Pattern Recognition 45 (2012) 3141–3153

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Bootstrapping Boosted Random Ferns for discriminative and efficient object classification

Michael Villamizar*, Juan Andrade-Cetto, Alberto Sanfeliu, Francesc Moreno-Noguer

Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, 08028 Barcelona, Spain

ARTICLE INFO

Keywords:
Object detection
Boosting
Bootstrapping
Random Ferns

ABSTRACT

In this paper we show that the performance of binary classifiers based on Boosted Random Ferns can be significantly improved by appropriately bootstrapping the training step. This results in a classifier which is both highly discriminative and computationally efficient and is particularly suitable when only small sets of training images are available.

During the learning process, a small set of labeled images is used to train the boosting binary classifier. The classifier is then evaluated over the training set and warped versions of the classified and misclassified patches are progressively added into the positive and negative sample sets for a new re-training step. In this paper we thoroughly study the conditions under which this bootstrapping scheme improves the detection rates. In particular we assess the quality of detection both as a function of the number of bootstrapping iterations and the size of the training set. We compare our algorithm against state-of-the-art approaches for several databases including faces, cars, motorbikes and horses, and show remarkable improvements in detection rates with just a few bootstrapping steps.

1. Introduction

Object detection and categorization has become a central topic in computer vision research, and recent approaches have shown impressive results under challenging situations such as changes in object appearance, occlusions or pose variations [1–10].

Most of these approaches apply statistical learning techniques over a set of labeled training data to build classifiers which are then used for the recognition task. However, the success of such classifiers is highly conditioned on the quantity and quality of the training data. While this does not generally represent an issue, there are situations in which obtaining training data is not feasible.

A common method to tackle object detection problems with reduced training sets is to introduce additional examples by synthetically generating warped versions of the original images, which can be efficiently represented by means of binary structures such as random trees [11] or Ferns [12]. However, these approaches were conceived to address tracking problems, where only one or very few object classes have to be identified. For problems involving a larger number of classes, a popular approach to handle situations with limited training data is to perform re-sampling over the set of available training examples and use small subsets of data to iteratively refine the classifier. This technique is known as *bootstrapping* and was first introduced in [13], and subsequently followed by many other authors [14–23]. Another alternative is using online learning algorithms that incrementally train the classifier as new data is obtained [24,25]. Yet, while these approaches work remarkably well and allow real-time applications, they are again limited to problems involving a reduced number of object classes.

In this paper we combine ideas of the aforementioned methods to propose a classifier which is highly discriminative, may be computed and evaluated very efficiently, and can handle situations with reduced amounts of training data. For this purpose we initially build a classifier that uses binary object descriptors based on Boosted Random Ferns [26]; and then re-train this classifier following a bootstrapping strategy, which has the effect of increasing the detection rate without requiring further training examples. During this re-training phase we enlarge the size of the positive training set by adding to it warped versions of the object samples that were not detected. If the object is correctly detected, but under a scale factor or offset, it is also added subject to this transformation to the training set. Furthermore, false positive object patches are added to the background training set. All these operations allow the classifier to successively become more dedicated to those challenging samples lying near the decision boundary.

As shown in the results section, just a few (two or three) bootstrapping iterations are enough to ensure a significant improvement of the recognition rates. We demonstrate the advantages of using Bootstrapped Boosted Random Ferns through extensive

* Corresponding author. Tel.: +34 934015806.
E-mail address: mvillami@iri.upc.edu (M. Villamizar).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.03.025

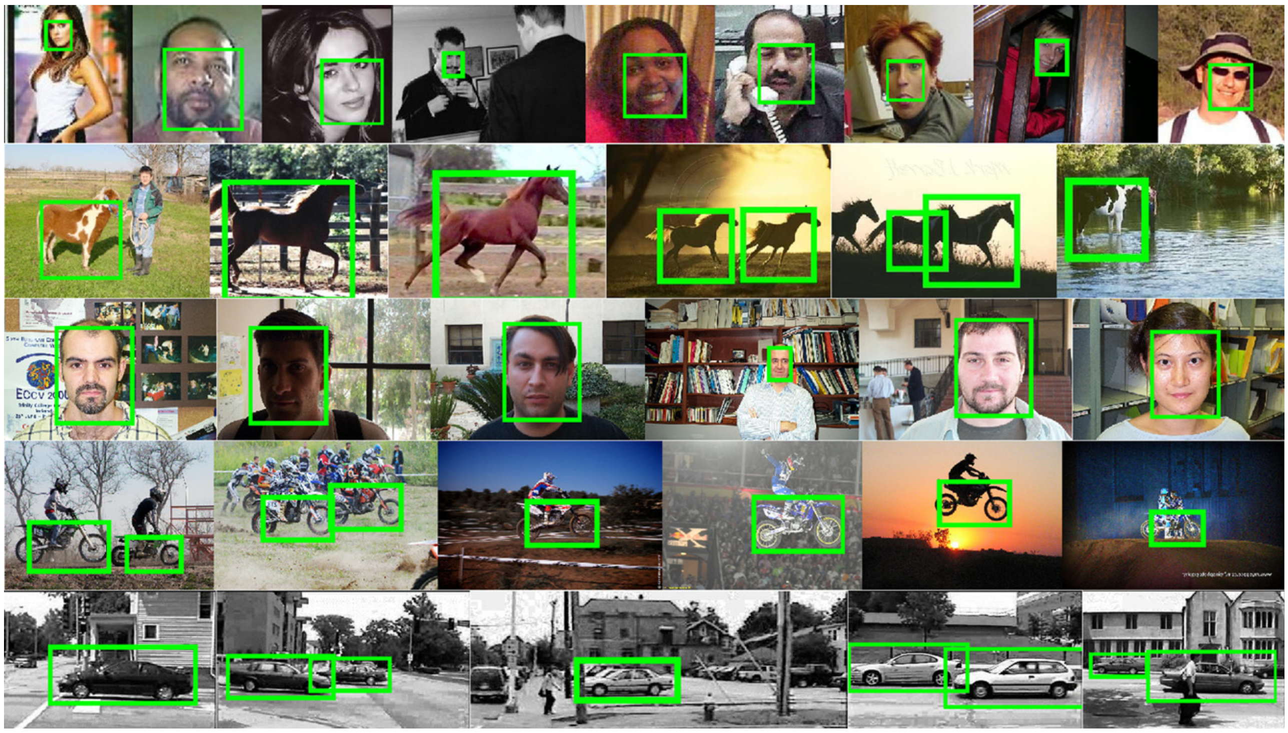

Fig. 1. Examples of detections—localization and recognition—obtained with our approach on several standard databases. We propose a classifier that combines the benefits of binary classifiers, boosting and bootstrapping, that yields high recognition rates despite challenging conditions such as cluttered background, occlusions, scale changes or lighting variations. In addition, it is very efficient, and detection may be performed in about one second per image.

evaluation in several databases (see Fig. 1). Note that although the individual ingredients (Bootstrapping and Boosted Random Ferns) of our work already exist, our main contribution is to bring these pieces together and build a binary classifier which is both highly discriminative and computationally efficient while retaining the simplicity of the original Boosted Random Ferns. This is in contrast to state-of-the-art approaches, which increase the discriminability of simple detectors through relatively complex additional stages which involve solving sophisticated optimization problems, or the iteration over generative and discriminative steps [27–31] that make them computationally expensive. As we will show in the results section, we achieve similar results as these recent approaches, but at a significantly reduced cost.

Another advantage of our approach is that contrary to most other methods, bootstrapping allows to use very generic features, making the technique sufficiently general to be used with very different object classes as shown in the experiments section.

2. Related work

Several machine learning techniques have been proposed over the years to improve the discriminability of object classifiers. A standard approach is to use boosting, i.e., building a strong classifier as a linear combination of weak learners [32–35].

A simple and efficient strategy to improve classifier response is that of bootstrapping, which consists of iteratively re-training the classifier with a training dataset that is progressively augmented by tagging as outliers false positives from previous iterations. The method was originally presented in [13], and has become widely used [14–23], especially when dealing with training sets with a reduced number of samples.

Co-training is another methodology used to automatically build the training set [36]. In this case, the samples used to train a specific classifier are chosen by taking the classification results of a secondary classifier. This approach, though, has the limitation that the positively labeled samples with high confidence for one classifier do not need to be informative for the second classifier. This is addressed by seeking the most informative samples lying at the decision boundary, which is not an easy task. This kind of approach is useful when dealing with a large number of training examples and features, and using all of them becomes prohibitive.

There are other approaches that can be used to accelerate training, typically by either sampling the features or the samples used to train the classifier [37–39]. This, however is not the situation envisioned in the current paper, as we will assume that our training set is limited. Under this situation, bootstrapping appears to be one of the most effective solutions.

The popularity gained by both boosting and bootstrapping lies in the combination of their simplicity and efficiency, while generally guaranteeing state-of-the-art performance. Their joint use has been previously reported for the creation of classifiers based on contour features [17]. In this paper, we propose to build a classifier based on a new type of features coming from Random Ferns computed over histograms of oriented gradients. We will show that appropriately using Adaboost on these features, and learning the parameters of the classifier using bootstrapping we achieve results which are comparable and even better than other current approaches based on far more complex strategies [6,17,31,40–43].

3. Overview of the method

The learning algorithm we describe in this paper combines in an original form the strengths of several existing tools. In particular, it incorporates the following elements:

- *Binary Descriptors*: Inspired on the Ferns descriptor [12], we use a binary representation of image patches based on a small

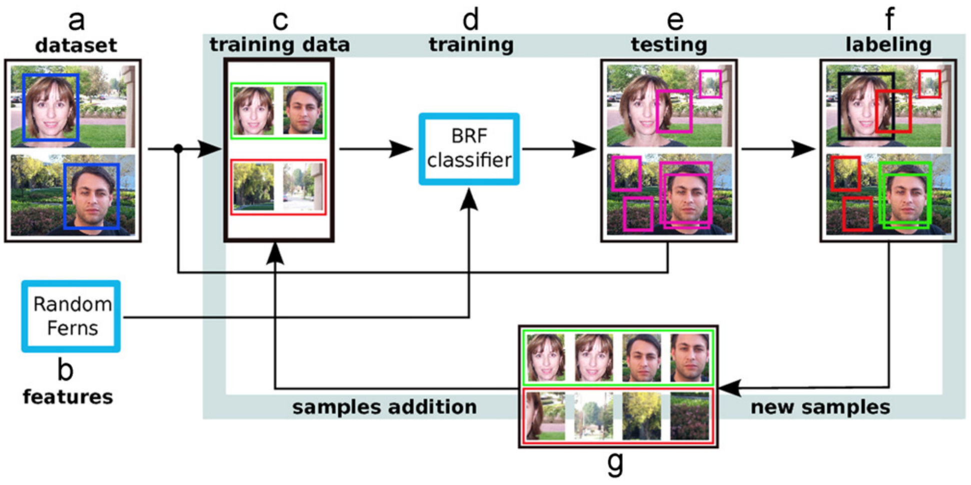

Fig. 2. Our proposed learning algorithm combines binary descriptors, boosting and bootstrapping. We assume we are given a dataset of labeled images with both positive (object) and negative (background) samples (a). We initially compute a set of features, called Random Ferns, which are defined as pairwise random comparisons of bins on local Histograms of Gradients (b). These Random Ferns remain constant throughout the whole learning process. The training data is extracted from the dataset images using the ground truth and an initial set of background images (c). We then use Real Adaboost to build the Boosted Random Ferns (BRF) classifier (d). The training process is iterated according to a bootstrapping strategy, where the classifier is tested over the whole training set (e). Since ground truth is available, the detection results may be labeled as true positives (green), false positives (red) and undetected (black) (f). Finally, warped versions of these undetected patches are appropriately merged to the positive and negative training sets in addition to detected patches for a new training phase (g). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

number of pairwise pixel comparisons. The image position of the pixels is randomly chosen, and hence the name *Random Ferns*. But instead of comparing pixel intensities as in [12,44], we perform comparisons between bins of local Histograms of Gradients (HoGs). This is a key ingredient to increase the generality of the method, as gradients are known to better represent lager intra-class variations than intensities.

- *Boosting*: We consider the Random Ferns computed over HoGs as weak classifiers, which are further linearly combined using Real Adaboost [45] to form a strong classifier. We call the resulting learning algorithm *Boosted Random Ferns* and describe it in [26]. Furthermore, we share Random Ferns across different object categories to reduce the computational load of the method.
- *Bootstrapping*: The previous boosting step is iteratively repeated following a bootstrapping strategy. After each iteration, the current classifier is evaluated over the training set, and warped versions of the misclassified samples are merged to the training data. This extended dataset is then used for a new Adaboost training phase. The result of the bootstrapping process is a classifier more dedicated to the challenging samples, and thus with a smaller global error rate.

The combination of all these ingredients and flow of information during the learning process is depicted in Fig. 2. In the following sections we describe each one of these constituent pieces in more detail. We then evaluate quantitatively the proposed algorithm.

4. Boosted Random Ferns

The features we use to build our weak classifiers are Random Ferns. The original formulation of this descriptor consisted on sets of logical pairwise comparisons on the intensity levels of randomly selected pixels in the input images [12]. Yet, although the use of pixel intensity comparisons yields robustness to illumination changes, the fact that the descriptor is computed on the intensity domain reduces its generalization when comparing objects belonging to the same class but with different appearances. Under these

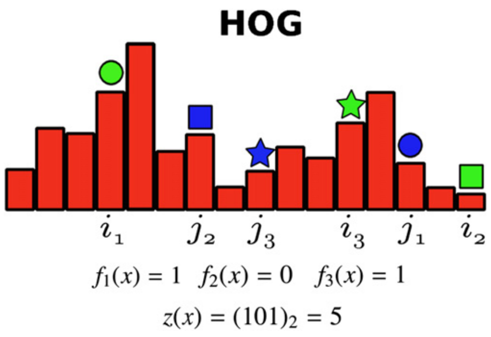

Fig. 3. The Random Ferns are computed over local HoGs. Given a Histogram of Oriented Gradients of a patch centered at a point x, we define the Fern by concatenating the responses of binary functions that compare random bins of the Histogram. The plot shows an example of a 3-dimensional Random Fern. The position of the bins which are compared is randomly chosen during the training stage of the classifier.

variations, gradient information has demonstrated a higher degree of stability. Therefore, following [26], we compute the Random Ferns instead by pairwise comparisons on the bins of local Histograms of Oriented Gradients.

More specifically, let HoG_x be the histogram computed from an image patch centered at x. We define the Random Fern $\mathcal{F} = [f_1(x), \ldots, f_R(x)]^T$ as an R-dimensional binary vector, where $f_k(s)$ is computed as follows:

$$f_k(x) = \begin{cases} 1 & HoG_x(i) > HoG_x(j) \\ 0 & HoG_x(i) \leq HoG_x(j) \end{cases}, \tag{1}$$

and where i and j are random bin locations of the histogram chosen during the training stage. Note that each Random Fern $\mathcal{F}$ captures the co-occurrence of R binary features, and thus, maps image samples x to a $K = 2^R$-dimensional space of binary descriptors.

$$\mathcal{F} : x \rightarrow z \tag{2}$$

where $z = 1,2,\ldots,K$. Fig. 3 shows an example of how a 3-dimensional Random Fern is computed.

In addition, as already proposed in [46], the efficiency of Random Ferns is increased when shared among different classes. That is, the position of the bins used for each Fern comparison is retained, and used for all classes. This, obviously, does not constrain the Fern response for each class to be the same. In the following let us denote by $\vartheta = \{\mathcal{F}_1,\ldots,\mathcal{F}_M\}$ our pool of M Random Ferns.

Class-specific classifiers are then built using a boosted combination of Random Ferns, which play the role of weak classifiers. More specifically, we seek to retrieve the Ferns $\mathcal{F}^t \in \vartheta$ and their image locations g^t to build the classifier $H_{C_j}(x)$ that most discriminates the class C_j from the background B. This classifier is computed with the Real Adaboost algorithm [45], that iteratively assembles T weak classifiers and updates their weighting values so as to successively dedicate to those challenging samples incorrectly classified by previous weak classifiers. Note that the same pool of Random Ferns is shared for the selection of each weak classifier. We therefore define the boosted classifier of class C_j by:

$$H_{C_j}(x) = \sum_{t=1}^{T} h^t_{C_j}(x) > \beta_{C_j}, \tag{3}$$

where x is a sample image, β_{C_j} is a classifier threshold, and $h^t_{C_j}(x)$ is a weak classifier computed by

$$h^t_{C_j}(x) = \frac{1}{2}\log\frac{P(\mathcal{F}^t|C,g^t,z^t(x)=k)+\epsilon}{P(\mathcal{F}^t|B,g^t,z^t(x)=k)+\epsilon}, \tag{4}$$

where $k = 1,2,\ldots,K$, and ϵ is a smoothing factor. At each boosting iteration t, the probabilities $P(\mathcal{F}^t|C_j,g^t,z^t)$ and $P(\mathcal{F}^t|B,g^t,z^t)$ are computed using a distribution of weights W over the training images:

$$P(\mathcal{F}^t|C_j,g^t,z^t=k) = \sum_{\substack{i:z^t(x_i)=k \\ y_i=+1}} W^t(x_i),$$

$$P(\mathcal{F}^t|B,g^t,z^t=k) = \sum_{\substack{i:z^t(x_i)=k \\ y_i=-1}} W^t(x_i), \tag{5}$$

where x_i for $i = 1,2,\ldots,N$ is the set of training samples. In order to choose which is the most discriminative weak classifier $h^t_{C_j}$ at iteration t, we use the common criterion of picking the one that minimizes the following Bhattacharyya distance Q_t between the distributions of the class C_j and background B:

$$Q^t = 2\sum_{k=1}^{K}\sqrt{P(\mathcal{F}^t|C_j,g^t,z^t=k)P(\mathcal{F}^t|B,g^t,z^t=k)}. \tag{6}$$

And finally, once the weak classifier is computed, its classification results are used to update the weight distribution W^t:

$$W^{t+1}(x_i) = \frac{W^t(x_i)\exp(y_i h^t_{C_j}(x_i))}{\sum_{i=1}^{N} W^t(x_i)\exp(y_i h^t_{C_j}(x_i))}. \tag{7}$$

The pseudocode for this whole process is given in Algorithm 1. From the practical point of view, it is worth to mention that in this work all class-specific classifiers are learned using the same parameters. In particular, for all experiments reported in the results section we have used T=300 weak classifiers, a pool of M=15 shared Random Ferns and R=7 binary features per Fern. The size of the fern pool and the number of features per fern set a compromise between recognition rates and computational burden. Our choice of these values is empirical. However,we must stress out that slight variations of these values do not hinder classifier performance,and on the contrary, the generalization of these parameters spreads out for the multiple datasets treated.

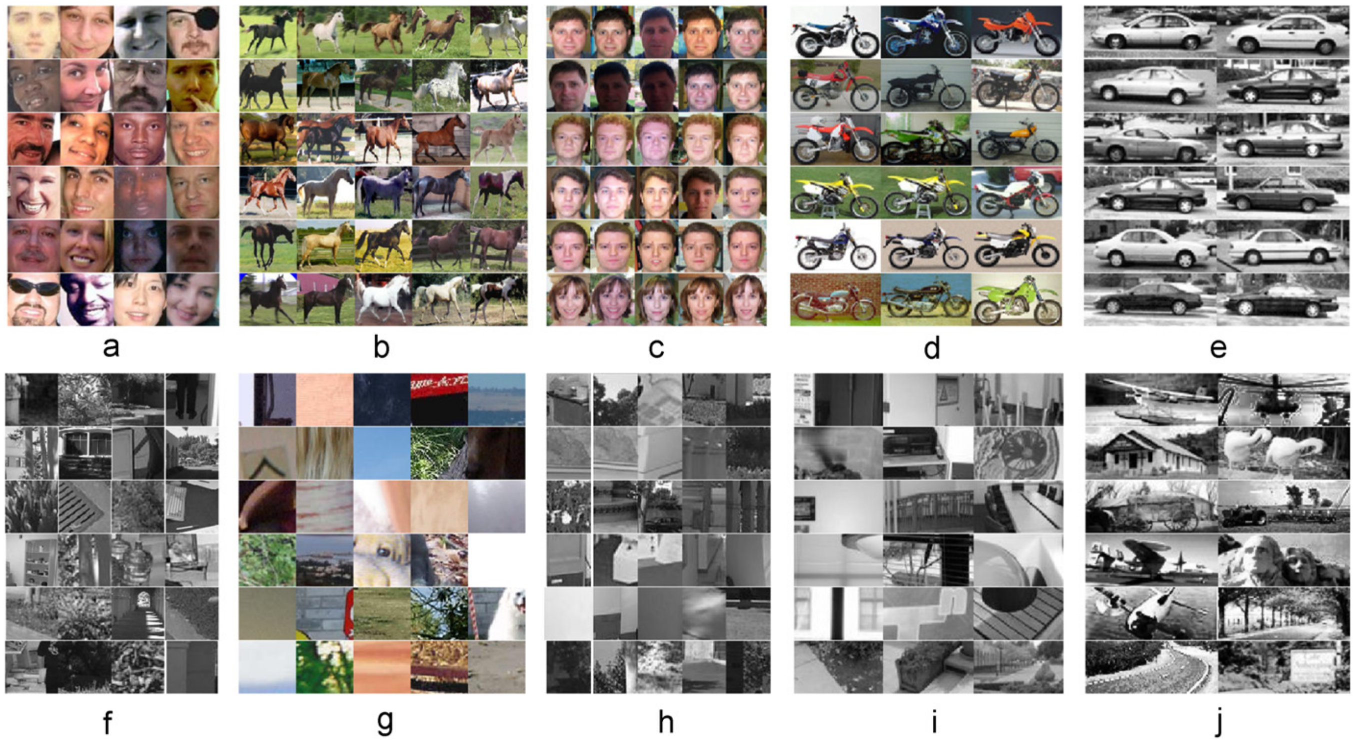

Fig. 4. Samples of the training sets we use for the four object categories. Top: positive (object) samples. Bottom: negative (background) samples. Here, it is important to emphasize that our method works with grayscale images, and more specifically on the gradients domain. In consequence, we use as negative samples the set of background images (grayscale images) given by the Caltech dataset [42] for some object categories. (a) Faces. (b) Horses. (c) Faces. (d) Motorbikes. (e) Cars. (f) Non-faces. (g) Non-horses. (h) Non-faces. (i) Non-motorbikes. (j) Non-cars.

Algorithm 1. Boosted Random Ferns.

```
1: Input:
     • $D_j=\{(x_1,y_1),\ldots,(x_N,y_N)\}$: Training dataset for the class
       $C_j$, where $x_i$ is an image sample and $y_i\in\{+1,-1\}$
       indicates if the sample belongs to the class $C_j$ or
       background $B$.
     • $\vartheta=\{\mathcal{F}_1,\ldots,\mathcal{F}_M\}$: Pool of Random Ferns.
     • $T,\beta_{C_j}$: Number of weak classifiers to be used, and
       classifier threshold, respectively.
2: Initialize the weights distribution over the training
   samples to $W^1(x_i)=\frac{1}{N}$, where $i=1,2,\ldots,N$.
3: for $t$=1 to $T$ do
4:   Use the current weight distribution $W^t$ to pick the Random
     Fern $\mathcal{F}^t\in\vartheta$ and image position $g^t$ that minimizes the
     Bhattacharyya distance $Q_t$ (Eq. 6).
5:   Use $\mathcal{F}^t$ and $g^t$ to compute the weak classifier $h^t_{C_j}$ (Eq. (4)).
6:   Update the weight distribution $W^t$ (Eq. (7)).
7: end for
8: Output: Build the strong classifier using Eq. (3).
```

5. Bootstrapping Boosted Random Ferns

In the previous section we described the class-specific classifier we built using a combination of boosting and binary features computed from a set of labeled training images. We next integrate this classifier within a bootstrapping framework in which warped versions of the training data is sequentially used to re-train the classifier. The advantages of adding a Bootstrapping step to refine the classifier are twofold:

1. It allows learning from databases with small number of samples.
2. It allows to put more emphasis on challenging training samples, and progressively build a classifier with a smaller error rate.

In the following subsections, we will describe each of the building blocks of the bootstrapping scheme we propose.

5.1. Initial training set

For each object class, we assume that a labeled dataset is available, which is made of both positive (object) and negative (background) images. For the positive subset, the object position is defined by means of a rectangular bounding box.

In order to generate the positive training samples from the database, all object images are cropped around their bounding box, aligned, and their sizes normalized. Yet, and in contrast to previous approaches [14], we do not apply any other image warping at this stage, besides the normalization step. As will be explained below, we synthetically warp some of the images during the learning process, but only when they are misclassified. This will allow the final classifier to put more emphasis on these samples.

On the other hand, the negative training sample set is generated by taking random patches from background images, with the same size as for the positive patches. Fig. 4 shows some of the training images we use in our experiments, containing both positive and negative samples.

5.2. Testing the classifier

Given the training sample sets properly cropped onto the object and background regions, we then proceed to build the classifier as discussed in Section 4. Once the Boosted Random Ferns classifier is learned, we evaluate its performance over all images within the dataset. That is, the original and full-size labeled images. During this operation, the classifier is tested at several scales for every image location. This yields a number of potential object detections $\tilde{x}_i$, each of them with an associated confidence level. Fig. 5 shows an example of how this testing is performed. The object classifier (red rectangle) is evaluated at every location and over several HoG resolutions. In order to reduce the computational cost, we compute the HoG pyramid using integral images [1,47].

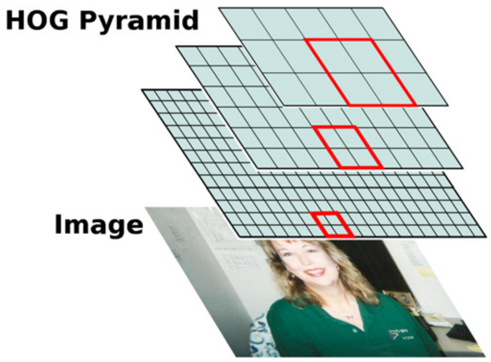

Fig. 5. The classifier—red rectangle—is tested over an HOG pyramid computed from the input image. Integral images allow fast evaluation of the classifier. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

The L object candidates with the largest confidence level are retained and are labeled as either true positive or false positive (negative) samples, based on the degree of overlapping with the ground truth object position. More specifically, an object sample $\tilde{x}_i$ with bounding box $B(\tilde{x}_i)$ is considered a true positive if the ratio of areas

$$r=\frac{|B(\tilde{x}_i)\cap B_{\text{true}}|}{|B(\tilde{x}_i)\cup B_{\text{true}}|} \tag{8}$$

is above a certain threshold $r_{\max}$, and where B_{true} indicates the ground truth bounding box. The candidate is considered as a negative sample otherwise.

In fact, this metric is a standard criterion used to evaluate the performance of classifiers [48], and usually a value of $r_{\max}=0.5$ is chosen. In our results section we will also set $r_{\max}$ to 0.5, except for a specific experiment where we will evaluate the overall performance of our system for different values of $r_{\max}$.

5.3. Generating new training examples

After testing the classifier over the entire dataset, we obtain a set of samples that may be classified as either positive or negative. In addition, we also have a subset of object samples that were not detected. In order to increase the recognition rate of our classifier we use these samples to generate new training images that will be included into the original training set. The criteria we use to build the new training examples are the following:

- Positive samples are included into the original set of positive images, although with the specific scale and offset under which they were detected. This will ensure that the correctly detected samples are not lost in subsequent training stages.
- False positives, i.e., background patches classified as object instances, are reintroduced into the original set of negative images.
- Undetected samples are re-introduced into the original set of positive images. However, to increase the focus of the classifier into these samples, we will produce repeated copies under different affine transformations, including skew and scale changes.

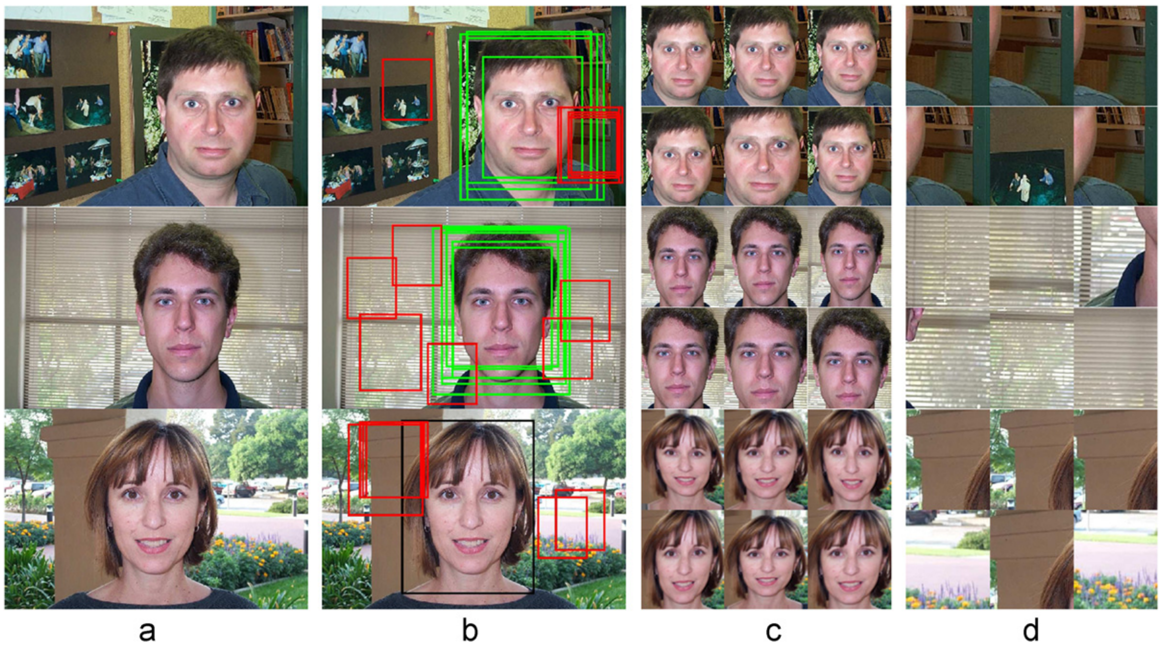

Fig. 6. New training samples collected after one bootstrapping iteration. Positive samples are generated from matching object instances (green boxes) subject to small scale variations and shifts. Negative samples are collected from false positives (red boxes), and include difficult background regions and some partial matches. In the bottom row, the classifier misses the target object (black rectangle) and thus new positive samples are virtually synthesized by applying random affine distortions over the original ground truth image. (a) Images. (b) Detections. (c) Positive samples. (d) Negative samples.

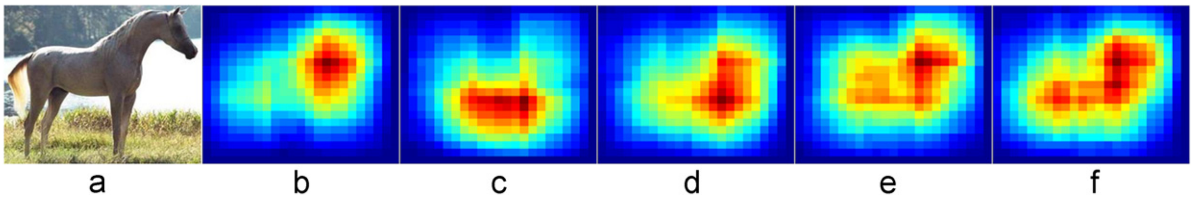

Fig. 7. Spatial distribution of the features during the bootstrapping process. (a) Object category example. (b–f) Spatial layout of Random Ferns for 0, 1, 2, 3 and 4 bootstrapping iterations. Reddish regions indicate higher concentration levels of weak classifiers (Random Ferns). Note that when increasing the number of iterations, the features move towards all of the discriminant features. This yields a classifier with higher recognition rates.

Fig. 6 shows the kind of new training samples generated by our bootstrapping strategy. Positive samples, shown in column (c) in the figure, are generated from matching object instances (green boxes) with the specific scale and shift variations under which they are detected. Negative samples on the other hand, are collected from false positives (red boxes), and are shown in column (d). These are the most valuable additions to the sample set for any bootstrapping algorithm, ours included, since they concentrate on the decision boundary of the classifier. The bottom row in the figure shows an example of misclassification on the target object (black rectangle) by the classifier. In these cases, new positive samples are virtually synthesized by applying random affine distortions over the original ground truth image.

Note that each bootstrapping step increases the size of the training set. In order to keep it to a reasonable size, we limit the number of new training images to the following values: for each image we pick the best $L=6$ object candidates; the positive samples are directly reintroduced into the training set; and for each undetected and negative sample we generate six new training images. With these numbers, initial training sets with about 50 samples, grow up to 600 in two bootstrapping steps. Of course, for larger initial training set sizes, these parameters would change to keep the training dataset tractable.

5.4. *Updating the classifier*

When the Boosted Random Ferns over local HoGs are retrained with the new extended training set, we obtain a classifier more dedicated to the problematic samples. As a consequence, the updated classifier puts more emphasis on the most distinctive parts of the object that let to reduce the global classification error. Fig. 7 illustrates the benefits of the bootstrapping process for the *horse* database. The color coded images represent the distribution of the location of the local HoG Fern-based weak classifiers chosen by our method. Reddish tones indicate higher concentrations of classifiers and bluish colors correspond to lower concentrations. Note that at early stages of bootstrapping, features concentrate on the single most discriminative part of the object around the neck. However, as the number of bootstrapping iterations increases, discriminative features move towards other parts of the horse such as the belly and the head.

The next section extensively evaluates our approach, and shows that only a few bootstrapping iterations are necessary to significantly improve the recognition rate of our classifier.

6. Experimental results

We evaluated our approach on several publicly available datasets including the GENKI face dataset [49], the INRIA horse dataset [50], the Caltech face dataset [42], our IRI freestyle motocross dataset [26] and the UIUC car dataset [30]. As shown in Fig. 4, all these datasets correspond to images of object categories acquired from a single viewpoint, but under large inter-class variations due to scale changes, cluttered background, occlusions, and significantly different lighting conditions.

The validation is performed through different types of experiments, in which we study the performance of the Bootstrapped Boosted Random Ferns (BBRFs) under different parameters, such as the number of bootstrapping iterations, the number of training images or the amount of overlapping $r_{\max}$. Since the features are randomly chosen during training, we perform both training and testing five times, and report the average results.

In the graphs and tables of this section, we will use the following notation: PR: precision–recall; EER: equal error detection Rate; ROC curve: receiver operating characteristic curve; FPPI: false positives per image; BRFs: Boosted Random Ferns; BBRFs: Bootstrapped Boosted Random Ferns.

6.1. GENKI face dataset [49]

Dataset: For our experiments we used the GENKI-SZSL subset, containing 3500 images of faces under different positions and sizes. Although, this dataset originally has a large number of training samples, we just used a small part of them in order to assess the performance of our approach for small training sets. More precisely, we trained the classifier with the first 100 sample images and tested with the remaining 3400 images.

Results: The detection plots, precision–recall curves, of BBRFs for several bootstrapping iterations are shown in Fig. 8 (left). We can observe that the performance of the classifier is improved as the number of bootstrapping iterations increases, achieving maximum classification rate at three iterations and decreasing marginally after that. This is depicted in Fig. 8 (middle) which shows the EER values (equal error rate) after each iteration. Table 1 summarizes the detection rates of our approach for the GENKI face dataset, as a function of the number of initial sample images used to train the classifier and the average number of bootstrapped samples per iteration. Note that after each iteration the classifier is re-trained with a larger amount of samples, the actual number of training images remains the same. Finally, the top two rows of Fig. 14 show a few sample detections. Note that we are able to address large variability in the size and appearance of the faces.

Comparing plain BRFs with Bootstrapped BRFs: We also compared the use of the bootstrapping strategy versus increasing the actual number of training images. The results are shown in Fig. 8 (right). Observe that when using the plain BRFs with no bootstrapping, the more the training images we use, the better is the detection rate. The best results are obtained using BRFs with 1000 training samples. Yet, we can obtain similar results by just using 100 training samples and three bootstrapping iterations. Table 2 reports the detection rates for this experiment using the precision–recall (PR) and receiver operating characteristic (ROC) curves.

6.2. INRIA horse dataset [40]

Dataset: This dataset has 170 images with one or more horses per image. The dataset also contains 170 background images which are used as negative samples. The main difficulties encountered in this dataset are due to scale changes, and also to slight poses changes and out-of-plane rotations. For training, as suggested in [40], the first 50 positive and 50 negative images are chosen. The remaining images are used to test the classifier.

Results: Bootstrapping produces significantly better recognition rates on this dataset. As shown in Fig. 9 (left, middle), our BRF

Table 1
GENKI face dataset. Equal Error detection Rates (EER) at increasing number of bootstrapping iterations. The EER is computed for the precision–recall (PR) curve. The last column reports the effective number of training images used after each iteration.

Method	**PR EER**	**Num. Images** pos./neg.	**Num. Boots. Images** pos./neg.
BBRFs/iter. 0	74.3% ± 0.4	100/100	100/100
BBRFs/iter. 1	77.8% ± 1.9	100/100	491/869
BBRFs/iter. 2	82.7% ± 0.3	100/100	868/1579
BBRFs/iter. 3	83.2% ± 0.4	100/100	1231/2000
BBRFs/iter. 4	83.0% ± 0.4	100/100	1577/2000
BBRFs/iter. 5	82.8% ± 0.5	100/100	1914/2000

Table 2
GENKI face dataset. Detection rates for the proposed BBRFs and BRFs with varying number of training images. The classifier performance is measured in terms of the Equal Error detection Rate for both the PR and ROC curves. ROC values are measured considering 1 false positive per image.

Method	**PR EER** (%)	**ROC 1.0 FPPI** (%)	**Num. Images** pos./neg.
BRFs/100 samples	74.3	78.2	100/100
BRFs/250 samples	77.7	81.3	250/250
BRFs/500 samples	79.7	83.4	500/500
BRFs/1000 samples	81.7	85.3	1000/1000
BBRFs/iter. 3	83.0	86.3	100/100

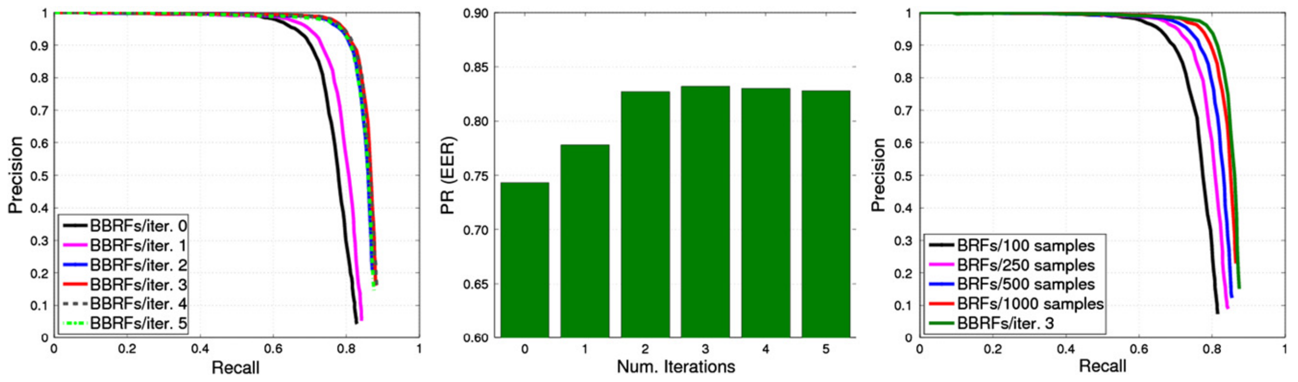

Fig. 8. *GENKI face dataset. Left*: precision–recall plots of BBRFs for several bootstrapping iterations. The detection rate increases with the number of iterations. *Middle*: EER values for each one of bootstrapping iterations. The maximum rate is achieved at three bootstrapping iterations. After that, the performance of BBRFs decreases slightly. *Right*: detection curves of BRFs [26] as a function of size of the training set. Note that bootstrapping 100 training images yields better results than directly training the BRFs with 1000 sample images.

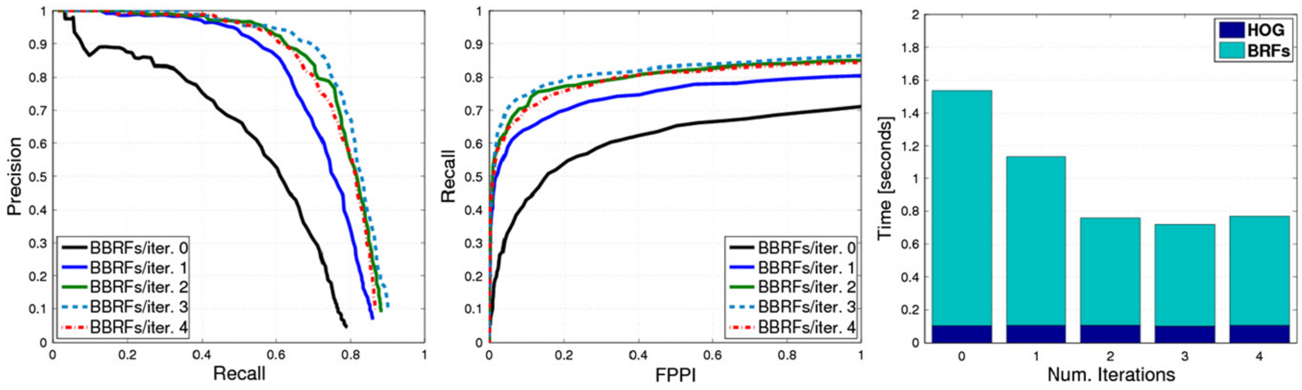

Fig. 9. *INRIA horse dataset*. The detection performance of Boosted Random Ferns on the horses dataset improves substantially as the number of bootstrapping iterations increases. (*left*) Precision–recall curve. (*middle*) Receiver-operator curve. (*right*) Detection times for BBRFs according to the number of bootstrapping iterations. The bootstrapped classifier is not only more discriminative but also more efficient.

Table 3
INRIA horse dataset. Quantitative detection results and comparison with the state-of-the-art. For every bootstrapping iteration the training data is extended with new positive and negative samples. Indeed, they are the same original samples (50 samples) under image distortions: scaling, location shifts and affine transformations.

Method	**Num. Samples** pos./neg.	**Num. Boots. Samples** pos./neg.	**PR EER**	**ROC 1.0 FPPI**
Ferrari et al. [40]	50/50	–	–	73.7%
Ferrari et al. [41]	50/50	–	–	80.8%
Riemenschneider et al. [28]	50/0	–	–	83.7%
Maji et al. [29]	50/50	–	–	86.0%
Yarlagadda et al. [27]	50/0	–	–	87.3%
Toshev et al. [51]	50/50	–	–	92.4%
Monroy et al. [23]	50/50	–	–	94.5%
BHOGs/iter. 0	50/50	–	55.6%	76.5%
BHOGs/iter. 1	50/50	151/198	56.2%	66.6%
BHOGs/iter. 2	50/50	251/344	64.4%	75.6%
BHOGs/iter. 3	50/50	351/488	63.2%	79.6%
BBRFs/iter. 0	50/50	–	$56.7\% \pm 4.6$	$71.2\% \pm 4.2$
BBRFs/iter. 1	50/50	295/576	$68.4\% \pm 3.5$	$80.0\% \pm 4.6$
BBRFs/iter. 2	50/50	577/1044	$75.0\% \pm 3.2$	$85.1\% \pm 2.9$
BBRFs/iter. 3	50/50	861/1350	$77.0\% \pm 3.0$	$86.0\% \pm 2.8$
BBRFs/iter. 4	50/50	1120/1590	$75.6\% \pm 1.3$	$84.4\% \pm 1.6$

classifiers are evaluated for several bootstrapping iterations, with the conclusion that it takes three iterations to optimize the results of bootstrapping. Note how for more iterations, overfitting starts to kick in and the performance of the classifier slightly degrades. Moreover, the method competes favorably with the state-of-the-art as shown in Table 3.

With regard to the state-of-the-art, our method outperforms some related works with a detection rate of $86.0\% \pm 2.8$ at 1 FPPI. This detection result and the number of training samples used at each bootstrap iteration are shown in Table 3. The proposed method is only superseded by Yarlagadda et al. [27], Toshev et al. [51] and Monroy et al. [23]. However, these works require larger computational effort. In Yarlagadda's work, for example, a novel Hough-based voting scheme is proposed for object detection. This method, requires an optimization procedure to group dependent parts and a verification stage to refine the voting hypotheses. In Toshev's work, a boundary structure segmentation model is proposed. Particularly, this method makes use of an initial segmentation based on superpixels and an optimization problem that is solved using semidefinite programming relaxation. In Monroy's work, the method integrates curvature information with HoG-based descriptors to produce better recognition and accuracy results. However, this method also requires additional processing steps, such as the computation of sophisticated edges, the extraction of segments based on connected elements and the computation of the distance accumulation. Besides, this work also performs bootstrapping. More precisely, three iterations are carried out to bootstrap more negative samples.

Finally, some detections results for this dataset are shown in Fig. 14. In spite of challenging poses of the horses, the BBRFs are capable of detecting the object category remarkably well.

Detection times: The proposed BBRFs are very fast to compute, and take about 1 s to detect horses in one image. On the contrary, Yarlagadda's method takes about a couple of minutes per image on this dataset and Riemenschneider et al.'s takes about 5 s per image. This fact reflects the computational benefit of our method. This efficiency is achieved by using simple but discriminative Random Ferns in combination with a boosting phase [26]. Fig. 9 (right) shows the detection times of our proposed method in terms of the bootstrap iterations. Specifically, the times for computing the HOG from images, and testing the BRFs are shown. We also see that the bootstrapped classifier is more efficient because it is more selective and can discard background image regions more quickly.

Comparison against Boosted HoGs: In order to highlight the advantages of the Random Ferns we have also compared our BBRFs approach against a boosted classifier based on local HoGs (BHOGs). More precisely, we have used AdaBoost to iteratively pick and compute the most discriminative HoGs. This results in a set of weighted and local HoGs that capture the most important object boundaries (Fig. 10 (left)). In order to perform a fair comparison, we have used the proposed bootstrapping strategy to progressively retrain the HoG classifier using larger sets of training samples. Fig. 10 (middle) and Table 3 show the detection performance of the BHOGs at different bootstrapping steps. It can be seen that BBRFs consistently outperform BHOGs, specially for larger number of iterations. This is also shown in the EER of the precision–recall curve depicted in Fig. 10 (right), which indicates that BBRFs are more adequate to capture intra-class variations and out-of-plane rotations than the HoGs templates.

6.3. *Caltech face dataset [42]*

Dataset: This dataset consists of 450 frontal people face images. The images are acquired both in indoor and outdoor situations and show extreme illumination changes. In our experiments we use 89 images (corresponding to five persons) for training, and 361 images (corresponding to the other 23 persons) for testing. The dataset also contains background images, and we choose the first 89 to generate our initial set of negative samples.

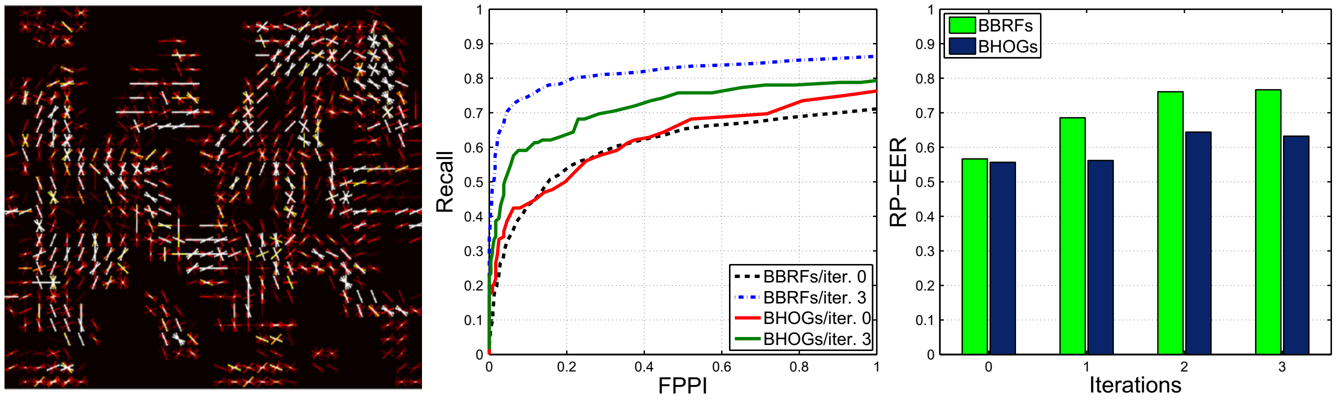

Fig. 10. *INRIA horse dataset*. Detection performance of Boosted HoGs and their comparison with BBRFs. (*left*) Layout and weights of object category gradients. (*middle*) FPPI curve in terms of the number of bootstrapping iterations. (*right*) EER rate on precision–recall curve.

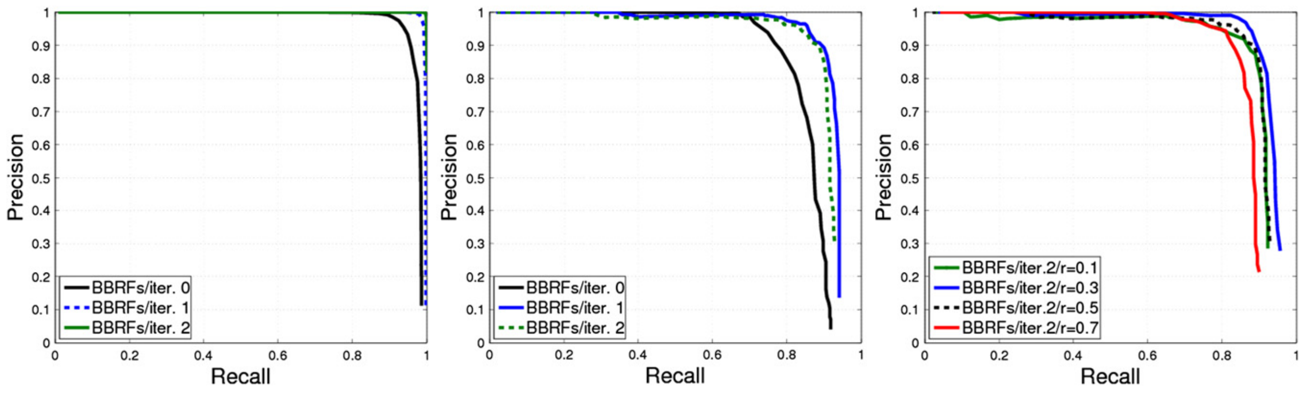

Fig. 11. *Caltech face and IRI freestyle motocross datasets. Left*: classifier evaluation on the Caltech face dataset with a training set with 89 images and a test set with 361 images. The plot shows detection performance for 0, 1 and 2 bootstrapping iterations in the form of ROC curves. Note that in this case, one iterations suffices to obtain nearly perfect classification rates. *Middle*: classifier evaluation for the IRI freestyle motocross dataset. ROC curves are shown for varying bootstrap iteration values. *Right*: effect of overlap ratio for the IRI freestyle motocross dataset. ROC curves for BBRFs classification with three bootstrap iterations and varying ratios of overlap (Eq. (7)). An overlap of $r=0.3$ yields the best recognition rates for this dataset.

Table 4
Caltech face dataset. Equal error detection rates for two competing approaches, as well as BRFs with and without bootstrapping. The EER is computed for both the precision–recall and the receiver operating characteristic curves.

Method	PR EER	ROC EER
Shotton et al. [17]	94.0%	96.5%
Fergus et al. [42]	–	96.4%
BBRFs/iter. 0	94.5% ± 1.2	94.4% ± 1.2
BBRFs/iter. 1	98.4% ± 0.5	98.4% ± 0.5
BBRFs/iter. 2	99.6% ± 0.2	99.7% ± 0.2
BRFs/ext. set 1	97.8% ± 0.6	97.9% ± 0.5
BRFs/ext. set 2	98.4% ± 0.4	98.4% ± 0.4

Table 5
IRI freestyle motocross dataset. EER rates over the PR and ROC curves. BBRFs are compared to BRFs with large initial sample sets. In this case BBRFs do not outperform BRFs.

Method	PR EER	ROC EER
Villamizar et al. [26]	91.0%	–
BBRFs/iter. 0	82.0% ± 2.6	82.2% ± 2.5
BBRFs/iter. 1	88.9% ± 2.0	88.6% ± 2.0
BBRFs/iter. 2	88.1% ± 1.7	87.9% ± 1.8
BBRFs/iter. 2/r=0.1	87.9% ± 3.6	87.5% ± 4.2
BBRFs/iter. 2/r=0.3	89.1% ± 2.7	88.5% ± 3.0
BBRFs/iter. 2/r=0.5	88.1% ± 1.7	87.9% ± 1.8
BBRFs/iter. 2/r=0.7	84.1% ± 3.4	84.8% ± 2.7

Note that a set of 89 training images represents quite a small set size, and hence, is a good scenario to exploit the benefits of our bootstrapping approach.

Results: The detection performance of BBRFs for this dataset with respect to the number of bootstrapping iterations is shown in Fig. 11 (left). In these experiments, the classifier is first learned using the whole training set (89 images), and then bootstrapped with true and false positives accordingly. Recognition is tested on the rest of the set (361 images). It takes only one bootstrap iteration to achieve remarkable detection rates, compared to the classifier trained without bootstrapping.

The fifth and sixth rows in Fig. 14 show some detection results for the face dataset. The proposed method is able to detect faces in images with extreme lighting conditions, partial occlusions, scale variations, and even in drawings. Detection fails for more severe occlusions and backlight.

Comparison: We compare our method against other state-of-the-art methods for this dataset in Table 4. The comparison is done according to the equal error rate (EER), the point at which accept and reject errors are equal, as computed from the precision–recall (PR) and receiving operating characteristic (ROC) curves. The last two rows in the table show results of using the BRFs classifier without bootstrapping, but with artificially enlarged initial training sets the same size as the bootstrapped sets, for which positive training samples were produced artificially with moderate affine transformations, and negative training samples were also added. This is to show that the increase in detection performance in the bootstrapped methods does not come at the expense of larger training sample sizes at each bootstrap iteration.

6.4. *IRI freestyle motocross dataset [26]*

Dataset: This dataset was originally proposed for validating a rotationally invariant detector. It has two sets of images, one including motorbikes with large rotations in the image plane, and a second one containing side views of motorbikes with a similar pose. In this work, we only tested our classifier over the second set, which contains 69 images with 78 motorbike instances. Although rotations are not considered, this set of images still represents a challenging dataset, mostly due to the presence of large scale changes and partial occlusions. As training data we used 50 positive and 50 negative images from another dataset, the Caltech motorbike dataset [42]. This is again a situation with a rather small training set.

Results: Our BBRFs classifiers are again tested for varying numbers of bootstrap iterations. The results are shown in Fig. 11 (middle). As with the faces dataset, one bootstrap iteration suffices to substantially increase classification performance. Furthermore, the results of a second comparison, this time with respect to the size of the overlap ratio r, are given in Fig. 11 (right). The overlap ratio controls the level of distortion for samples to be added in the next bootstrap iteration. That is, low values of this ratio allow increased feature variation in the positive samples set. Large values of r mean little intra-class variation for the positive training sample set, thus limiting the effect of bootstrapping. As shown in the plot, the best detection rates are achieved at $r=0.3$ for this dataset.

Detection rates via EER on the PR and ROC curves are summarized in Table 5, and compared against EER rates for the method proposed in [26]. To allow for a fare comparison with respect to the size of the training set, the latter method contains 800 positive and 800 negative sample images, collected from the Caltech motorbike dataset [42]. Bootstrapping has a difficult time superseding the already good classification levels of BRFs on this very challenging dataset.

Table 6
UIUC car dataset. EER of our approach at varying bootstrapping iterations compared to competing methods. Our approach achieves virtually the same detection results as other methods even when using a significantly smaller number of training samples.

Method	PR EER	Num. Images pos./neg.
Agarwal et al. [30]	39.6%	500/500
Fritz et al. [52]	87.8%	500/500
Mutch et al. [43]	90.6%	500/500
Mikolajczyk et al. [31]	94.7%	50/0
Leibe et al. [6]	95.0%	50/0
Lampert et al. [53]	98.6%	500/500
Gall et al. [7]	98.6%	500/450
BBRFs/iter. 0	$93.9\% \pm 2.3$	50/50
BBRFs/iter. 1	$89.8\% \pm 2.5$	50/50
BBRFs/iter. 2	$97.6\% \pm 0.5$	50/50
BBRFs/iter. 3	$97.9\% \pm 0.5$	50/50

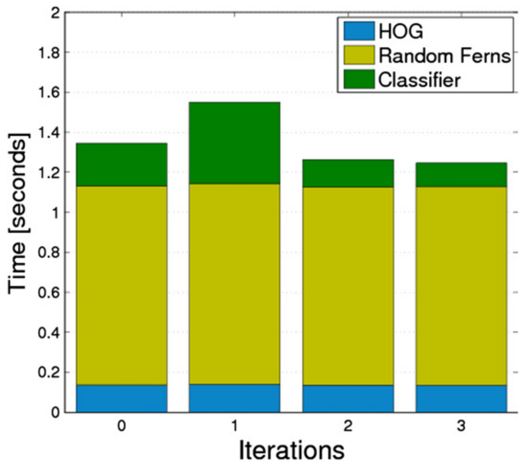

Fig. 12. *UIUC car dataset*. Classifier performance for varying number of bootstrapping iterations. Note that as the number of bootstrapping iterations increases, the detector becomes more discriminant with the consequent increase in classifier efficiency.

6.5. *UIUC car dataset [30]*

Dataset: This dataset contains graylevel side-view images of cars with very large intra-class variation. This is a very large dataset with 550 positive samples and 500 negative samples. Yet, for the experiments reported below, we will use a much smaller subset of training images.

Results: We train the BBRFs using an input training set of only 50 positive and 50 negative samples. Our aim is to compute an effective but also efficient object classifier using a small set of training data. As we did in the previous experiments we show the precision–recall curve for increasing number of bootstrap steps. Surprisingly, the classifier does not improve after the first iteration, but it does afterwards. This is probably because the UIUC car dataset does not include background regions in the positive

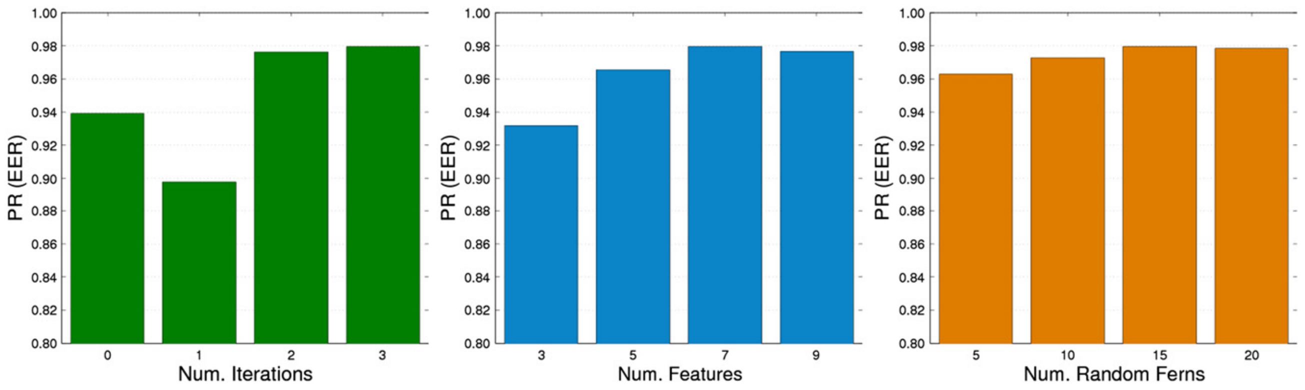

Fig. 13. *UIUC car dataset. Left*: the BBRFs are evaluated as a function of the number of iterations and using just 50 positive and 50 negative images for training. *Middle*, *Right*: detection rates as a function of the number of Random Ferns and the number of binary features per Fern.

samples, and hence, the first bootstrapping step only introduces affine transformations of positive samples in the training data.

Table 6 compares equal error detection rates on the PR curve for the BBRFs and other state-of-the-art approaches. Competitive results are obtained only after the second bootstrap iteration, overcome only by Lampert et al. [53] that use subwindow search and by Gall et al. [7] that rely on class specific Hough forests for object detection. Note however, that we are achieving equivalent

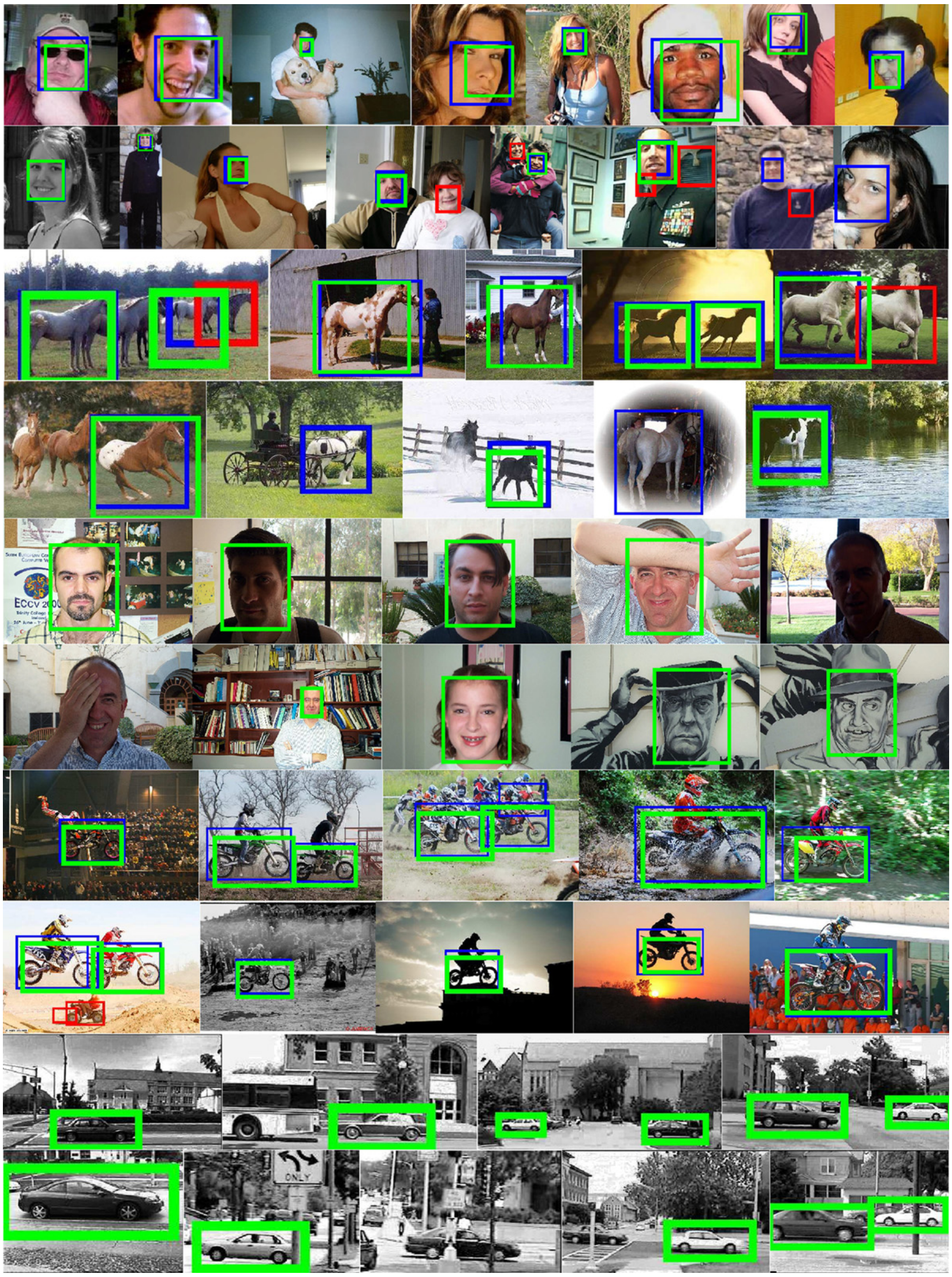

Fig. 14. Detection results of Bootstrapped Boosted Random Ferns on several databases. The method is capable of correctly finding multiple object classes with low failure rates despite very challenging intra-class variations such as pose change, color, texture, and size for the horses data set; illumination and size for the faces data set; size and difficult background for the motorcycles dataset; or shape and size for the cars dataset. Correct matches are labeled in green and missed matches are labeled in red. The blue squares indicate ground truth in some of the training data. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

results using approximately one tenth of the training data used by these methods.

Efficiency: Fig. 12 shows average detection times for the BBRFs for varying numbers of bootstrapping iterations. The plots indicate the average time per image spent by the algorithm during recognition while testing over the reduced dataset of 100 images. Given that bootstrapping increases classifier specifity, a BBRFs classifier with four bootstrapping iterations is more efficient than the same method with fewer numbers of bootrstrapping iterations. The time spent computing Ferns and HoGs on the image remains constant.

Parameters: We also evaluated our approach under different parameters that determine the configuration of the BRFs, such as the number M of Ferns and the number R of binary features per Ferns. The results of this experiment are depicted in the middle and right plots of Fig. 13. The best detection rates are obtained for ($R=7$) and ($M=15$), which agrees with the results already reported in [26].

7. Conclusions

We have presented a method to improve the performance of binary classifiers based on Boosted Random Ferns by bootstrapping the learning step. The presented method shows that bootstrapping allows significant reduction in the size of the training sets without jeopardizing classification performance. This results in a classifier which is both highly discriminative and computationally efficient and is particularly suitable when only small sets of training images are available.

We compared our algorithm against state-of-the-art approaches for several databases including faces, cars, motorbikes and horses, and showed remarkable improvements in efficiency and detection rates with just a few bootstrapping steps.

Acknowledgments

This work was supported by the Spanish Ministry of Science and Innovation under Projects RobTaskCoop (DPI2010-17112), PAU (DPI2008-06022), PAU+ (DPI2011-27510) and MIPRCV (Consolider-Ingenio 2010) (CSD2007-00018), and the EU ARCAS Project FP7-ICT-2011-287617.

References

[1] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[2] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[3] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[4] H. Bay, T. Tuytelaars, L.V. Gool, SURF: speeded up robust features, in: European Conference on Computer Vision, 2006.

[5] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[6] B. Leibe, A. Leonardis, B. Schiele, Robust object detection with interleaved categorization and segmentation, International Journal of Computer Vision 77 (1–3) (2008) 259–289.

[7] J. Gall, V. Lempitsky, Class specific Hough forests for object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[8] F. Moreno-Noguer, Deformation and illumination invariant feature point descriptor, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[9] F. Moreno-Noguer, A. Sanfeliu, D. Samaras, Dependent multiple cue integration for robust tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (4) (2008) 670–685.

[10] M. Villamizar, H. Grabner, J. Andrade-Cetto, L. Gool, A. Sanfeliu, F. Moreno-Noguer, Efficient 3d object detection using multiple pose-specific classifiers, in: British Machine Vision Conference, 2011.

[11] V. Lepetit, P. Fua, Keypoint recognition using randomized tree, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (9) (2006) 1465–1479.

[12] M. Ozuysal, P. Fua, V. Lepetit, Fast keypoint recognition in ten lines of code, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[13] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[14] K.K. Sung, T. Poggio, Example-based learning for view-based human face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1) (1998) 39–51.

[15] C.P. Papageorgiou, M. Oren, T. Poggio, A general framework for object detection, in: International Conference on Computer Vision, 1998.

[16] H. Schneiderman, T. Kanade, A statistical method for 3D object detection applied to faces and cars, in: IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[17] J. Shotton, A. Blake, R. Cipolla, Contour-based learning for object detection, in: International Conference on Computer Vision, 2005.

[18] J. Luo, M. Boutell, R. Gray, C. Brown, Image transform bootstrapping and its applications to semantic scene classification, IEEE Transactions on Systems, Man, and Cybernetics—Part B 35 (3) (2005) 563–570.

[19] A. Fred, A. Jain, Combining multiple clusterings using evidence accumulation, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (6) (2005) 835–850.

[20] S. Munder, D. Gavrila, An experimental study on pedestrian classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1863–1868.

[21] Z. Kalal, J. Matas, K. Mikolajczyk, Weighted sampling for large scale boosting, in: British Machine Vision Conference, 2008.

[22] Z. Kalal, J. Matas, K. Mikolajczyk, P-N learning: bootstrapping binary classifiers by structural constraints, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[23] A. Monroy, A. Eigenstetter, B. Ommer, Beyond straight lines—object detection using curvature, in: International Conference on Image Processing, 2011.

[24] H. Grabner, H. Bischof, On-line boosting and vision, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[25] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, N. Navab, Dominant orientation templates for real-time detection of texture-less objects, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[26] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, A. Sanfeliu, Efficient rotation invariant object detection using boosted random Ferns, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[27] P. Yarlagadda, A. Monroy, B. Ommer, Voting by grouping dependent parts, in: European Conference on Computer Vision, 2010.

[28] H. Riemenschneider, M. Donoser, H. Bischof, Using partial edge contour matches for efficient object category localization, in: European Conference on Computer Vision, 2010.

[29] S. Maji, J. Malik, Object detection using a max-margin Hough transform, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[30] S. Agarwal, D. Roth, Learning a sparse representation for object detection, in: European Conference on Computer Vision, 2002.

[31] K. Mikolajczyk, B. Leibe, B. Schiele, Multiple object class detection with a generative model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[32] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.

[33] R. Schapire, Strength of weak learnability, Machine Learning 5 (2) (1990) 197–227.

[34] A. Demiriz, K. Bennett, J. Shawe-Taylor, Linear programming boosting via column generation, Machine Learning 46 (2002) 225–254.

[35] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Annals of Statistics 28 (2) (2000) 337–407.

[36] A. Levin, P. Viola, Y. Freund, Unsupervised improvement of visual detectors using cotraining, in: International Conference on Computer Vision, 2003.

[37] N. Duffield, C. Lund, M. Thorup, Priority sampling for estimation of arbitrary subset sums, Journal of the ACM 54.

[38] G. Escudero, L. Marquez, G. Rigau, Boosting applied to word sense disambiguation, in: European Conference on Machine Learning, 2000.

[39] F. Fleuret, D. Geman, Stationary features and cat detection, Journal of Machine Learning Research 9 (2008) 2549–2578.

[40] V. Ferrari, F. Jurie, C. Schmid, Accurate object detection with deformable shape models learnt from images, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[41] V. Ferrari, L. Fevrier, F. Jurie, C. Schmid, Groups of adjacent contour segments for object detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (1) (2008) 36–51.

[42] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2003.

[43] J. Mutch, D. Lowe, Multiclass object recognition with sparse, localized features, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[44] Z. Kalal, J. Matas, K. Mikolajczyk, Online learning of robust object detectors during unstable tracking, in: ICCV Online Learning for Computer Vision Workshop, 2009.

[45] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (3) (1999) 297–336.
[46] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, A. Sanfeliu, Shared random Ferns for an efficient detection of multiple categories, in: International Conference on Pattern Recognition, 2010.
[47] F. Porikli, Integral histogram: a fast way to extract histograms in cartesian spaces, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[48] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, A. Zisserman, The PASCAL visual object classes challenge 2007 (VOC2007) results, in: ICCV PASCAL Visual Recognition Callenge Workshop, 2007.
[49] ⟨http://mplab.ucsd.edu⟩, The MPLab GENKI Database, GENKI-SZSL Subset.
[50] V. Ferrari, F. Jurie, C. Schmid, From images to shape models for object detection, International Journal of Computer Vision 87 (3) (2010) 284–303.
[51] A. Toshev, B. Taskar, K. Daniilidis, Object detection via boundary structure segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.
[52] M. Fritz, B. Leibe, B. Caputo, B. Schiele, Integrating representative and discriminant models for object category detection, in: International Conference on Computer Vision, 2005.
[53] C. Lampert, M. Blaschko, T. Hofmann, Beyond sliding windows: object localization by efficient subwindow search, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008.