the answer of the classifier set H_l at level l and the output of the previous levels recursively. On a subregion $\mathtt{x}$ of the image, the confidence of the ensemble at level l is defined as the sum of the confidence of the previous level and the response of the K_l weak classifiers in H_l, as given by the following formula

$$conf_l(\mathtt{x}) \equiv conf_{l-1}(\mathtt{x}) + \sum_{k=1}^{K_l} \beta_{k,l} h_{k,l}(\mathtt{x}) \tag{11}$$

where the confidence of the first level is defined as

$$conf_0(\mathtt{x}) \equiv \sum_{k=1}^{K_0} \beta_{k,0} h_{k,0}(\mathtt{x}) \tag{12}$$

and the coefficients $\beta_{k,l}$ are given by

$$\beta_{k,l} = log\left(\frac{1 - \epsilon_{k,l}}{\epsilon_{k,l}}\right) \tag{13}$$

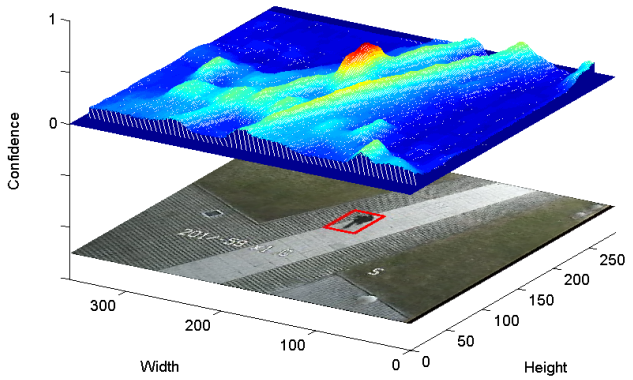


Fig. 3 Example of a confidence map on a frame; the highest peak represents the position of the target.

as in the Boosting weighting scheme. The smaller the error of $h_{k,l}$ on the training samples, the larger the coefficient $\beta_{k,l}$ assigned to it. The number of classifiers K_l in level l is $|H_l|$, where the operator $|.|$ provides the cardinality of a set, and it follows that $\sum_l K_l = M$.

As detailed in Algorithm 2, in the training phase the weak learners are updated with the Boosting algorithm. At classification (probing) time, if the ensemble H_l at level l validates the subregion $\mathtt{x}$, establishing that

$$\text{sign}(\text{conf}_l(\mathtt{x})) = +1 \tag{14}$$

then the sample is evaluated by level $l+1$. The output of a level is based on the response of the previous levels and on its hypotheses one, as per (11). The output of the cascaded ensemble H_{out} on the input $\mathtt{x}$ is thus given by:

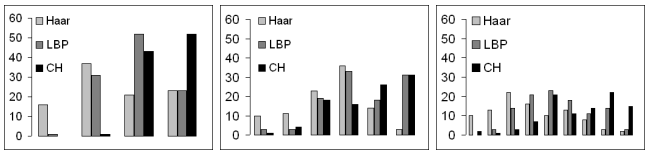
$$H_{out}(\mathtt{x}) = \text{sign}\left(\text{conf}(\mathtt{x})\right) \tag{15}$$

where

$$\text{conf}(\mathtt{x}) = \sum_{l=1}^{L}\sum_{k=1}^{K_l} \beta_{k,l} h_{k,l}(\mathtt{x}) \tag{16}$$

Eventually, considering the answer of the classifier on every subregion of the image a confidence map is built. The target position corresponds to its maximum, and can be found by a simple Max function. An example of how the confidence map looks like is presented in Fig. 3.

The computational effort while running depends on the amount of hypotheses applied per region. With our solution, not all the hypotheses of the ensemble are necessarily applied at the same time on the region of interest, but possibly only a small subset whose cardinality depends on the thresholds W. The probing time for the strong classifier is $\mathcal{O}(\sum_{l=1}^{L} K_l) = \mathcal{O}(\sum_{l=1}^{L} |H_l|) = \mathcal{O}(|H|)$, since in the worst case all the hypotheses are tested.



(a) (b) (c)

Fig. 6 Examples of population of a cascade of 300 features for (a) $W = 0.6$ (b) $W = 0.4$ (c) $W = 0.2$.

4 Experiments

We have applied the proposed approach to object detection and tracking for video surveillance purposes. In this section, several experiments performed on standard real-world video sequences are presented to validate the proposed framework. The hardware employed in all the tests is an AMD Athlon64 3500+ with 1GB of RAM. All the modules have been implemented in C++ using fast data structures, i.e. integral images and integral histograms, to reduce the computational requirements.

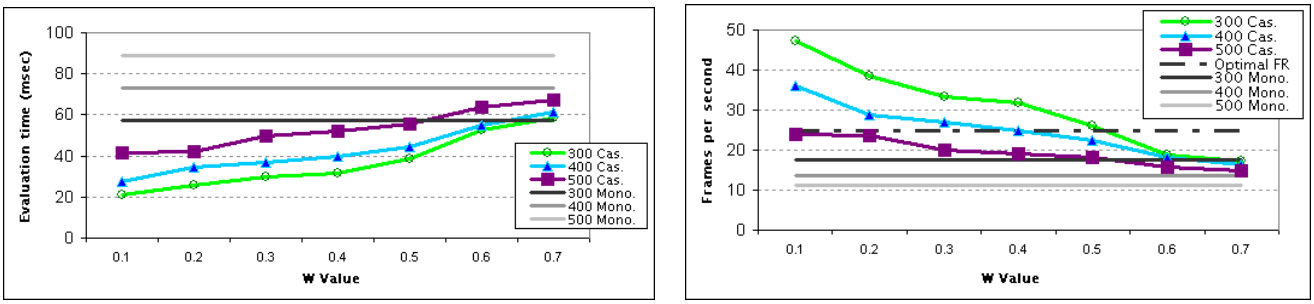
4.1 Settings

We used three different types of features to describe moving objects: Haar features [16], Local Binary Patterns (LBP) [19], and colour histograms [27]. The common start-up procedure for ensemble tracking methods involves that the object to be tracked be set up by a change detection procedure or by hand [20]. Similarly to most of the recent literature [10], we employ a fixed size window to track the object. As a natural evolution of the present work we will consider in the future the application of the cascade to varying scale target tracking.

As regards the ρ parameter for the classifiers update, it was set to 0.25 taking into account the frame rate and the typical movement speed of observed objects [30]; small changes in its value produced no significant effects.

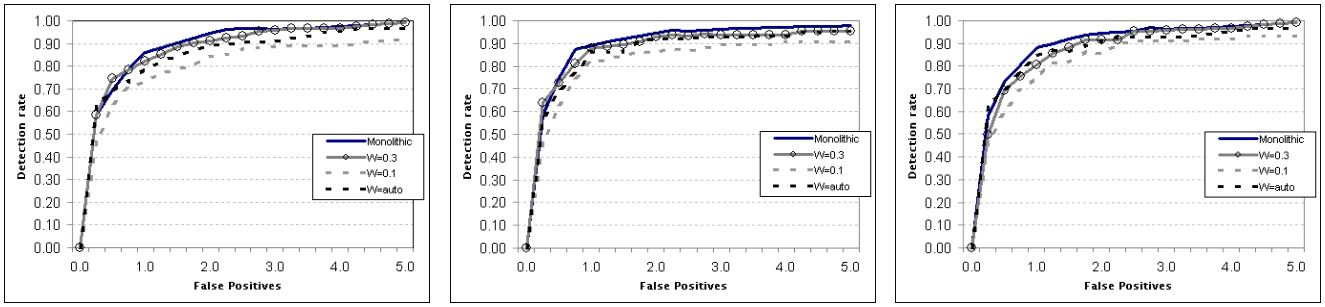
4.2 Comparison of different values of W

First of all, we performed an experiment testing different values of W, since the variation of this parameter leads to a trade-off that involves the time of execution and the accuracy of the ensemble. This experiment wants to compare, in terms of accuracy and speed, our cost-based cascade framework with an ensemble of fixed dimension, called *monolithic*, when tracking is performed on an object of interest. We applied three ensembles, consisting of 300, 400 and 500 heterogeneous classifiers respectively, to a standard sequence taken from [29]. The video comprises 1340 frames at 320×240 pixels resolution in which a puppet doll is moved under a light bulb. The target was manually initialized in the first frame on the puppet's muzzle with a 40×40 pixels area.



(a) (b)

Fig. 4 (a) Average evaluation time (in milliseconds) for the different-sized cascades for increasing W values. (b) Frames per second achieved by the proposed algorithm when varying the number of classifiers involved and W.



(a) (b) (c)

Fig. 5 ROC curves for three proposed cascades in the case of 300 (a), 400 (b) or 500 (c) features involved. The monolithic ensemble is compared with the cascades obtained using $W = 0.1$, $W = 0.3$ and the proposed solution that automatically tunes W.

Fig. 4 shows the average evaluation time and the correspondent frame rate for each ensemble type presented. The evaluation time comprises the time required for building the cascade and the time required for actually probing it on the image. For each ensemble type, the performances of the monolithic and the cascaded versions are shown. As it can be seen, the monolithic version is generally slower than the cascaded counterpart and the performance gap narrows for increasing values of W. That is, the higher the value of W the slower the cascade is as its levels will eventually collapse into one (monolithic classifier). Of course, low values of W yield high frame rates but at the cost of accuracy.

To evaluate the accuracy of the classifiers, we have measured their sensitivity and the specificity on the aforementioned video sequence by varying the acceptance function of the classifiers (14) and plotting ROC curves. Note that the acceptance threshold has been changed only for the sake of the experiment, the focus should be on the effects given by different values of W. In Fig. 5 are shown the ROC curves that indicate the performances of the cost-based cascades; these should be correlated with the results of Fig. 4. Each chart shows the plot of the ROC curves obtained by the monolithic classifier and by three cascades for different settings of W: the first one has $W = 0.1$, in the second one $W = 0.3$, and in the third one W is set to be self-tuning according to the proposed mechanism (10) for a target frame rate $FR = 25$. The two fixed values have been chosen to reflect different possible conditions: the lower W, the stricter is the limit imposed to the number of eligible hypotheses, thus preferring a fast system with more lightweight levels. A medium tolerance threshold W allows a larger amount of hypotheses in the initial levels. The higher W the larger the number of weak learners admitted in the first stages of the computation, thus generating a shorter cascade that requires more operations in the initial stages.

We tested three ensembles composed of 300, 400 and 500 heterogeneous features respectively; considering both the Area Under the Curve (AUC) and the speed, as expected the ensembles that are more accurate are those that require more computational time. In particular, the non-cascaded approach scored the highest accuracy when applied to the Sylvester scenario. But the most accurate system is also the most computationally demanding. The cascade with $W = 0.3$ is slightly less accurate than the monolithic detector, but it dramatically reduces the application time as the number of features grows, incrementing dramatically the frame rate. The same can be said of the other cascades, included the self-adapting one with variable W.

The rationale behind this is that the most expensive features (LBP and colour histograms) are the more robust and accurate, so they better classify the sample but

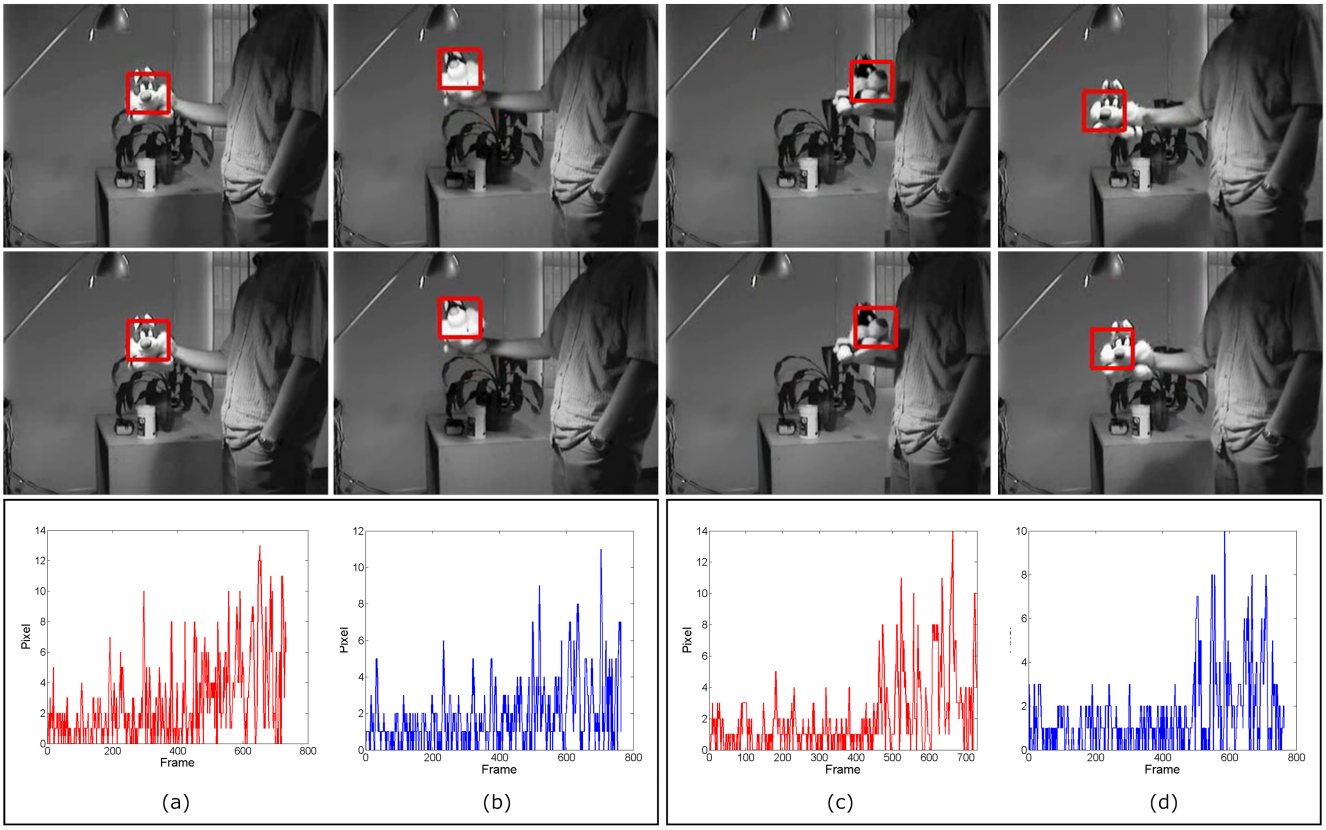


Fig. 7 Results on the Sylvester sequence [29]. The first row shows the monolithic approach, while in the second row the output of the cascaded ensemble with automatic setting of W is displayed. The bottom row presents the shift in pixel with respect to the ground truth for the x (left box) and y (right box) coordinate of cascade [(a) and (c)] set, and monolithic [(b) and (d)]. The cascade detector resulted slightly more sensitive to variations; in the worst case, it showed a 14 pixels shift, while 11 pixels was the maximum offset for the monolithic classifier. The cascaded achieved 25 frames per second (fps), while the monolithic ensemble 15 fps.

they are placed in the low levels of the cascades due to their high cost. For instance, in Fig. 6 the distribution of 300 features through levels for the three different values of W is shown. As we can see, when strict cost constraints are imposed, a small number of "heavy" features is included in the first levels, while lightweight features, even if inaccurate, are preferred. However, while in the monolithic ensemble all the features are applied at the same time, this is avoided when the cascade comes into play, because the levels are applied subsequently when necessary, speeding up the application.

4.3 Comparison with the Online Boosting

In the first two rows of Fig. 7, we can see the output of the monolithic detector and the output of the cascaded detector with automatic setting of W, both comprising 400 features. In the last row, the distance of the static and the cascade detector from the ground truth is presented; the shift is represented by its absolute value in pixels. In the second part of the sequence both detectors have a slight offset with respect to the ground truth. This is due to the wide and sudden changes in both the illumination and in the orientation of the target, and, in particular, due to the transition between frontal view to the full profile or to a top view. The cascade detector resulted slightly more sensitive to variations. In the worst case, the cascade showed a 14 pixels shift, while 11 pixels was the maximum shift for the monolithic classifier. The average error of the monolithic classifier was $4,0$ pixels, while the average error of the cascade was $4,8$ pixels in the experiment.

Regarding the confidence of the cascades compared with the detector, the more the classifiers in the ensemble, the better the confidence value. In Fig. 8 is shown the trend of the confidence values for different ensemble sizes ((a) - (c)) in the case of monolithic and cascade approach. The variation of the parameter W in the case of an automatic setting is presented in Fig. 8 (d). In the first part of the sequence, the object maintains a frontal position and its appearance does not vary noticeably; in the second half, on the contrary the puppet moves rotating (out-of-plane direction), rolling and translating, causing the confidence to decrease.

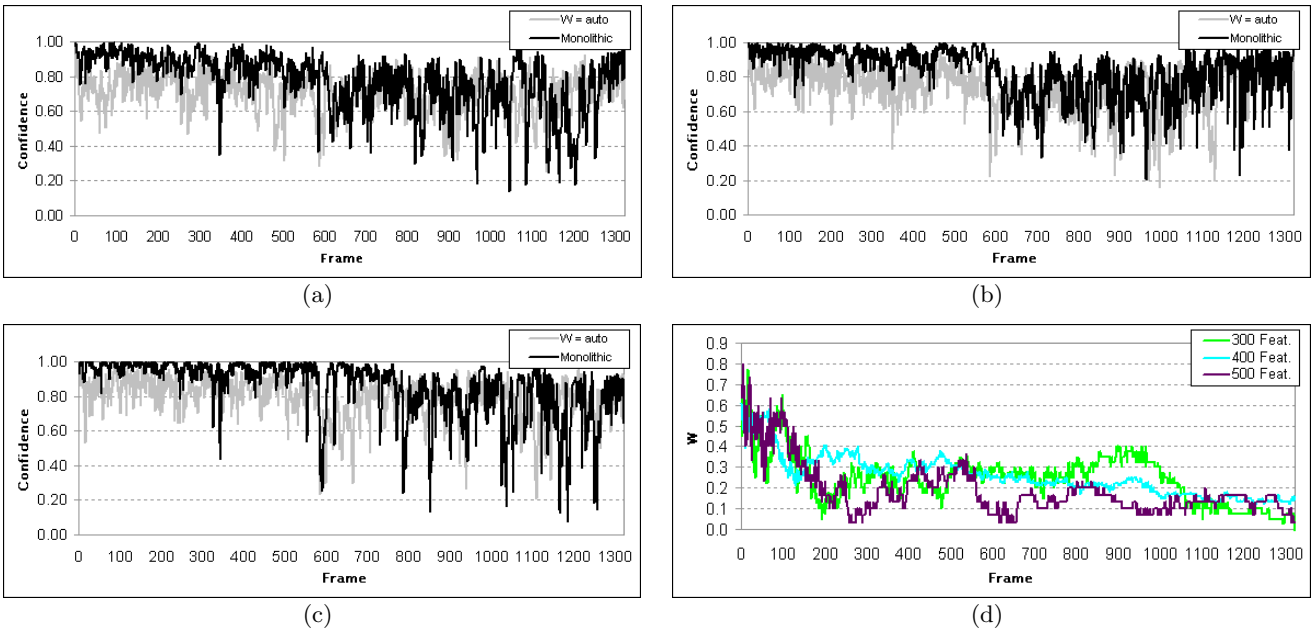


Fig. 8 Confidence values for a 300 (a), 400 (b), and 500 (c) cascade classifier on the video sequence of Fig. 7. (d) Dynamically tuned W values obtained by the proposed cascaded classifier.

Table 1 Results on CAVIAR sequences

Sequence name	#Obj	#Frames	μ COB	σ COB	μ OB	σ OB	fps COB	fps OB
INRIA sequences								
Walk1	1	319	6.30	3.19	4.65	2.34	24.6	18.1
Walk2	1	504	6.42	2.20	5.22	2.28	25.3	18.8
Walk3	1	509	7.39	2.56	5.24	2.18	25.1	18.2
Browse1	1	562	6.13	3.11	4.30	3.45	25.2	18.2
Browse2	1	417	7.39	2.25	5.92	4.44	25.2	18.5
Browse3	1	397	7.18	3.15	6.24	4.82	25.5	18.6
FightRunAway1	2	163	8.52	3.32	7.80	3.58	24.6	18.5
FightRunAway2	1	199	10.89	3.38	8.18	3.28	24.6	18.6
FightOneManDown	3	675	12.24	3.66	10.83	3.11	24.2	18.6
FightChase	1	165	6.58	3.14	5.48	3.03	24.3	18.4
Lisbon sequences								
WalkByShop1cor	4	1718	8.78	3.02	8.18	3.86	24.5	18.2
EnterExitCrossingPaths1cor	1	215	7.56	2.18	7.54	2.48	25.2	18.3
OneLeaveShopReenter1cor	2	278	7.68	2.32	6.74	2.74	25.0	18.3
ShopAssistant1cor	3	443	12.11	3.64	10.54	3.34	24.2	18.3

The cascaded ensemble processed the video at an average of 25 frames per second (fps), as shown in Fig. 9, while the monolithic ensemble performed at an average of 15 fps. The imposed optimal frame rate FR_{opt} was 25 frames per second.

In Tab. 1 are presented the results obtained on several sequences from the CAVIAR [1] dataset. The cascaded Online Boosting (COB) and the Online Boosting (OB) results have been compared in terms of euclidean distance from the ground truth. The mean error (μ COB and μ OB) and its standard deviation (σ COB and σ OB) are reported for both algorithms that are using 400 classifiers each. The data refer to an average of the number of subjects ($\#Obj$) involved, and $\#Frames$ refers to the number of frames of tracking only. We set $FR_{opt} = 25$ and we tracked one target in the scene. The search area was restricted to 35% more than the size of the target. The bounding box in the initial frame has set been manually or via a change detection algorithm.

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

4.4 Multiple targets

To test the performance of the system when more than one object is in the scene, we employed the CAVIAR video sequence dataset to measure the times of computation of the proposed approach based on ensembles of different sizes, compared with the fixed size ensemble.

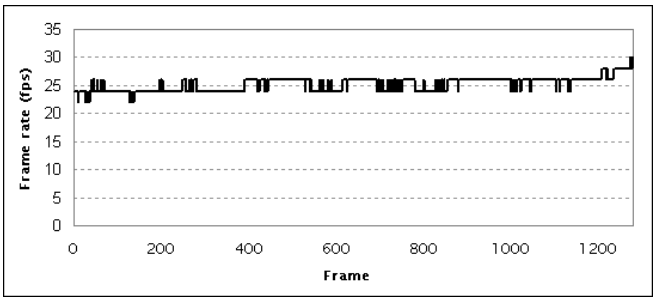


Fig. 9 Frame rate of the cascaded ensemble composed of 400 features. The choice of W is automatic (Fig. 8 (d)) and the optimal frame rate was set to 25 fps.

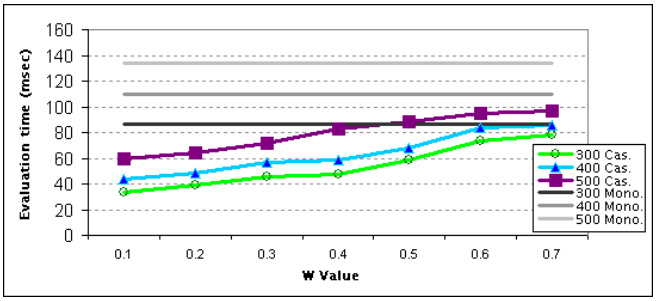


Fig. 10 Average time of computation for the different-sized cascades while increasing W.

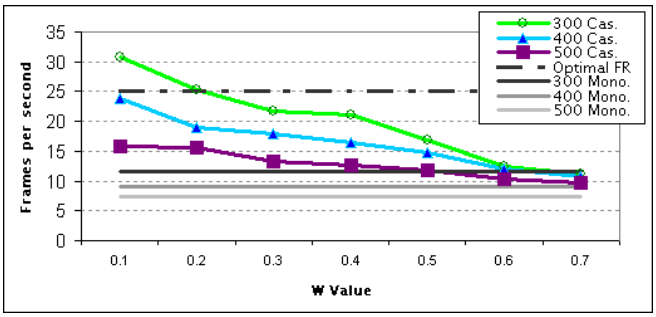


Fig. 11 Frames per seconds achieved by the proposed algorithm when varying both the number of classifiers and W.

We initialized the targets with a simple change detector output, and then the Online Boosting algorithm and the proposed one are employed to track two targets. In this case, a simple background subtraction technique could not be employed to detect the pedestrians during all the frames of the sequence because of the sudden shadows on the floor and because of the illumination changes all around the scene. On the contrary, the detection via classification is more robust and can be applied at each frame.

In Fig. 10 and Fig. 11 the times of computation and the frame rates for different algorithms are presented. Three fixed-size ensembles, consisting of 300, 400 and 500 classifiers, are compared with three cascades of the same size. As predictable, with respect to a single-object scenario the times of computation are doubled; the frame rate is halved, and the cascades remain faster than the non-cascaded ensembles. As we can see, for all the involved algorithms, as W increases the amount of time per operations gets higher; in particular, after a certain value of W the cascades have the same computational cost as the monolithic approach or greater. This can be explained with the fact that in time of evaluation of the cascades is included also the overhead of sorting the classifiers and building the whole structure. Generally, we can state that for high values of W the cascades increase their accuracy but also their computational costs, becoming not preferable to a fixed-size approach because of the overhead of organizing the whole structure.

In Fig. 12 the confidences of the ensembles on the already mentioned video sequence are displayed. Here only the result on the first 550 frames is shown and, for sake of readability, only the confidence of the cascade ensemble consisting of 400 features is presented. In the first part of the sequence a man is walking in the corridor until exiting the scene. Another person comes from the upper part of the image and meets with the previous man re-entering the scene. As we can see in the graph, the confidence of the detector decreases in some frames when there are ambiguities in the scene, as the two men crossing or a variation of appearance or shape. This does not affect the final frame rate, however it slightly affects the accuracy in the detection, as a small shift of few pixels is visible.

With continuous updates and using 400 heterogeneous features to build the cascade, the proposed approach processed the output at about 25 frames per second, as it was the desired frame rate FR_{opt}. The values of W, that is automatically adjusted through the sequence, are presented in Fig. 13. In the first frames W takes high values, decreasing progressively in the next frames. This behaviour allows to build short but accurate cascades at the beginning of the sequence, while at the end of the sequence the cascades are longer and consist of more levels to reduce the computational burden.

5 Conclusions

In this paper, we devised an algorithm that dynamically builds a cascade of classifiers to speed-up the Online Boosting technique. The cascade explicitly considers the computational cost of the involved features to maintain real-time performance. The structure of the cascade and its classifiers are automatically adjusted balancing speed and accuracy. Comparisons with monolithic online ensembles, in terms of accuracy and speed, on standard real-world video sequences have demonstrated the effectiveness of our idea.

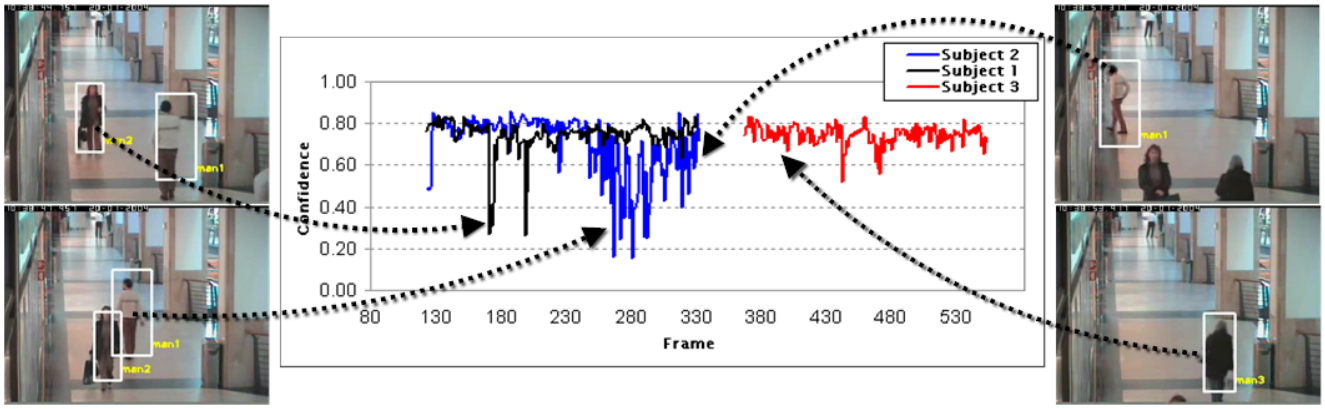


Fig. 12 Confidences of three cascade ensembles consisting of 400 classifiers, each one assigned to a distinct subject in the same CAVIAR sequence. When the two targets cross each other, the accuracy in the detection is affected, and a small shift of few pixels is visible. Generally the confidence of the detector decreases when there are ambiguities, or variations in the appearance of the objects.

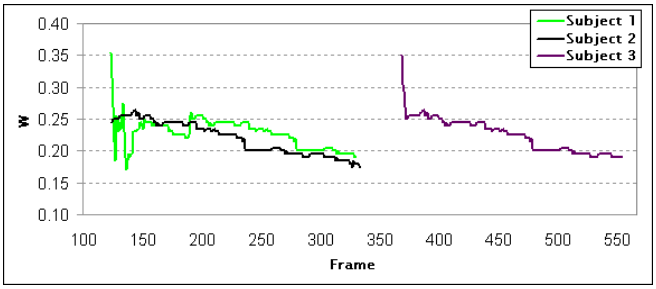


Fig. 13 Values of W in a sample sequence taken from the CAVIAR dataset and referring to the scene presented in Fig. 12. W starts with a high value, and then decreases allowing lighter computational requirements.

References

1. Yali Amit and Donald Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, 1999.
2. Shai Avidan. Ensemble tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):261–271, 2007.
3. Lubomir Bourdev and Jonathan Brandt. Robust object detection via soft cascade. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 236–243, Washington, DC, USA, 2005. IEEE Computer Society.
4. S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1-3):65–86, 2008.
5. Robert T. Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1631–1643, October 2005.
6. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Thirteen International Conference on Machine Learning*, pages 148–156, 1996.
7. N. Garca-Pedrajas, D. Ortiz-Boyer, R. del Castillo-Gomariz, and C. Hervs-Martnez. Cascade ensembles. *Lecture Notes in Computer Science*, 3512:598–603, 2005.
8. Valérie Gouet-Brunet and Bruno Lameyre. Object recognition and segmentation in videos by connecting heterogeneous visual features. *Computer Vision and Image Understanding*, 111(1):86–109, 2008.
9. H. Grabner, J. Sochman, H. Bischof, and J. Matas. Training sequential on-line boosting classifier for visual tracking. In *International Conference on Pattern Recognition*, 2008.
10. Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 260–267, Los Alamitos, CA, USA, 2006.
11. W. Hu, W. Hu, and S. Maybank. Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 38(2):577–583, April 2008.
12. M.M. Islam, X. Yao, S.M.S. Nirjon, M.A. Islam, and K. Murase. Bagging and boosting negatively correlated neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 38(3):771–784, June 2008.
13. J. Kittler, M. Hatef, R. P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.
14. Seong-Whan Lee and Sang-Yup Kim. Integrated segmentation and recognition of handwritten numerals with cascade neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 29(2):285–290, 1999.
15. S.Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1112–1123, 2004.
16. Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 900–903, September 2002.
17. X.M. Liu and T. Yu. Gradient feature selection for online boosting. In *International Conference on Computer Vision*, pages 1–8, 2007.
18. D.D. Masip, Á. Lapedriza, and J.J. Vitri. Boosted online learning for face recognition. *IIEEE Trans. Syst., Man, Cybern. B*, 39(2):530–538, April 2009. Accepted for future publication Systems, Man, and Cybernetics, Part B.
19. T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.
20. Nikunj C. Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340– 2345, Oct. 2005.

21. Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112, Key West, Florida. USA, January 2001. Morgan Kaufmann.
22. Sakrapee Paisitkriangkrai, Chunhua Shen, and Jian Zhang. Fast pedestrian detection using a cascade of boosted covariance features. *IEEE Trans. Circuits Syst. Video Techn.*, 18(8):1140–1151, 2008.
23. T. Parag, F. Porikli, and A. Elgammal. Boosting adaptive linear weak classifiers for online learning and tracking. In *International Conference on Computer Vision and Pattern Recognition*, 2008.
24. D. Parikh and R. Polikar. An ensemble-based incremental learning approach to data fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(2):437–450, April 2007.
25. Minh-Tri Pham and Tat-Jen Cham. Online learning asymmetric boosted classifiers for object detection. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
26. R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, Third Quarter 2006.
27. Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 829–836, Los Alamitos, CA, USA, 2005. IEEE Computer Society.
28. G. Ratsch, S. Mika, B. Scholkopf, and K.R. Muller. Constructing boosting algorithms from svms: An application to one-class classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1184–1199, September 2002.
29. David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2007.
30. L. Snidaro and I. Visentini. Fusion of heterogeneous features via cascaded on-line boosting. In *Proceedings of the Eleventh International Conference on Information Fusion*, pages 1340–1345, Cologne, Germany, June 30th-July 3rd 2008.
31. Jan Sochman and Jiri Matas. Waldboost - learning for time constrained sequential detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 150–156, 2005.
32. P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154, 2004.
33. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, Kauai, Hawaii, December 2001.
34. I. Visentini, L. Snidaro, and G.L. Foresti. On-line boosted cascade for object detection. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, Tampa, Florida, USA, December 2008.
35. B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In *International Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
36. Bo Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
37. J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(3):369–382, 2008.
38. R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *International Conference on Computer Vision*, pages 1–8, 2007.
39. T. Yamashita, H. Fujiyoshi, S. Lao, and M. Kawade. Human tracking based on soft decision feature and online real boosting. In *International Conference on Pattern Recognition*, 2008.
40. J. Yao and J. M. Odobez. Fast human detection from videos using covariance features. In *European Conference on Computer Vision Visual Surveillance workshop (ECCV-VS)*, 2008.
41. Ping Zhang, Tien D. Bui, and Ching Y. Suen. A novel cascade ensemble classifier system with a high recognition performance on handwritten digits. *Pattern Recogn.*, 40(12):3415–3429, 2007.
42. Wei Zhang, Bing Yu, Gregory J. Zelinsky, and Dimitris Samaras. Object class recognition using multiple layer boosting with heterogeneous features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 323–330, Washington, DC, USA, 2005. IEEE Computer Society.