Boosted Random Ferns for Object Detection

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

Abstract—In this paper we introduce the *Boosted Random Ferns* (BRFs) to rapidly build discriminative classifiers for learning and detecting object categories. At the core of our approach we use standard random ferns, but we introduce four main innovations that let us bring ferns from an instance to a category level, and still retain efficiency. First, we define binary features on the histogram of oriented gradients-domain (as opposed to intensity-), allowing for a better representation of intra-class variability. Second, both the positions where ferns are evaluated within the sliding window, and the location of the binary features for each fern are not chosen completely at random, but instead we use a boosting strategy to pick the most discriminative combination of them. This is further enhanced by our third contribution, that is to adapt the boosting strategy to enable sharing of binary features among different ferns, yielding high recognition rates at a low computational cost. And finally, we show that training can be performed online, for sequentially arriving images. Overall, the resulting classifier can be very efficiently trained, densely evaluated for all image locations in about 0.1 seconds, and provides detection rates similar to competing approaches that require expensive and significantly slower processing times. We demonstrate the effectiveness of our approach by thorough experimentation in publicly available datasets in which we compare against state-of-the-art, and for tasks of both 2D detection and 3D multi-view estimation.

Index Terms—Image processing and computer vision, object detection, random ferns, boosting, online-boosting.

1 Introduction

Detecting a specific object of interest in unconstrained images is a challenging task, as object appearance may suffer large variations due to viewpoint and illumination changes, occlusion or clutter. Yet, it has been shown that if several training images of the object are available, its appearance can be accurately modeled using simple classifiers built from hundreds of binary features. Decision Trees or Random Ferns are used for this purpose, and allow for real time object detection [23], [29], [38], [57].

When instead of dealing with single instances we seek to model object categories, it is necessary to further consider intra-class variation, especially due to differences in object shape and texture. To handle this additional complexity, current approaches resort to more elaborate statistical techniques such as Boosting [36], [44], [49], [64] or Support Vector Machines [7], [10], [20] that learn class appearance from large training sets. Yet, while most of these approaches provide high detection rates on many datasets, they are usually combined with computationally demanding features, like SIFT [32], making them slow when integrated on a standard sliding window approach. To speed up the search, cascade strategies [58], [65] or branch-and-bound schemes [24], [26] have been proposed. Recently, an alternative paradigm based on a random forest codebook and Hough voting, has been shown to yield high detection rates with significant efficiency levels (a few seconds per image) [16]. This is further accelerated in [48], where binary HOG features are combined with a random forest classifier, to achieve pedestrian detection at about five frames per second. In any case, all these approaches make the problem computationally tractable by optimizing the search strategy, and little is done at the feature level.

In this paper we combine the strengths of the Random Ferns method for single instance detection [38] with a boosting strategy to explicitly compute a reduced set of features that is both highly discriminative and fast to evaluate. More specifically, our classifier consists of a combination of weak learners that evaluate fern-like features on HOGs. Each of these ferns is defined by two parameters: its 2D location on the image plane, and the set of bin indices within the HOG. The main contribution of this paper, is in learning these two parameters using either offline and online boosting. The latter is especially interesting when dealing with very large training datasets that are hard to batch process, or in tracking, where data arrives sequentially. As shown in Fig. 1, boosting automatically places our ferns on the most salient regions of the object and discards non-informative areas. In addition, we also propose sharing the set of histogram bin indices among different ferns. Both these contributions, help to significantly alleviate the computational load of the classifier, yielding detection times on the order of 0.1 seconds.

We demonstrate the effectiveness of our approach, named *Boosted Random Ferns* (BRFs), or Online BRFs (OBRFs) –depending on the boosting strategy we use– through several experiments involving 2D and 3D multi-view localization of object categories. In Fig. 1 we plot the response of our classifier on several public datasets. We will show that both in terms of speed and detection rates our results are at the top of the state-of-the-art.

Preliminary versions of this work already appeared in [53] and [51] for single view-point object detection,

• *All authors are with tthe Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, 08028, Spain. Email: {mvillami, cetto, sanfeliu, fmoreno}@iri.upc.edu.*

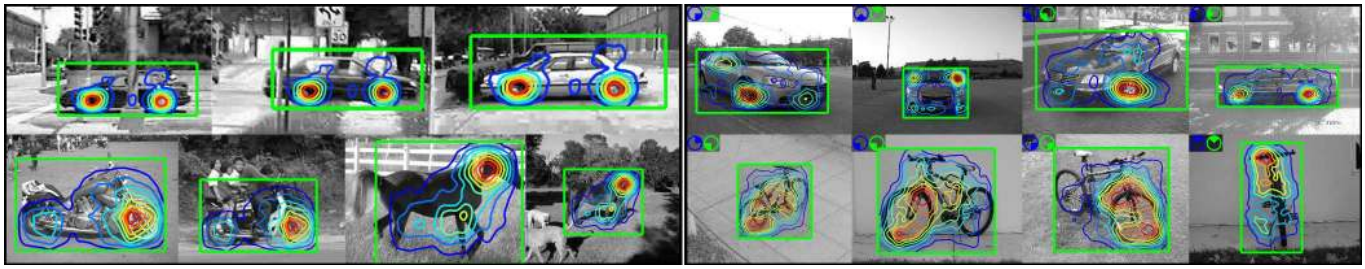

Fig. 1: Results for single- and multi-view object detection. Left: 2D detection results on public datasets of cars, motorbikes and horses. The green rectangles indicate the location and scale at which an object was detected. The level contours show the distribution of the centers for the 300 ferns used to build the classifier. Ferns concentrate on regions that have been found to be more discriminative. Note how these locations are salient regions that one would naturally choose as discriminative, like the wheels of a car or the head of a horse. Right: Multi-view detection results on cars and bikes datasets. The pie-like circles on the top-left of each image indicate the estimated (green) and ground truth (blue) orientation of the object.

and in [52] for multiple view-point detection. In this paper we unify the formulation of all these papers, perform a more in-depth analysis of the algorithm and propose the online version of the method. Thorough discussions about the parameter selection, additional experiments, and further comparisons against competing approaches are also included in this version.

2 Related Work

Techniques for object category detection have matured greatly in recent years, achieving high performance rates on many challenging datasets [2], [3], [8], [10], [16], [18], [20], [34], [62]. The methodology more widely used to tackle this problem is based on the sliding window approach, where a classifier first extracts features from a rectangular window scanned over the image and then computes their probability to belong to a specific object class. SVM [2], [7], [10], [20], [24], [34] and Boosting [25], [36], [44], [49], [58], [61], [64] are typical classifier used for this purpose.

Regarding the features that feed the classifiers, we may find approaches using pixel intensities [45], [58], [55], contours [11], [44], [50], HOG [7], [10], [60], [62], [64] or any combination of them [28], [37]. Yet, the best detection rates are obtained when considering orientation and scale invariant SIFT-like descriptors [4], [32]. This benefit, though, comes at the price of a heavy computational load, especially if dense computation is needed. Even when these descriptors are only extracted at singular image locations, like corners or salient blobs, and are compactly represented using bags-of-words, the intermediate steps required for their computation still hinder real time performance [27]. There have been several attempts to reduce this overhead, such as the use of a cascade of classifiers [9], [58], [64], or a branch-and-bound strategies to discard regions of the search space where the object is not expected to be [24], [26]. In JointBoost [49] , the computational complexity is reduced by sharing features between classifiers. Yet, this can only be exploited for multiclass and multiview problems. For a two class problem, JointBoost boils down to a simpler boosting technique called GentleBoost [13].

Another alternative to reduce the computational burden is to use features that can be efficiently extracted. In this regard, simple binary features computed on the intensity domain have been shown to accurately capture the varying appearance of a target object. Classifiers like Random Trees, Forests or Ferns [21], [23], [29], [38], [45], [56], [57] have then been proposed for matching these features very quickly, yielding similar results as those obtained with SIFT [32]. However, most of these methods only tackle problems with single object instances, and do not generalize to complete categories.

An interesting exception are the Hough Forests [16], that combine a random forest codebook, computed in a intensity-contour domain to identify object patches, with a Hough voting scheme to determine the location of object centers. Yet, although this methodology improves efficiency compared to sliding-window approaches, it still takes a few seconds per image. This scheme has been recently combined with a coarse-to-fine cascading to achieve pedestrian detection at 5 frames per second [48].

In this paper we advocate for a different strategy to pursuit efficiency. Instead of explicitly focusing on designing an efficient architecture, or a feature that can be rapidly extracted, we will stick to a standard sliding window architecture and to the well-known fern features, and will attempt to reduce the cost of our algorithm by using boosting to optimize the location of only a small number of ferns, and also to optimize an even smaller shared set of binary comparison indices for them. This results in a simple yet very powerful classifier, that achieves state-of-the-art results without requiring most of the pre- and post-processing computations or hand-crafted strategies that are usually tailored to a particular dataset. We would like to point that the combination of boosting with trees (note that ferns are indeed one dimensional trees), has already been explored in [5]. This work evaluates a large number of supervised learning methods of the early 90's on simple binary classification problems, and in particular, it shows very promising results when using different variants of boosted trees.

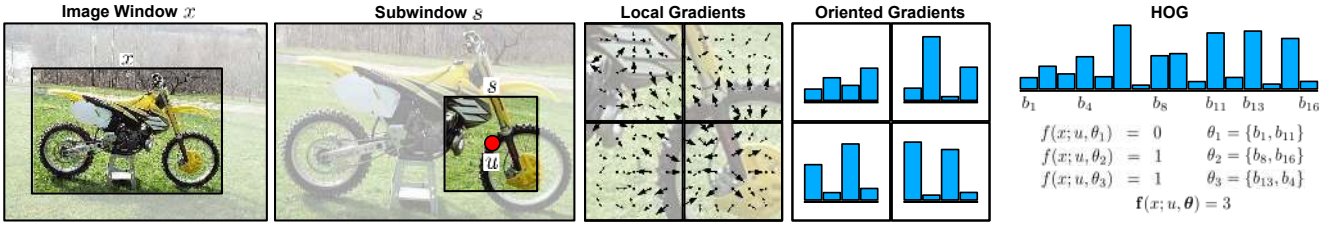

Fig. 2: HOG-based fern. Given an image window x, a Histogram of Oriented Gradients (HOG) is locally computed in a subwindow s, centered at location u. The HOG computation is carried out by calculating gradients within s and casting votes for a particular spatial partition and orientation bin. Votes are weighted according to the gradient magnitude. For this example, a 2×2 spatial grid and 4 orientation bins are considered. The resulting HOG-descriptor is then a concatenation of local and adjacent distributions of oriented gradients. The features f we consider in this paper, compare the value of two of these bins chosen at random during training. The fern $\mathbf{f}$ is made from several such individual features.

3 Boosted Random Ferns

Our approach to object detection uses a fixed-size rectangular sliding window to scan an input image, and computes a classifier response on such window. The classifier returns the probability that such window belongs to a particular object class. Non-maximal neighborhood suppression is further applied to remove multiple overlapping detections, and the process is repeated for different window sizes, to handle scaling.

In this paper we contribute in three main aspects for building the classifier: we define a set of random binary features in the HOG domain, we provide a boosting strategy to select the most discriminant of these features, and devise a framework to share them among the weak learners during boosting. These three contributions significantly alleviate overall computation time, both for training and for testing, while preserving competitive detection rates. We next describe them in detail.

3.1 Random Ferns in the HOG-domain

To compute a set of features on a given image window x, we draw inspiration from the Random Ferns (RFs) classifier [38]. In RFs, a feature f corresponds to a binary comparison of intensity image values at two randomly chosen pixel locations u_1 and u_2 within the window

$$f(x; u_1, u_2) = \mathbb{I}(x(u_1) > x(u_2)) \tag{1}$$

where $\mathbb{I}(a)$ is the indicator function, i.e., $\mathbb{I}(a) = 1$ if a is true, and 0 otherwise.

The computation of these features in the intensity domain has been shown effective to model the varying appearance of patches in a particular object instance [16], [38]. However, our purpose is not modeling just one single object instance, but entire categories. In order to cope with large intra-class variability, we use as features density values of a local histogram of oriented gradients (HOG) computed over a square subwindow s of size $S \times S$ inside x (See Fig. 2).

More specifically, let u be the center of s (measured from the upper-left corner of x), and let $\mathrm{HOG}(x; u, b)$ be the value of the b-th bin of the HOG computed in such subwindow. Each local gradient in s contributes, weighted by its magnitude, to its corresponding orientation bin in the histogram. Then, our binary features are of the form

$$f(x; u, \theta) = \mathbb{I}(\mathrm{HOG}(x; u, b) > \mathrm{HOG}(x; u, b') \tag{2}$$

where $\theta = \{b, b'\}$ are two randomly chosen indices within the interval $[1, B]$, with B being the total number of bins in the histogram.

Just as with intensity-based features in [38], we also represent the appearance of s by aggregating M local features into a binary feature vector called fern; in our case, a HOG-based fern

$$\mathbf{f}(x; u, \boldsymbol{\theta}) = [f(x; u, \theta_1), \ldots, f(x; u, \theta_M)] \,. \tag{3}$$

Note how our fern is fully defined by two parameters: u, the 2D center of s, and M random pairs of histogram bin indices $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_M\}$. These are the parameters for which we will seek to optimize classifier response. Each fern output is an M-dimensional binary vector, which at the implementation level, is represented by an integer value $z \in 0, \ldots, 2^M - 1$.

Fig. 2 (right) shows an example of how a HOG-based fern is computed on a local subwindow s, inside x, and centered at u. In this case, $M = 3$ binary comparisons of HOG bins are considered, with individual outputs 0,1,1. The overall output of this fern is $z = (011)_2 = 3$.

3.2 Building the Classifier

Our HOG-based ferns encode the appearance of small patches inside the window x, which in turn is to be assigned to a particular object or background class. In order to build a classifier for the whole window, we combine the response of T ferns, each centered on a specific position u_t and with their corresponding sets of random pairs of histogram bin indices $\boldsymbol{\theta}_t$, $t \in \{1, \ldots, T\}$.

The major issue that needs to be resolved is which are the fern positions and the associated histogram bin indices that maximize the performance of the classifier. These parameters, once defined, are kept constant when scanning any test image. As for the set of positions, one obvious solution would be to use a very large number T of ferns, uniformly distributed over the whole window x. However, this would be inappropriate, since evaluating many ferns would incur significant computational

Algorithm 1: Boosted Random Ferns (Training)

Input: Training set of object/background class labeled image window samples $(x_1, y_1), \ldots, (x_N, y_N)$.
Output: Object classifier $H(x)$.

Initialize the image window sample weights $w_1(i) = 1/N$, $i = 1, ..., N$
Create the list of 2D pixel positions $u_1, \ldots, u_L$ in x.
Create a random pool of histogram bin index pairs $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R$.
for $t = 1, \ldots, T$ **do**
  for $l = 1, \ldots, L$ **do**
    for $r = 1, \ldots, R$ **do**
      Compute the class-conditioned probabilities $p(z|\mathcal{O})$ and $p(z|\mathcal{B})$ (Eqs. 7 and 8).
      Compute the Bhattacharyya coefficient $Q(u_l, \boldsymbol{\theta}_r)$ (Eq. 9).
  Select the optimal parameters $u_{l^*}, \boldsymbol{\theta}_{r^*}$ for the weak learner $h_t(x)$. (Eq. 10).
  Save the probability histograms $p(z|\mathcal{O})$ and $p(z|\mathcal{B})$.
  Evaluate the classification odds on each image window sample to update the sample weights $w_{t+1}(i)$, $i = 1 \ldots, N$. (Eqs. 6 and 13).
Assemble the final strong classifier $H(x)$, i.e., save parameters u_{l^*} and $\boldsymbol{\theta}_{r^*}$ for all weak learners.

cost, besides that we would be possibly evaluating non-informative or non-discriminative regions of the image.

With regards to the set of parameters $\boldsymbol{\theta}$ to use for each feature, the standard solution adopted in the literature [39], is to assign a different set of random parameters to each fern (pixel coordinates in their case, histogram bin indices in ours), using as many sets $\boldsymbol{\theta}_t$ as ferns. However, as we will discuss below, the cost of testing the classifier can be highly reduced if different ferns share the same parameters. We will see that sampling these features from a pool with $R << T$ parameter sets, we virtually obtain the same classification results as using a different parameter set per fern, but at a significantly smaller cost.

3.2.1 Training the Classifier

In order to train the classifier, and to optimize the fern positions and histogram bin indices, we resort to a Real Adaboost [43] strategy. As it is standard in AdaBoost, we define a strong two-class classifier $H(x) = \{+1, -1\}$ for the object ($\mathcal{O}$) and background ($\mathcal{B}$) classes respectively, using a combination of T weak learners h_t,

$$H(x) = sign\left(\sum_{t=1}^{T} h_t(x) - \beta\right), \tag{4}$$

where β is a threshold that determines the classifier tolerance. Every weak learner returns a confidence score estimating the reliability of classification as the log odds

$$h_t(x) = \frac{1}{2}\log o(x; u_t, \boldsymbol{\theta}_t), \tag{5}$$

where the odds represent a conditional probability ratio

$$o(x; u_t, \boldsymbol{\theta}_t) = \frac{p(\mathbf{f}(x; u_t, \boldsymbol{\theta}_t)|\mathcal{O}) + \epsilon}{p(\mathbf{f}(x; u_t, \boldsymbol{\theta}_t)|\mathcal{B}) + \epsilon} \tag{6}$$

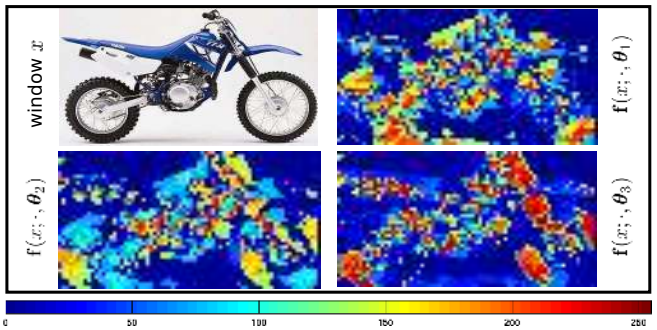

Fig. 3: Example showing the response of ferns densely computed over the image but using only three different vectors of histogram bin indices $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$, and $M = 8$. The fern response is color coded, where each color indicates a particular output value $z \in \{0, \ldots, 255\}$.

and ϵ is a small positive constant used to avoid division by zero. A large score for $h_t(x)$ indicates significant difference between the object and background class probabilities predicted by $\mathbf{f}(x; u_t, \boldsymbol{\theta}_t)$, meaning that the fern position u_t and parameter choice $\boldsymbol{\theta}_t$ produce a discriminative weak learner.

To obtain the adequate values for u_t and $\boldsymbol{\theta}_t$ for each weak learner, we train with a set of N labeled sample pairs $(x_1, y_1), \ldots, (x_N, y_N)$, where $y_i \in \{+1, -1\}$ is the object or background class label associated with each sample image window x_i. Every training sample has also an associated weight $w_t(i)$, initially set to $w_1(i) = 1/N$, and iteratively updated as detailed in Alg. 1.

More specifically, we first define a set $u_1, \ldots, u_L$ of all possible 2D pixel coordinates within a window x, and a pool $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R$ of R different sets of random pairs of histogram bin indices.

Next, to build every weak learner, we iterate for every possible pair $(u_l, \boldsymbol{\theta}_r)$ and assemble the fern $\mathbf{f}(x; u_l, \boldsymbol{\theta}_r)$. For instance, in Fig. 3 we visualize the output of ferns densely computed at every location u_l of an image window x, for three different sets $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$.

Each such fern is then evaluated with respect to the whole training set, and its object and background class probability distribution histograms built with

$$p(z|\mathcal{O}) = \sum_{i:\mathbf{f}(x_i;u_l,\boldsymbol{\theta}_r)=z \ \wedge \ y_i=+1} w_t(i) \tag{7}$$

$$p(z|\mathcal{B}) = \sum_{i:\mathbf{f}(x_i;u_l,\boldsymbol{\theta}_r)=z \ \wedge \ y_i=-1} w_t(i) \tag{8}$$

where $p(z|\mathcal{O})$ stands for $p(\mathbf{f}(x; u_l, \boldsymbol{\theta}_r) = z|\mathcal{O})$, the probability that a fern with parameters u_l and $\boldsymbol{\theta}_r$ has output z given that image window x belongs to the object class; or $p(z|\mathcal{B})$, to the background class. Fig. 4 shows an example of these class-conditional probabilities for two ferns sharing the same feature parameters $\boldsymbol{\theta}_r$, but located at different positions u_l within the window x.

Following [43], we then compute the Bhattacharyya coefficient, an upper bound on the training error we seek to minimize

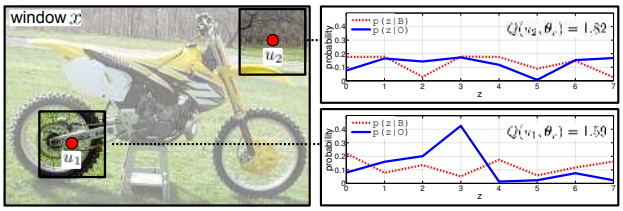

Fig. 4: Class-conditional probabilities of two ferns sharing the same feature parameters $\boldsymbol{\theta}_r$ and tested at two different locations u_1 and u_2. Similarity between these distributions is computed using the Bhattacharyya coefficient Q.

$$Q(u_l, \boldsymbol{\theta}_r) = 2 \sum_{z=0}^{2^M-1} \sqrt{p(z|\mathcal{O})\; p(z|\mathcal{B})}\,. \tag{9}$$

The lower the value of Q, the more discriminant the weak learner $h_t(x)$ is. For instance, in Fig. 4, the fern located at u_1 would be more discriminant than the one at u_2. We therefore keep as parameters for the weak learner h_t, those that minimize Q,

$$(l^*, r^*) \;=\; \arg\min_{l,r} Q(u_l, \boldsymbol{\theta}_r) \tag{10}$$

$$h_t(x) \;=\; \frac{1}{2} \log o(x; u_{l^*}, \boldsymbol{\theta}_{r^*})\;, \tag{11}$$

and store the probability distributions $p(z|\mathcal{O})$ and $p(z|\mathcal{B})$, evaluated in u_{l^*} and $\boldsymbol{\theta}_{r^*}$, used later at test.

Finally, at the end of the iteration, this weak learner is used to update the weights associated to the image window samples

$$w_{t+1}(i) = \frac{w_t(i)\;\exp(-y_i h_t(x_i))}{\sum_{j=1}^{N} w_t(j)\;\exp(-y_j h_t(x_j))} \tag{12}$$

which simplifies to

$$w_{t+1}(i) = \frac{w_t(i) \sqrt{o(x_i; u_{l^*}, \boldsymbol{\theta}_{r^*})^{-y_i}}}{\sum_{j=1}^{N} w_t(j) \sqrt{o(x_j; u_{l^*}, \boldsymbol{\theta}_{r^*})^{-y_j}}} \tag{13}$$

This updating rule increases the weight for the incorrectly classified samples and decreases the weight for the correctly classified samples, with the purpose of focusing the next weak learner $t+1$ on the misclassified ones.

3.2.2 Testing the Classifier

We next explain how BRFs are tested on an input image and give an intuition on the computational effort needed, pinpointing the benefits of sharing fern features.

Given an input image of size $U \times V$, we initially compute its intensity gradients, and for each pixel position, we store them in a HOG with B bins, which encodes the orientation of the pixel neighborhood along such directions. This can be done very quickly using integral images [59], yielding an $U \times V \times B$ HOG image $\mathcal{H}$. A sliding window is then scanned through $\mathcal{H}$, and for every window instance x we need to evaluate T ferns, each centered on a specific position u_{l^*} and with HOG bin indices $\boldsymbol{\theta}_{r^*}$. Here, it is important to note that despite the set of fern positions u_{l^*} can be sparsely distributed within x, we are sliding our window over the entire image. Thus in practice, each fern needs to be densely evaluated for all image locations.

Therefore, instead of evaluating each fern on each individual window x, we alter the order of computations and first convolve the HOG image $\mathcal{H}$ with all needed histogram bin comparisons, R of them. The computation of these convolutions is the most consuming part of our classifier, as it requires $U \times V \times M$ binary comparisons per parameter set $\boldsymbol{\theta}_{r^*}$. The result is a $U \times V \times R$ lookup table with a dense sampling of all possible fern responses.

This explains why sharing $R << T$ histogram bin comparisons among T different ferns, highly improves the efficiency of the classifier [54], [51]. In fact, since the complexity of the rest of computations involved in the test is negligible compared to the cost of these convolutions, we can roughly approximate a $\frac{T}{R}$-fold speed-up achieved by the sharing scheme, compared to the original version of the BRFs presented in [53], where each fern had its own parameter set. With this, a typical classifier with $T = 300$ ferns and $R = 10$ distinct sets of histogram bin indices, yields speed-ups of up to $30\times$.

Testing a classifier entails accessing the aforementioned lookup table at locations u_{l^*} and for the parameter sets $\boldsymbol{\theta}_{r^*}$, and using the stored fern values as indices to the probability distribution histograms $p(z|\mathcal{O})$ and $p(z|\mathcal{B})$ that were saved during the training session. These probabilities are in turn used for the computation of the odds (Eq. 6) and consequently, of the weak learner response (Eq. 5). The response for the entire set of T weak learners is added up (Eq. 4), and decisions about detection are made by thresholding this value. Note that even when only R different fern parameters are used to build the weak learner, we still have T object and background class probability distribution histograms, one per learner, computed with different weight values depending on its order during the training session (Eq. 13). Thus, the whole operation has a computational load linear on T.

In order to handle object scaling, this detection process is repeated at multiple resolutions of $\mathcal{H}$, and multiple scales of the input image. In addition, since our boosting strategy sorts the weak learners according to their discriminant power, we can safely reject an image window x if after evaluating the first K weak learners its accumulated response is negative. In our experiments, $K = 50$ proved a good value for this parameter. This provides additional speed-ups to the detection process.

3.3 A Simple 2D Classification Problem

We designed a simple 2D classification problem to illustrate and compare the performance of BRFs vs other standard classifiers, namely Random Ferns (RFs) [38], Random Forests (RForest) [6] and GentleBoost (GBoost) [13], a variant of JointBoost [49] for two-class problems. This kind of controlled experiments are customary in analyzing classifiers performance [6], [22]. We carried out a controlled comparison in terms of the most relevant parameters concerned with the computation of BRFs, i.e., the number M of features

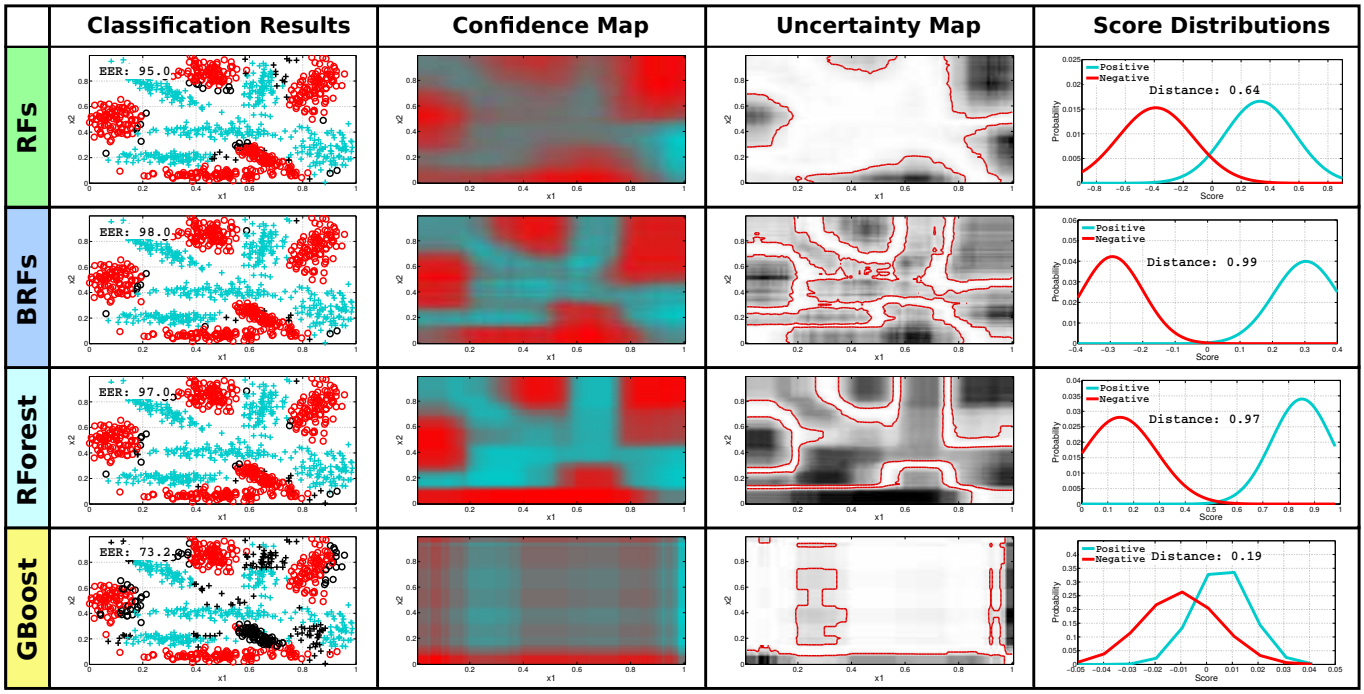

Fig. 5: 2D classification results of BRFs, RFs [38], Random Forest [6] and GentleBoost classifier [13]. First column: classification results on 500 positive and negative testing samples. Correctly classified samples are shown in cyan (positive samples) and red (negative samples). Misclassified samples are shown in black. Second column: confidence maps provided by the classifiers over the 2D feature space. Third column: uncertainty maps where brighter regions correspond to uncertain classification values. Red contours indicate uncertainty of 90%. Fourth column: score distributions for the positive and negative classes.

per fern, and the number T of weak learners tested, where each classifier is associated to a specific fern. The experiment is performed in a synthetic case with two non-separable multi-modal classes (see Fig. 5-Left). For further synthetic experiments with varying degrees of complexity we refer the reader to the Appendix 5.

The classifiers are built using axis-aligned split functions (2D decision stumps) as binary features. Each decision stump f maps a given sample $x \in [0,1]\times[0,1]$ to a Boolean label, $f(x) = \mathbb{I}(x_j > \tau)$, where x_j corresponds to a specific (horizontal or vertical) coordinate of x, and τ is a random threshold in the interval $[0,1]$. No feature sharing is used in this example.

To build the classifiers, we use large pools of 500 ferns (sets of decision stumps), each with individual features selected at random. Every approach then uses a specific strategy to select and ensemble ferns into a set of weak learners and obtain the final classification rule $H(x)$: BRFs use Real AdaBoost; RFs simply choose the features randomly; RForest picks those that maximize information gain at a node-level and GBoost ensembles multiple one-dimensional decision stumps using GentleBoost.

All methods were tested on samples drawn from the same distributions as those used in the training sets, and each experiment was repeated 10 times to account for randomness in feature selection. We base our analysis on two metrics: the Equal Error Rate (EER) on the precision-recall curve[1]; and the Hellinger distance[2] that measures the degree of separability between the positive and negative distributions. We also report the performance in terms of the training and testing (running) times for various numbers of weak learners (WLs) and tree depth values (D), that in the case of RFs and BRFs corresponds to the number of features (M). For the GBoost, we directly use an implementation publicly available[3]. Since no trees are considered, depth $D = 1$.

Fig. 5 shows a qualitative, visual evaluation of the classification performance of all methods, for a case with 100 WLs and $D = 6$ ($D = 1$ for GBoost). Note that BRFs and RForest yield clearly defined decision boundaries (second and third columns of Fig. 5) and a separation (last column) between the positive and negative classes that is much larger than in the case of the RFs and GBoost. This favors the correct classification. An exhaustive quantitative analysis is summarized in the Table of Fig. 6. Again, RForest and specially BRFs, consistently outperform other approaches, providing both much higher Equal Error Rates and larger Hellinger Distances for equivalent classifier configurations.

It is also remarkable that BRFs perform very efficiently, and in particular they can be trained up to $10\times$ faster

1. The equal error rate is the point in the precision-recall curve where precision=recall.

2. The squared Hellinger distance for two distributions P and Q is defined as: $H^2(P,Q) = 1 - \sqrt{k_1/k_2}\exp(-0.25k_3/k_2)$, with $k_1 = 2\sigma_P\sigma_Q$, $k_2 = \sigma_P^2 + \sigma_Q^2$, and $k_3 = (\mu_P - \mu_Q)^2$.

3. http:/people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html

	Equal Error Rate [%]				Hellinger Distance [%]				Training Time [sec.]				Run Times [msec.]				D
RFs	65.3	67.8	70.9	72.1	13.7	12.8	14.0	17.5	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.03	1
	69.8	78.4	80.5	80.4	20.3	26.2	27.7	29.8	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03	2
	83.5	89.4	90.4	91.2	35.6	46.0	49.0	50.0	0.01	0.01	0.02	0.03	0.02	0.03	0.03	0.05	4
	90.8	93.8	94.3	94.9	50.1	60.9	66.6	64.8	0.01	0.02	0.03	0.05	0.02	0.03	0.04	0.06	6
BRFs	73.6	75.9	75.9	75.4	19.1	20.4	18.6	20.9	0.29	0.64	1.21	2.36	0.02	0.02	0.02	0.03	1
	95.4	97.0	97.3	97.6	67.0	83.7	86.9	85.3	0.29	0.66	1.23	2.38	0.02	0.02	0.02	0.03	2
	97.0	97.6	98.0	97.5	86.7	94.6	95.2	94.7	0.35	0.71	1.28	2.47	0.02	0.02	0.03	0.04	4
	97.5	97.3	97.9	97.7	92.4	95.4	97.4	97.3	0.44	0.80	1.38	2.63	0.02	0.03	0.04	0.06	6
RForest	76.6	72.7	72.8	73.0	38.9	34.6	32.3	39.2	0.17	0.38	0.74	1.48	0.27	0.60	1.15	2.29	1
	72.6	75.3	74.3	74.2	28.6	32.1	26.2	29.6	0.41	1.06	2.02	4.02	0.37	0.90	1.70	3.41	2
	89.8	92.9	91.8	90.9	57.3	59.4	60.1	59.0	1.21	2.95	5.87	11.78	0.55	1.34	2.67	5.42	4
	96.8	96.5	96.8	96.9	92.7	93.2	95.2	94.9	2.31	5.91	11.55	23.89	0.71	1.73	3.45	7.25	6
GBoost	73.2	75.4	75.4	75.5	21.0	21.9	19.7	21.5	0.01	0.01	0.02	0.04	0.02	0.02	0.02	0.02	1
# WLs	10	25	50	100	10	25	50	100	10	25	50	100	10	25	50	100	

Fig. 6: Classification performance of RFs, BRFs, RForest and GBoost in the scenario of Fig. 5 for different values of weak learners (WLs) and tree depth (D). First column: mean EERs on the precision-recall curve. Second column: Hellinger distances between classes. Third and fourth columns: computational times for training and testing the classifiers.

D	RForest				BRFs			
1	10	25	50	100	10	25	50	100
2	28	71	138	272	20	50	100	200
4	78	189	386	779	40	100	200	400
6	150	378	753	1533	60	150	300	600
#WLs	10	25	50	100	10	25	50	100

Fig. 7: Number of features used by the RForest and BRFs.

	RFs		BRFs		RForest		GBoost	
N	EER	HD	EER	HD	EER	HD	EER	HD
50	89.0	55.7	94.0	82.0	70.0	15.4	71.6	12.4
100	90.6	56.1	95.2	87.6	77.1	28.1	70.0	15.6
200	94.6	61.6	97.2	95.6	88.4	54.0	72.1	16.8
500	94.0	62.6	97.7	97.6	96.1	90.0	76.5	21.7
1000	94.1	62.6	97.5	97.9	96.8	94.3	75.5	21.2

Fig. 8: Equal Error Rate (EER) and Hellinger distance (HD), as a function of the training set size (N).

than RForest. RFs and GBoost are more efficient to train, but as mentioned above, at a significantly loss in performance. The reason of the efficiency gain of BRFs compared to RForest is that for the latter the size of the trees exponentially grow with the depth D, requiring thus to explore much more features than in the case of the BRFs, in which the number of features remains fixed to the depth of the tree. This is depicted in Fig 7.

Another consequence of the simplicity of the BRFs is that they can be learned with relatively small training sets. As shown in Fig. 8, the performance of the BRFs hardly decays when reducing the size of the training set from $N = 1000$ to 50 examples. All other methods significantly drop their performance. This is quite accentuated in the case of the RForest which, to avoid correlation among trees, performs bagging by randomly splitting the training set into different sample sets per tree. For the BRFs, fern independence, and therefore decorrelation, is automatically enforced by the boosting strategy, which is more effective than randomly splitting the training set. To illustrate this, in Fig. 20 of the appendix we show the process of how boosting picks the four first ferns of the classifier in a 2D classification problem. The example demonstrates that each fern generates a decision boundary decorrelated from all previous ones.

3.4 Online Boosted Random Ferns

We have just seen that BRFs can be learned very efficiently and with relatively small amounts of training data. We next show that they can indeed be trained online in almost real time. We denote this version of the classifier as Online Boosting Random Ferns (OBRFs). Essentially, OBRFs combine the BRFs with an online Boosting algorithm and with an incremental bootstrapping strategy to systematically enlarge the training set. We next describe each of these two main ingredients.

3.4.1 *Online Boosting*

We compute OBRFs using an adaptation of the online Boosting algorithm proposed in [19], which allows building a strong classifier from sequentially incoming training data. For this purpose, [19] introduces the concept of *selector* and builds a strong classifier from a linear combination of T selectors:

$$H(x) = sign\left(\sum_{t=1}^{T} h_t^{sel}(x) - \beta\right) . \tag{14}$$

The basic idea is that we are given a pool of K weak learners $h_k(x)$, and upon the arrival of a new training sample (x, y), each selector $h_t^{sel}(x)$ is updated to the weak learner that minimizes the misclassification error

$$e_{t,k} = \frac{\lambda_{t,k}^{wrong}}{\lambda_{t,k}^{wrong} + \lambda_{t,k}^{corr}}, \tag{15}$$

where $e_{t,k}$ is the error of the k-th weak learner in the t-th selector, estimated from the weights of correctly $\lambda_{t,k}^{corr}$ and wrongly $\lambda_{t,k}^{wrong}$ classified samples seen by far. These weights are estimated and updated incrementally using the importance sample weight λ. Once the selector $h_t^{sel}(\mathbf{x})$ has chosen a weak learner, the weight λ is updated and passed to the next selector $t + 1$.

Regarding specific details of our implementation, it is worth to point that the pool of weak learners is made of ferns $\mathbf{f}(x; u_k, \boldsymbol{\theta}_k)$, where $u_k \in \{u_1, \ldots, u_L\}$ and

Algorithm 2: Online Boosted Random Ferns

Input: Previous online classifier $H(x)$
Input: Input sample (x, y), with $y = \{+1, -1\}$
Input: Classification weights $\lambda_{t,k}^{corr}$, $\lambda_{t,k}^{wrong}$
Output: Updated online classifier $H(x)$

1 Initialize the importance weight $\lambda = 1$
2 **for** $r = 1, \ldots, R$ **do**
3 **for** $l = 1, \ldots, L$ **do**
4 Test the fern $\mathbf{f}(x; u_l, \boldsymbol{\theta}_r)$ for every location u_l of the input sample x to compute the fern outputs z. (Eq. 3)
5 **for** $k = 1, \ldots, K$ **do**
6 Update the estimate $\Theta_{k,z}$ of the weak learner h_k using the computed fern output z, (Eq. 16)
7 **if** $y = +1$ **then**
8 $\eta_{k,z}^{+1} = \eta_{k,z}^{+1} + 1$
9 **else**
10 $\eta_{k,z}^{-1} = \eta_{k,z}^{-1} + 1$
11 **for** $t = 1, \ldots, T$ **do**
12 **for** $k = 1, \ldots, K$ **do**
13 **if** $sign(h_k(x) - \beta) = y$ **then**
14 $\lambda_{t,k}^{corr} = \lambda_{t,k}^{corr} + \lambda$
15 **else**
16 $\lambda_{t,k}^{wrong} = \lambda_{t,k}^{wrong} + \lambda$
17 Select the weak learner $h_t^{sel}(x)$ such that $h_k(x)$ minimizes the misclassification error $e_{t,k}$ (Eq. 15).
18 Update the importance weight λ,
19 **if** $sign(h_t^{sel}(x) - \beta) = y$ **then**
20 $\lambda = \lambda \cdot \frac{1}{2 \cdot (1 - e_t)}$
21 **else**
22 $\lambda = \lambda \cdot \frac{1}{2 \cdot e_t}$
23 Assemble the final strong classifier $H(x) = sign\left(\sum_{t=1}^{T} h_t^{sel}(x) - \beta\right)$. (Eq. 14)

$\boldsymbol{\theta}_k \in \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R\}$, thus yielding a total of $K = L \cdot R$ weak learners. Each of these classifiers represents the probability of the sample x to belong to the object class. In order to speed up the learning process, we set this probability to a Bernoulli distribution,

$$h_k(x) = p(y = 1 | \mathbf{f}(x; u_k, \boldsymbol{\theta}_k)) \sim \Theta_{k,z}^{\mathbb{I}(y=1)} (1 - \Theta_{k,z})^{\mathbb{I}(y=-1)}$$

where $\Theta_{k,z}$ is the distribution parameter indicating the probability that a sample in the fern $\mathbf{f}(x; u_k, \boldsymbol{\theta}_k)$ with output z belongs to the positive class. These parameters can be computed online and very quickly based on a Maximum Likelihood Estimate over the labeled set of samples we have previously observed,

$$\Theta_{k,z} = \frac{\eta_{k,z}^{+1}}{\eta_{k,z}^{+1} + \eta_{k,z}^{-1}} \tag{16}$$

where $\eta_{k,z}^{+1}$ and $\eta_{k,z}^{-1}$ are the accumulated number of positive and negative samples with output z in $\mathbf{f}(x; u_k, \boldsymbol{\theta}_k)$. These numbers are initialized to one, and recomputed for every new input sample (x, y).

Algorithm 2 summarizes the OBRFs computation. Note that all ferns are precomputed (lines 1 and 4) outside of the main loop where the weak learners are updated (lines 5 to 10). This yields a reduction of the computational cost for updating the classifier, allowing frame rates of more than 1 fps in real implementations.

Approaches	Cars PR-EER	Horses PR-EER	Motorbikes PR-EER
BRFs/INT	94.9	54.6	53.4
BRFs/HOG	98.9	74.1	84.4

TABLE 1: Equal error rates achieved by BRFs on various datasets using signed binary comparisons of pixel intensities (INT) or signed binary comparisons of cells of the local histogram of oriented gradients (HOG).

3.4.2 Incremental Bootstrapping

In order to improve the stability and accuracy of the OBRFs we use bootstrapping to feed the classifier with additional input data. For this purpose, the classifier $H(x)$ is evaluated in the input image to extract both positive and negative additional samples that are used to update the weak learners $\{h_k\}_{k=1}^{K}$ and their corresponding classification weights ($\lambda_{t,k}^{corr}$, $\lambda_{t,k}^{wrong}$). The criterion used to decide if a sample is positive or negative is based on the overlapping measure defined by $r = \frac{|B(x_i) \cap B_{\mathrm{A}}|}{|B(x_i) \cup B_{\mathrm{A}}|}$ where $B(x_i)$ and B_{A} denote the bounding boxes for the current sample x_i and the image annotation, respectively. If $r > 0.9$, we assume a positive sample. And if $r < 0.1$ the sample is labeled as negative. This simple strategy is shown to improve the robustness of the classifier to target deformations and background clutter.

4 Applications

We now show extensive experiments in several computer vision problems that validate the performance of BRFs and OBRFs compared to competing approaches in three problems: object class detection, rotation-invariant object detection and multi-view object detection.

4.1 Object Class Detection

For object class detection we train and test BRFs in three public datasets, the UIUC cars dataset [1], the INRIA horses dataset [11], and the TUD motorbikes dataset [14]. These datasets are well-known and allow performance comparison with a wide range of detection methods.

Unless otherwise stated, in these experiments, BRFs are computed with values ranging from $M = \{2, \ldots, 8\}$ features per fern, and $T = \{25, \ldots, 300\}$ weak learners, sharing $R = 10$ random fern parameter sets $\boldsymbol{\theta}_r$. To allow fast feature computation, HOGs are built using 4 unsigned gradient orientations over 6×6 pixel windows, and HOG pyramids are built using integral images.

The experiments were designed to evaluate classifiers based on the type of features, the use of boosting, the amount of fern parameter sharing, and the possibility of performing online-learning.

4.1.1 Feature Space

To evaluate the effect of the type of features we compute BRFs using either binary pixel intensity comparisons or binary comparisons of cell values on the HOG space.

	Cars				Horses				Motorbikes				# WLs
RFs	15.9	23.8	40.5	40.0	14.6	25.7	27.6	20.6	18.3	34.1	35.5	25.3	25
	47.8	49.4	73.1	73.7	27.1	43.1	45.6	39.9	28.0	42.6	51.0	44.9	50
	49.4	65.7	80.3	88.9	42.2	58.1	63.6	54.9	40.5	48.5	65.2	62.9	100
	65.3	70.3	88.7	90.2	55.5	68.9	72.2	72.2	61.3	61.1	70.6	66.0	300
BRFs	75.9	84.5	82.0	85.8	37.4	45.8	41.6	39.4	56.3	60.6	59.7	66.9	25
	87.1	90.7	92.4	94.1	48.1	49.6	53.5	51.5	68.6	73.3	75.8	73.3	50
	94.1	96.6	96.4	98.2	55.7	59.9	64.0	63.7	75.8	79.8	83.4	81.6	100
	96.0	98.4	98.6	98.9	63.6	68.5	70.1	74.1	78.2	83.8	85.6	84.4	300
# Features	2	4	6	8	2	4	6	8	2	4	6	8	

TABLE 2: Comparison of not boosting (RFs) vs. boosting (BRFs) on three distinct object datasets. The values indicate mean equal error rates on the precision-recall curve for varying numbers of fern features, and the number of weak learners used (WLs).

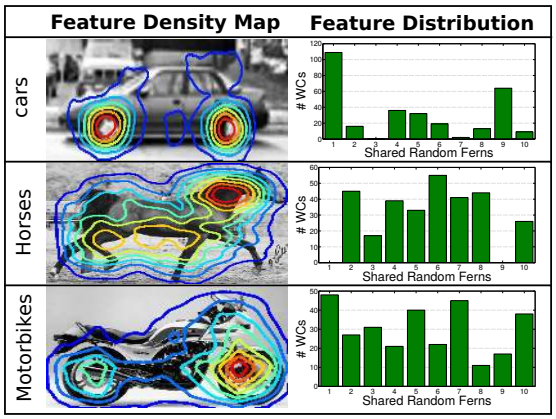

Fig. 9: In the left column, spatial layout of ferns for different object categories. In the right column, distribution of weak learners for each class.

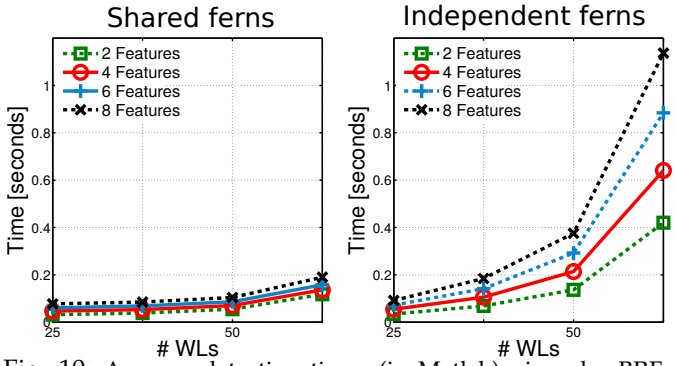

Fig. 10: Average detection times (in Matlab) given by BRFs on the UIUC cars dataset with and without feature sharing for varying feature sizes and total number of weak learners.

The results, shown in Table 1, indicate that HOG-based features yields significant increase in recognition rates. This is particularly relevant for the horses and motorbike datasets, delivering EER gains of up to 30%.

4.1.2 Boosting

We compare the performance of estimating fern parameters with and without boosting, for different numbers of weak learners T, and features per fern M. Table 2 shows mean equal error rates over five training and testing rounds for the three datasets. It can be seen that BRFs yield better detection results than RFs, especially for small fern sizes and few weak learners. This shows once more how RFs require a more stringent method to achieve competitive detection rates, and that performance strongly depends on the amount of features per fern. Here, it is important to clarify that this RFs implementation differs from the original one [38] in that features are computed as comparisons of cells of a local HOG instead of as comparisons of intensity values.

A useful property of boosting is its ability to perform feature selection. The discriminative selection of random ferns during the training phase focuses the classifier in those object parts which are more relevant for classification. In Fig. 9, we observe the results of feature selection for the three different object classifiers. The left frames depict the spatial layout of the ferns used to ensemble the classifiers. The colored contour levels indicate the positional density and weight of individual ferns that give rise to the strong classifier. Red contours indicate higher density of ferns. We can see how ferns concentrate in those regions semantically relevant, such as the wheels of cars and bikes, or the neck and head on horses. The right side of the figure contains plots of shared fern distributions for each strong classifier, that is, the height of the i-th column indicates the number of weak learners that use the parameters $\boldsymbol{\theta}_i$.

4.1.3 Feature sharing

The most important benefit of sharing the sets of histogram bin indices $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R$ among different ferns is the reduction in computational cost when testing the classifier. The pool of ferns is reduced to a small set that can be computed prior to classifier evaluation. Thus, the cost of feature computation will mostly depend on the size of this pool, and not in the total number of weak learners needed. Fig. 10 shows average detection times for the shared and non-shared [53] versions of BRFs, varying the total number of features per fern, and the total number of weak learners. The plots correspond to experiments on the UIUC car dataset, and when the sets of histogram bin indices are shared, we consider a pool of size $R = 10$.

The computational efficiency gained when sharing ferns is at the slight expense of discriminant power of the classifier. This is specially relevant for small fern sizes (N small) and when only a few weak learners are used to build the global classifier. As the number of weak learners or features is increased, fern sharing tends to produce similar results than its non-shared counterpart.

Settings	Cars		Horses		Motorbikes	
	Shar.	Indep.	Shar.	Indep.	Shar.	Indep.
25 WLs 2 Features	75.9	84.9	37.4	42.0	56.3	62.7
50 WLs 4 Features	90.7	95.7	49.6	57.4	73.3	75.8
100 WLs 6 Features	96.4	97.7	64.0	68.0	83.4	82.1
300 WLs 8 Features	98.9	98.9	74.1	73.9	84.4	84.8

TABLE 3: Comparison of shared vs. independent ferns. The table shows mean EER rates of BRFs for the three datasets.

R	Cars		Horses		Motorbikes	
	EER	Time (ms)	EER	Time (ms)	EER	Time (ms)
2	97.8	23	70.6	15	78.0	45
5	99.0	44	73.9	27	83.8	86
10	98.9	76	74.0	47	84.4	150
15	99.1	111	73.4	68	85.3	221
30	98.9	213	74.6	126	84.7	428

TABLE 4: Execution times (in Matlab) and PR-EER rates for various pool sizes of BRFs (R).

Table 3 shows mean EERs for the two alternatives for different fern sizes and total number of weak learners.

Table 4 jointly shows execution time and equal error rates for varying pool sizes. Note how classifier rates are improved as the shared fern pool size gets larger. However, larger fern pools also mean higher computational cost. A good compromise between efficiency and performance will depend on the computational resources available, and for our case, it comes at pool sizes of 10 shared ferns. After that point, recognition rate improvements are negligible.

4.1.4 Comparison to the State of the Art

Table 5 reports ROC (at 1.0 false positives per image) and EER scores of our method and other competitive works when available. BRFs attains very competitive detection rates, yet is straightforward, efficient, and easy to implement. Other works reporting outstanding recognition results, mainly for the horse and motorbike categories, come at the expense of much demanding computational load with detectors that use multiple cues and descriptors [28], [37], [63]. In [28], for instance, descriptors include a top-down image segmentation expensive to compute. [63] proposed Hough-based voting and an optimization procedure to group dependent parts and a verification stage to refine the voting hypotheses, taking a couple of minutes to detect objects. Similarly, [37] integrates curvature information with HOG-based features, yielding high recognition rates but also significant computational burden through the extraction of edges and connected segments tailored to a specific class.

In contrast, BRFs are not class specific, are very simple to compute, and run in less than one second per image in an unoptimized Matlab implementation on a 3.1GHz CPU and up to 10 fps in C++ code. The high detection rates of BRFs are possible not only because we are using more discriminant and boosted cues, but, as in [51], by also adding a bootstrapping mechanism to account for limited training sample sizes. The experiments for the horse dataset are performed with three bootstrapping iterations using 500 weak learners.

Method	Cars PR-EER	Horses ROC	Motorbikes PR-EER
Agarwal et al. [1]	39.6	-	-
Ferrari et al. [12]	-	73.7	-
Ferrari et al. [11]	-	80.8	-
Gall et al. [15]	98.6	-	-
Leibe et al. [28]	-	-	92.8
Leibe et al. [27]	95.0	-	87.0
Maji et al. [33]	-	86.0	-
Mikolajczyk et al. [35]	94.7	-	89.0
Monroy et al. [37]	-	94.5	-
Riemenschneider et al. [41]	-	83.7	-
Toshev et al. [50]	-	92.4	-
Yarlagadda et al. [63]	-	87.3	-
BRFs	98.9 ±0.2	88.6±1.6	87.7 ±1.4
OBRFs	98.7 ±0.4	88.1±1.5	88.4 ± 1.9

TABLE 5: BRFs recognition rates compared to the state of the art for object detection in the chosen object datasets.

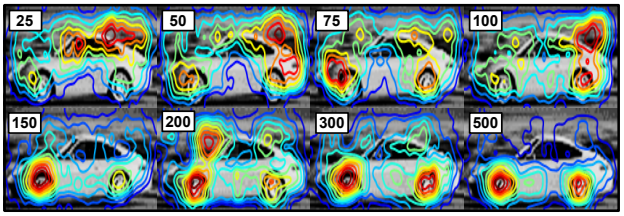

Fig. 11: Spatial layout of ferns at various instances of the training phase in the OBRFs for the UIUC car dataset. Note that after using 300 of the training examples the feature map did already converge.

4.1.5 Online Boosted Random Ferns

We have also evaluated the online version of our Boosted Random Ferns on the same experiments just presented. In this case, training images are sequentially fed to train a classifier with $T = 300$ weak learners, $R = 10$ fern sets $\boldsymbol{\theta}_r$, and $M = 6$ binary features. Once this is done, the classifier is evaluated over the test images. The results are given in the last row of Table 5, yielding virtually the same performance as the offline version. Note, however, that when training the online version the classifier is available at any time of the training, and there is no need to wait until the end. Indeed, as shown in Fig. 11 for the UIUC car dataset, after using about 50% of the training set, the feature map of the detector did already converge. This yields an additional reduction of the computational load required for training the classifier.

4.2 Rotation-Invariant Object Detection

BRFs are versatile enough to be used also in cascade, first to hypothesize object pose, and then to refine detection at that pose. This two-step strategy allows to efficiently detect objects subject to in-plane rotation, without having to densely test the classifier at multiple orientations. The technique is demonstrated for the IRI Freestyle Motocross Dataset [53]. The image set used from the dataset in these experiments includes 100 images of 128 motorbikes under arbitrary in-plane rotations.

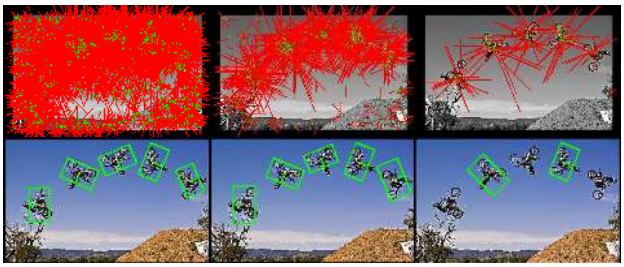

Fig. 12: Cascaded BRFs. Top: object pose hypotheses. Bottom: object classifications results. Each column corresponds to a different value of the tolerance parameter $\beta = (0, 2, 4)$ in Eq. 4.

4.2.1 Orientation Estimation

The pose estimator is trained with image window samples at labeled orientations $\psi(i) \in \{\psi_1, \ldots, \psi_W\}$, generated synthetically from a set of artificially rotated object image windows. Class probability distribution histograms are built for each orientation, i.e.,

$$p(z, \psi|\mathcal{O}) = \sum_{i:\mathbf{f}(x_i;u_l,\boldsymbol{\theta}_r)=z \,\wedge\, y_i=+1 \,\wedge\, \psi(i)=\psi} w_t(i) \;,$$

and optimal weak learner parameters (position and HOG bin indices) are chosen so as to minimize the joint Bhattacharyya bound for all possible orientations

$$Q(u_l, \boldsymbol{\theta}_r) = 2 \sum_{\psi=\psi_1}^{\psi_W} \sum_{z=0}^{2^M-1} \sqrt{p(z, \psi|\mathcal{O})\; p(z|\mathcal{B})} \;.$$

The resulting pose estimator is made up of $T = 300$ weak learners, $R = 7$ HOG bin tests $\boldsymbol{\theta}_r$, and $W = 16$ distinct orientations. The threshold β in Eq. 4 is set permissively enough to allow a significant number of false positives. The top row in Fig. 12 shows object pose estimates for varying β. The red line segments indicate hypothesized object poses, and their length indicates scale.

4.2.2 Detection refinement

In the second level of the cascade, no sliding window is needed. Instead, the above-mentioned detections are fed as testing image windows to another BRFs classifier. This second classifier is built without taking orientation into account, and trained only with image samples at canonical orientations, i.e., images with non-rotated object instances. To achieve rotation invariance, in the test phase, ferns must be evaluated at the adequately rotated HOG bin indices. A simple lookup table is formed to bring the fern parameter set $\boldsymbol{\theta}$ to a new $\boldsymbol{\theta}'$ for each of the W possible values of ψ. The object classifier then proceeds as usual, with 100 weak learners. No feature sharing is used in this case.

4.2.3 Comparison to the State of the Art

To show the benefit of using cascaded BRFs for rotation invariant recognition, we designed an experiment in which the approach is compared against two other methods. The first consists in repeatedly testing with a sliding window, and at multiple orientations, the classifier trained at the canonical orientation. The second method is reported in [31], and proposes the use of equivariant filters and a kernel-weighted model.

Method	Motorbikes		Time
	PR-EER	AP	(secs.)
Pose estimation + Detection BRFs	93.75	90.6	6.1
Individually rotated BRFs	85.94	79.5	19.6
Liu et al. [31]	92.00	-	-

TABLE 6: Performance comparison of cascaded BRFs vs. other methods for rotation invariant object detection.

Fig. 13: Cascaded BRFs for rotation invariant object detection. Green rectangles are true positives, while red ones are false positives. Blue rectangles are ground truth.

Performance rates and computation times for the three methods are shown in Table 6. The performance rates include EER and average precision[4] (AP) were available.

The proposed approach achieves better recognition rates at a significantly smaller computational load for this dataset than using a single BRFs tested at multiple orientations. This is because hypotheses verification at every possible orientation increases not only the computational cost, but also the false positives rate. Our method also compares favorably to using equivariant filters, as reported in [31], despite this work is specifically designed to handle in-plane rotations, while BRFs can be used as general purpose classifiers. Fig. 13 shows detection results for this dataset. In green positive detections, in blue ground truth poses, and in red false positives.

4.3 Multi-View Object Detection

When dealing with multiple view object detection, we need to handle the problem of varying aspect ratios across different viewpoints. Standard approaches like JointBoosting [49] and ClusterBoost [61], do not address this issue and consider constant object sizes. Deformable Part Models [10] handle varying object sizes by incorporating multiple HOG filters, one of which serves as an anchor or 'root', plus local 'part' filters that can shift around it. However, learning this classifier is computationally intensive and slow.

In this paper, we propose an alternative two-step approach in which we first use a Hough voting scheme to generate hypotheses about the object center and corresponding pose, and then we test pose specific BRFs in each of these hypotheses. We next describe these steps.

4. Average precision amounts to the average value of precision (p) for the whole recall (r) interval, and is computed as the area under the precision-recall curve $AP = \sum_{r=0}^{1} p(r)\Delta r$.

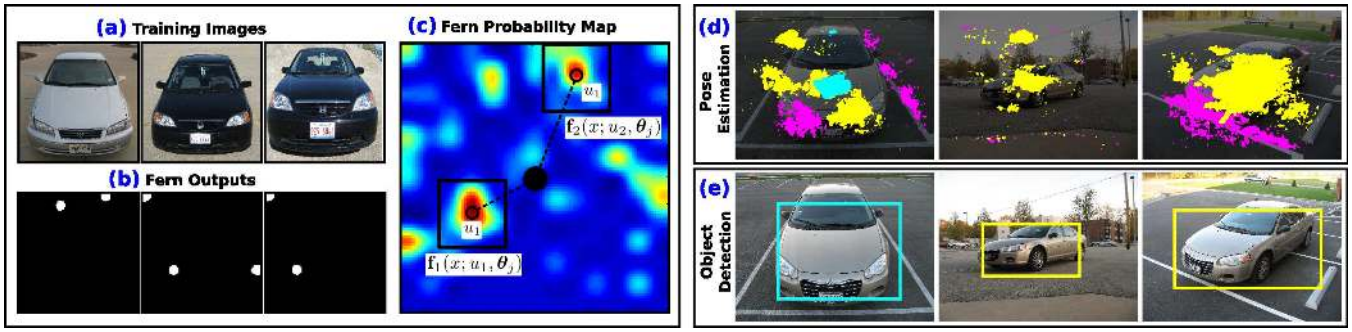

Fig. 14: Two-step approach to multi-view object detection. The first step produces potential hypotheses about the object location and its 3D pose (color encoded). The second step validates each of these hypotheses using pose-specific classifiers (BRFs).

4.3.1 Pose Hypotheses Estimation

In the same spirit as other Hough voting object classifiers [3], [15], we build a pose hypotheses estimator that, at runtime, yields a number of candidate object locations with their corresponding pose. The construction of this estimator is done during training as follows. Let $\mathcal{X}(i)$ be the subset of training samples for one specific pose i (see Fig. 14-a), and assume we are given a fern $\mathbf{f}_j(x; u, \boldsymbol{\theta}_j)$, and an output z_j for this fern. Our goal, is to compute an object pose prior for this specific fern and output. For this purpose, $\mathbf{f}_j$ is evaluated over all training samples $\mathcal{X}(i)$. By retaining only those pixel positions for which the fern output is z_j we build binary response maps as those shown in Fig. 14-b. These binary responses are then added to build a probability map (Fig. 14-c) that, for every pixel position in x, gives the likelihood that if the object is under the i-th pose, the fern $\mathbf{f}_j$ will output z_j. We finally extract the local maxima of this probability map (we plot two maxima in the figure, but we typically choose up to five), and keep its relative displacement to the object center.

This process is iterated over all fern outputs, all ferns, and all poses. We build a lookup table indexed by the fern id, fern output, pose id, and that stores the location of the object center.

At runtime, given a new sample window x, we densely test all ferns at every pixel position. Let's say we test fern $\mathbf{f}_j$ at position u_l and we get that $\mathbf{f}_j(x; u_l, \boldsymbol{\theta}_j) = z_j$. Then, for each pose id, we perform a query at the lookup table, to get votes for the center of the object. Once all ferns are evaluated in all pixel positions, we build clusters of pose indexed object positions, such as those shown in Fig. 14-d.

4.3.2 Object Detection

For the second step, individual BRFs are trained for all different viewpoints, but sharing the same ferns. These strong classifiers are computed using $T = 300$ weak learners each, $R = 15$ ferns parameters sets $\boldsymbol{\theta}_r$, and $M = 7$ HOG bin indices. The best valued pose hypotheses from the first step are then tested on their corresponding viewpoint classifier. Fig. 14-e shows the maximum response of the classifiers, where the bounding box color encodes the pose id.

We use the same configuration of parameters for the Online BRFs, with the only difference that training samples are sequentially used to update the classifier.

4.3.3 Comparison to the State of the Art

We evaluate the approach using the 3DObject dataset of [42], that contains object categories seen from several viewpoints and multiple scales. We report results on the car and bicycle categories. Each class contains 10 object instances observed from 8 different viewpoints, each with significant variation of camera height and distance to the object. The pose estimator and the BRFs classifiers are all trained using five class instances only. The remainder instances are used for testing.

As shown in Table 7, both BRFs and OBRFs compare competitively with respect to the state of the art. Detection rates are computed as the mean values over five training-testing trials to account for randomness. Furthermore, the AP values are computed using the 2010 Pascal protocol[5]. Fig. 15 (left) plots the confusion matrix obtained with BRFs for the car view pose estimation case for which a detection rate of 91.5% was attained. Observe that the low ratio of confusions is mainly due to object symmetry (e.g., viewing the object from opposite sides). Again, the results obtained with OBRFs are equivalent.

The performance of the BRFs is also illustrated in Fig. 1-right, where the localization (green boxes) and orientation estimates (upper-left green circles) are shown. The figure also shows the spatial distribution of features (visualized using colored contours) for the different pose-specific classifiers, as well as ground truth orientations (upper-left blue circles). Note that although all of the orientation-specific classifiers share the same random ferns, their presence in the final classifier has different weights and locations, making it able to distinguish different parts of the object.

The computational benefits of using the initial step when testing BRFs are shown in Fig. 15 (middle, right) for the car class. A tolerance parameter (β_e) for the Hough-based estimator is used to indicate the amount of pose estimation hypotheses, obtained in the first step, that are passed to the classifiers in the second step

5. This protocol, emanating from the Pascal Visual Object Class Challenges is used to smooth out PR values to produce better estimates of the area under the precision-recall curve.

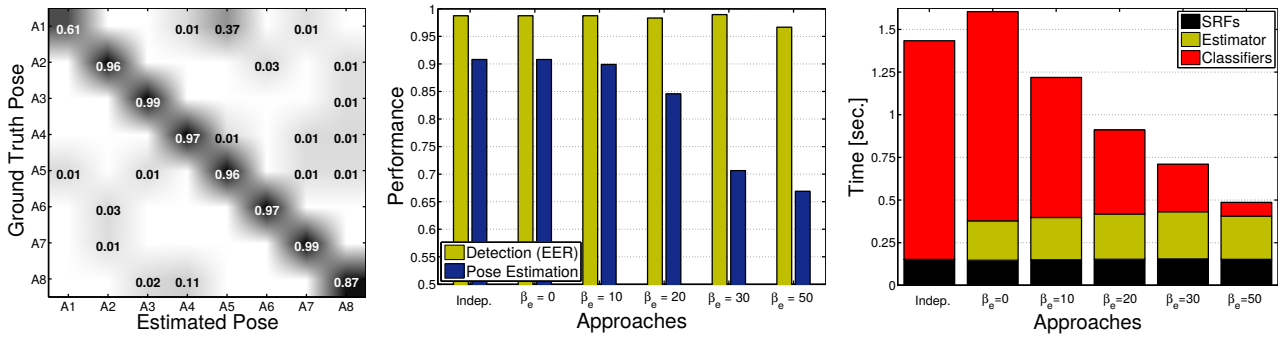

Fig. 15: BRFs Perfomance in multiview object detection for the car class. Left: confusion matrix for object orientation estimation. Middle: detection and pose estimation accuracy as the parameter β_e is increased, limiting the number of hypotheses that are trimmed from the Hough map. Right: decrease in computational speed for the two steps in the algorithm, as a function of β_e.

Method	Cars		Bicycles	
	Detec.	View.	Detec.	View.
Bao et al. [2]	98.0	95.3	93.1	92.3
Glasner et al. [18]	99.2	85.3	-	-
Liebelt et al. [30]	76.7	70.0	69.8	75.0
Stark et al. [46]	89.9	81.0	-	-
Su et al. [47]	55.3	67.0	-	-
Xiang et al. [62]	98.4	93.8	93.0	91.4
BRFs	98.9	91.5	94.6	89.2
OBRFs	98.9	90.2	92.9	86.2

TABLE 7: Multi-view object detection rates and view pose recognition accuracy on the 3DObject dataset.

	EPFL Dataset		KITTI Dataset		
Method	Detec.	View.	Easy	Moder.	Hard
Felzenszwalb [10]	98.1	56.6	72.2	46.0	35.4
Glasner et al. [18]	89.5	-	-	-	-
Ozuysal et al. [40]	85.4	41.6	-	-	-
Xiang et al. [62]	96.4	64.8	-	-	-
BRFs	90.1	60.1	72.4	56.0	41.5

TABLE 8: Multi-view object detection rates and view pose recognition accuracy on the EPFL and KITTI cars datasets.

for verification. The first column in the plots (Indep.) represents the case when no prior pose estimation is made, and window-sliding is used to feed all possible locations and orientations to the classifiers. The center plot indicates EER detection and pose estimation rates, whereas the plot to the right shows overall computation times. Note that the time spent on computing the shared random ferns (SRFs) is constant regardless of the amount of pose hypotheses to be tested. Limiting the amount of hypotheses significantly reduces computational cost at the expense of impoverished orientation estimation rates. Nonetheless, the detection rates remain unaltered.

Similar to previous experiments for multi-view car detection, the proposed approach is evaluated in the EPFL Car dataset [40]. This dataset contains 20 car instances rotating on a platform. The first 10 cars are used to train the pose estimator and 32 pose-specific classifiers (at 16 different angles and with two aspect ratios per orientation), whereas the remaining 10 cars are used during testing.

Once more, our method achieves significant performance when compared to the state of the art as depicted in Table 8. The reported values were attained for $\beta_e = 10$, and as in the previous case, averaging over five training-testing iterations to account for randomness, and also using the 2010 Pascal protocol for the computation of the AP metric. Contrary to other works that require expensive features combined with more complex learning and detection strategies, as is the case of the Deformable Part Models (DPMs) approach [10] that obtains the best detection rates, our method is able to train a pose-specific classifier in a couple of minutes and to detect cars in less than one second despite difficult scene conditions.

Finally, we have performed a one-to-one comparison against the DPMs with a subset of the KITTI Dataset [17], for multi-view car detection. We have split the $7,481$ original training images into a training set of $5,000$ images and a test set of $2,481$ images. For the DPMs we use the code and pretrained detectors publicly available at http:/www.cs.berkeley.edu/~rbg/latent/. Table 8 summarizes the results of this experiment for different levels of difficulty considered in the evaluation: easy, moderate and hard. Observe that in this case BRFs even slightly outperform DPMs, despite BRFs are using much simpler features and constraints when building the classifier. In Appendix 5 we show some detection results.

5 Conclusions

In this work, we introduced Boosted Random Ferns (BRFs), an efficient and effective classifier for the detection of object categories. Four key aspects make the method particular. On the one hand, we propose to use as features sets of random binary comparisons in the HOG domain. We call them HOG-based ferns. These features are very fast to compute and are very effective to model intra-class appearance variability. The second key contribution of the method is the idea of boosting these features to select the most discriminant ones. Boosting identifies the most relevant regions within each class (bike wheels, horse's neck and head, etc.). The third key contribution is the idea of sharing features amongst weak learners, and also during classifier cascading. Feature sharing provides remarkable computational advantages when compared to the state of the art. Finally, we show that the classifier can be even trained online,

with sequentially arriving data. As future work we plan introducing BRFs within a DPM formulation, or even in a Deep Network (e.g. in the first convolutional layer), to combine the strengths of all these methods. We also believe that our features can be appropriate to tackle problems involving RGB-D data.

APPENDIX A

2D Classification Problem

This section reports several additional synthetic experiments where we compare the performance of the Boosted Random Ferns (BRFs) proposed in this paper against other related approaches, namely Random Ferns (RFs) [38], Random Forests (RForest) [6] and Gentle Boost (GBoost) [13].

We perform two additional experiments. In the first one (Fig. 16 and Table 18) the two classes are linearly separable. In the second experiment (Fig. 17 and Table 19) the two classes are not linearly separable. In both cases, the results of RForest and BRFs clearly deliver much better classification rates than RFs and GBoost. Note, however, that BRFs are significantly much faster than RForest, both in train and test.

APPENDIX B

Ferns Selection: A Simple Example

In Fig. 20 we show the process of how the four first ferns of a BRFs classifier are chosen, in an synthetic two class problem, where classes are linearly separable. The type of binary features and experimental setup are the same as to those described in Section 3.3. For visualization purposes, we reduced the training set to 100 positive and 100 negative samples. At the top of the figure we plot the equivalent tree-like structure of every fern and their corresponding parameters, chosen from a pool of 500 random values. The middle row depicts the weights $w_t(i)$ for the 200 training samples, computed using Eq. 13 after each boosting step. The bottom-most graphs plot the classification results, where the size of each sample is proportional to its weight. Note that larger weights at step t usually correspond to misclassified samples or samples lying near the frontier that separates the two classes. The subsequent classifier at step $t+1$ will prioritize the correct classification of theses samples. By doing this, we prevent correlation (i.e, selection of the same decision boundary) among different ferns, a key aspect for the generalization capabilities of the classifier.

APPENDIX C

KITTI Dataset: Detection Examples

Fig. 21 shows the output of the BRFs on some sample images. The green boxes represent the location and scale of detected cars while the pie-like circles indicate the estimated orientation of every car. False detections are shown with red boxes. Note that the proposed method is able to detect cars at multiple scales and under complex lighting conditions and mild occlusions.

ACKNOWLEDGMENTS

This work was partially supported by the Catalan Agency for Management of University and Research Grants for the Consolidated Group VIS (2014 SGR 897). A.S. acknowledges support also from project ColRobTransp (DPI2016-78957-R), and J.A. and F.M. from project RobInstruct (TIN2014-58178-R), both funded by the Spanish Ministry of Economy, Industry and Competitiveness. J.A., A.S., and F.M. were also partially supported by the European Union projects AEROARMS (H2020-ICT-2014-1-644271) and LOGIMATIC (H2020-Galileo-2015-1-687534).

REFERENCES

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Europ. Conf. on Computer Vision*, pages 113–130, 2002.

[2] S.Y. Bao, Y. Xiang, and S. Savarese. Object co-detection. In *Europ. Conf. on Computer Vision*, pages 86–101, 2012.

[3] O. Barinova, V. Lempitsky, and P. Kholi. On detection of multiple object instances using Hough transforms. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 34(9):1773–1784, 2012.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comp. Vision and Image Underst.*, 110(3):346–359, 2008.

[5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Int. Conf. in Machine Learning*, 2006.

[6] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2):81–227, 2011.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 886–893, 2005.

[8] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 617–624, 2011.

[9] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 2241–2248, 2010.

[10] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 32(9):1627–1645, 2010.

[11] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 30(1):36–51, 2008.

[12] V. Ferrari, F. Jurie, and C. Schmid. Accurate object detection with deformable shape models learnt from images. In *Conf. on Comp. Vision and Pattern Recogn.*, 2007.

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

[14] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *Int. Conf. on Comp. Vision*, pages 1363–1370, 2005.

[15] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1022–1029, 2009.

[16] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 33:2188–2202, 2011.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *Int. Conf. on Comp. Vision*, pages 1275–1282, 2011.

[19] H. Grabner and H. Bischof. On-line boosting and vision. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 260–267, 2006.

[20] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *Europ. Conf. on Computer Vision*, pages 408–421, 2010.

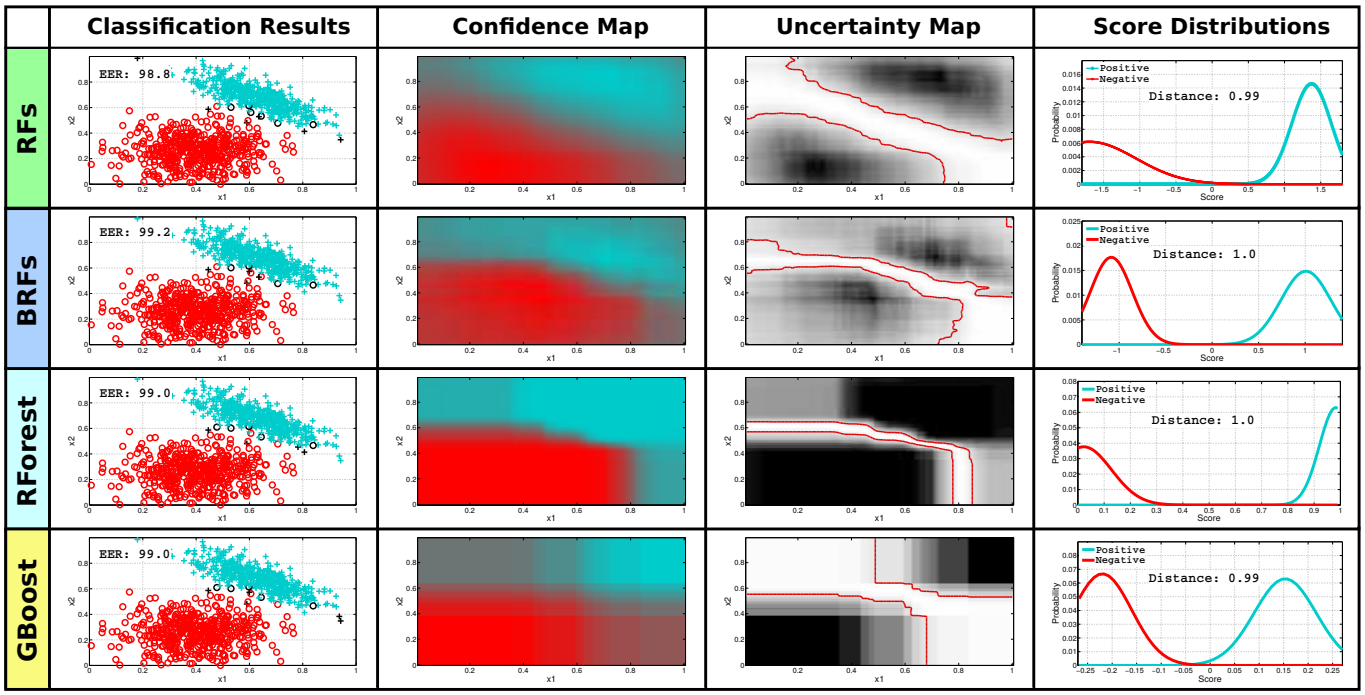

Fig. 16: 2D classification results of BRFs vs. RFs [38], Random Forest [6] and GentleBoost classifier [13]. First column: classification results on testing samples. Misclassified samples are shown in black. Second column: confidence maps provided by the classifiers over the 2D feature space. Third column: uncertainty maps where brighter regions correspond to uncertain classification values. Red contours indicate uncertainty of 90%. Fourth column: score distributions for the positive and negative classes.

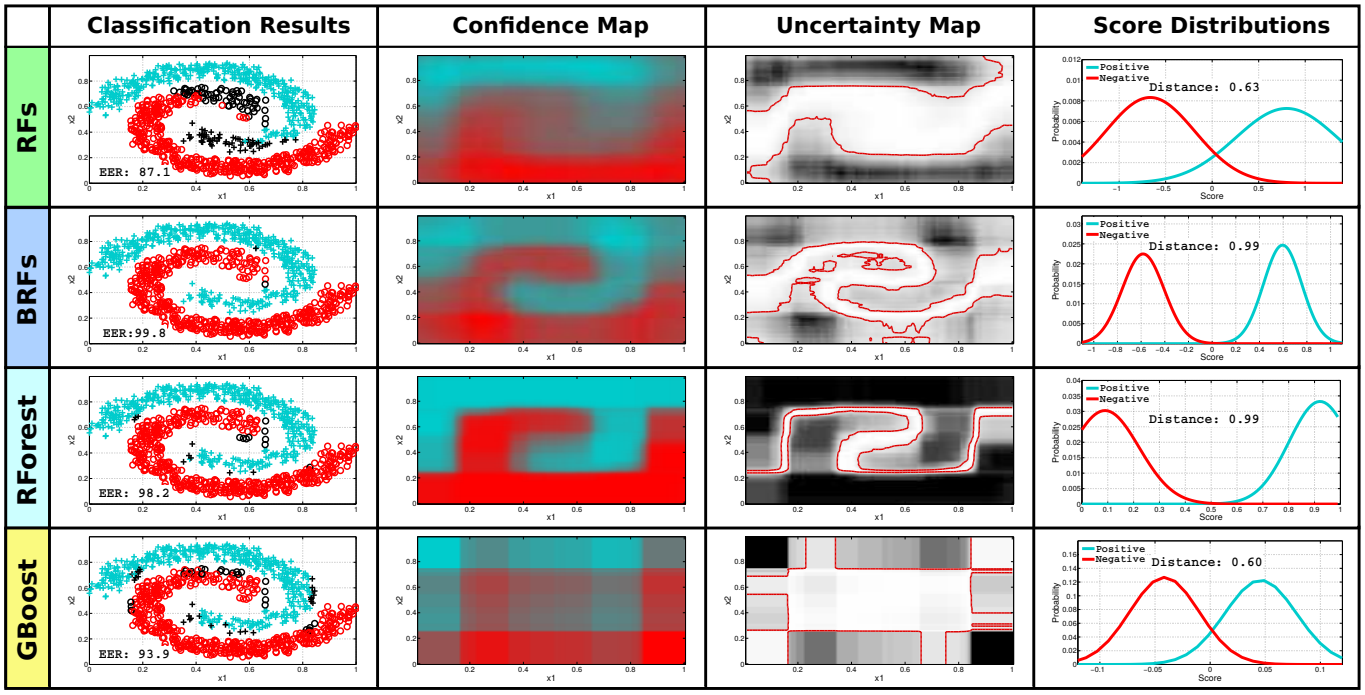

Fig. 17: 2D classification results of BRFs vs. RFs [38], Random Forest [6] and GentleBoost classifier [13]. First column: classification results on testing samples. Misclassified samples are shown in black. Second column: confidence maps provided by the classifiers over the 2D feature space. Third column: uncertainty maps where brighter regions correspond to uncertain classification values. Red contours indicate uncertainty of 90%. Fourth column: score distributions for the positive and negative classes.

[21] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 49–56, 2010.

[22] T-K. Kim and R. Cipolla. Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *Neural Information Processing Systems*, volume 11, pages 841–848, 2008.

[23] E. Krupka, A. Vinnikov, B. Klein, A.B. Hillel, D. Freedman, and S. Stachniak. Discriminative ferns ensemble for hand pose recognition. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 3670–3677, 2014.

[24] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In

	Equal Error Rate [%]				Hellinger Distance [%]				Training Time [sec.]				Run Times [msec.]				D
RFs	76.5	71.9	74.3	73.8	33.7	30.9	32.0	34.5	0.01	0.01	0.02	0.02	0.03	0.02	0.02	0.03	1
	75.4	78.7	76.9	76.6	31.2	38.5	40.5	40.9	0.01	0.01	0.02	0.02	0.02	0.02	0.03	0.03	2
	80.0	80.5	81.2	80.0	41.7	48.0	49.1	49.9	0.01	0.02	0.02	0.04	0.02	0.03	0.03	0.05	4
	84.6	84.8	85.4	86.8	55.4	58.4	60.8	63.1	0.01	0.02	0.03	0.05	0.02	0.03	0.04	0.06	6
BRFs	92.2	94.8	95.6	95.8	55.2	61.4	63.4	64.3	0.30	0.62	1.22	2.37	0.02	0.03	0.02	0.03	1
	98.2	98.9	99.1	99.0	80.5	91.3	90.4	89.5	0.30	0.64	1.23	2.38	0.02	0.02	0.02	0.03	2
	98.9	99.5	99.6	99.6	94.4	97.6	98.1	98.1	0.35	0.71	1.30	2.48	0.02	0.02	0.03	0.04	4
	99.5	99.6	99.7	99.9	99.3	99.7	99.8	99.8	0.45	0.79	1.40	2.58	0.02	0.03	0.04	0.06	6
RForest	79.3	79.9	75.0	74.7	35.0	34.8	36.2	35.4	0.19	0.37	0.74	1.49	0.28	0.59	1.16	2.34	1
	80.1	78.8	76.7	77.5	32.8	33.0	33.7	33.5	0.41	1.01	2.00	3.98	0.37	0.85	1.67	3.34	2
	91.2	91.8	92.3	91.9	73.9	74.0	77.6	77.6	1.15	2.77	5.45	11.47	0.50	1.19	2.39	4.93	4
	97.7	96.7	97.5	97.4	98.5	96.7	98.5	98.0	1.94	4.58	9.56	18.76	0.61	1.48	2.97	6.05	6
GBoost	89.8	92.5	94.5	95.1	55.4	58.9	60.3	61.3	0.01	0.01	0.02	0.04	0.02	0.02	0.02	0.02	1
# WLs	10	25	50	100	10	25	50	100	10	25	50	100	10	25	50	100	

Fig. 18: 2D Classification Performance. Left: mean EERs on the precision-recall curve for the RFs and BRFs classification methods in the two scenarios shown in Fig. 5. Right: equal error rates on the precision-recall curve for the Boosted Random Ferns and Random Ferns classification methods in Scenario B, and varying number of weak learners and features. The plot shows mean EERs values and their standard deviation.

	Equal Error Rate [%]				Hellinger Distance [%]				Training Time [sec.]				Run Times [msec.]				D
RFs	94.5	98.5	98.4	99.0	69.7	91.7	85.7	95.0	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.03	1
	98.0	98.8	99.3	99.2	88.2	95.9	95.9	96.3	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03	2
	98.9	99.2	99.3	99.4	96.2	98.0	98.8	99.0	0.01	0.02	0.02	0.03	0.02	0.03	0.03	0.04	4
	99.1	99.3	99.2	99.4	98.9	99.6	99.7	99.7	0.01	0.02	0.03	0.05	0.02	0.03	0.04	0.06	6
BRFs	99.0	99.2	98.9	99.1	99.1	99.5	99.0	99.2	0.28	0.62	1.21	2.36	0.02	0.02	0.02	0.03	1
	99.2	99.1	99.5	99.1	99.8	99.7	99.9	99.8	0.29	0.64	1.22	2.37	0.02	0.02	0.02	0.03	2
	99.1	99.3	99.4	99.3	100	100	100	100	0.35	0.70	1.29	2.46	0.02	0.02	0.03	0.04	4
	99.1	99.3	99.2	99.3	100	100	100	100	0.43	0.80	1.38	2.57	0.02	0.03	0.04	0.06	6
RForest	97.1	97.3	96.8	96.8	99.1	99.3	98.7	98.6	0.17	0.37	0.75	1.47	0.26	0.60	1.17	2.28	1
	98.3	98.1	98.5	98.1	99.9	99.9	100	99.9	0.41	1.03	2.03	4.00	0.37	0.86	1.69	3.36	2
	98.9	99.2	99.3	99.2	100	100	100	100	0.89	2.28	4.41	8.89	0.41	1.01	1.93	4.01	4
	99.0	99.3	99.2	99.2	100	100	100	100	1.01	2.67	5.06	10.25	0.44	1.09	2.09	4.27	6
GBoost	99.0	99.3	99.0	99.1	99.1	99.4	98.7	98.8	0.01	0.01	0.02	0.04	0.02	0.02	0.02	0.02	1
# WLs	10	25	50	100	10	25	50	100	10	25	50	100	10	25	50	100	

Fig. 19: 2D Classification Performance. Left: mean EERs on the precision-recall curve for the RFs and BRFs classification methods in the two scenarios shown in Fig. 5. Right: equal error rates on the precision-recall curve for the Boosted Random Ferns and Random Ferns classification methods in Scenario B, and varying number of weak learners and features. The plot shows mean EERs values and their standard deviation.

Conf. on Comp. Vision and Pattern Recogn., pages 1–8, 2008.

[25] I. Laptev. Improving object detection using boosted histograms. *Image and Vision Computing*, 27(5):535–544, 2009.

[26] A. Lehmann, P. V. Gehler, and L. Van Gool. Branchandrank: Non-linear object detection. In *British Mach. Vision Conf.*, 2011.

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

[28] B. Leibe, K. Mikolajczyk, and B. Schiele. Segmentation based multi-cue integration for object detection. In *British Mach. Vision Conf.*, pages 1169–1178, 2006.

[29] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 28(9):1465–1479, 2006.

[30] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1688–1695, 2010.

[31] K. Liu, Q. Wang, W. Driever, and O. Ronneberger. 2D/3D rotation-invariant detection using equivariant filters and kernel weighted mapping. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 917–924, 2012.

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

[33] S. Maji and J. Malik. Object detection using a max-margin Hough transform. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1038–1045, 2009.

[34] T. Malisiewicz, A. Gupta, and A.A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Int. Conf. on Comp. Vision*, pages 89–96, 2011.

[35] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 26–36, 2006.

[36] T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 30(7):1257–1269, 2008.

[37] A. Monroy, A. Eigenstetter, and B. Ommer. Beyond straight lines-object detection using curvature. In *Int. Conf. on Image Processing*, pages 3561–3564, 2011.

[38] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 32(3):448–461, 2010.

[39] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1–8, 2007.

[40] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 778–785, 2009.

[41] H. Riemenschneider, M. Donoser, and H. Bischof. Using partial edge contour matches for efficient object category localization. In *Europ. Conf. on Computer Vision*, pages 29–42, 2010.

[42] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *Int. Conf. on Comp. Vision*, pages 1–8, 2007.

[43] R.E. Schapire and Y. Singer. Improved boosting algorithms using

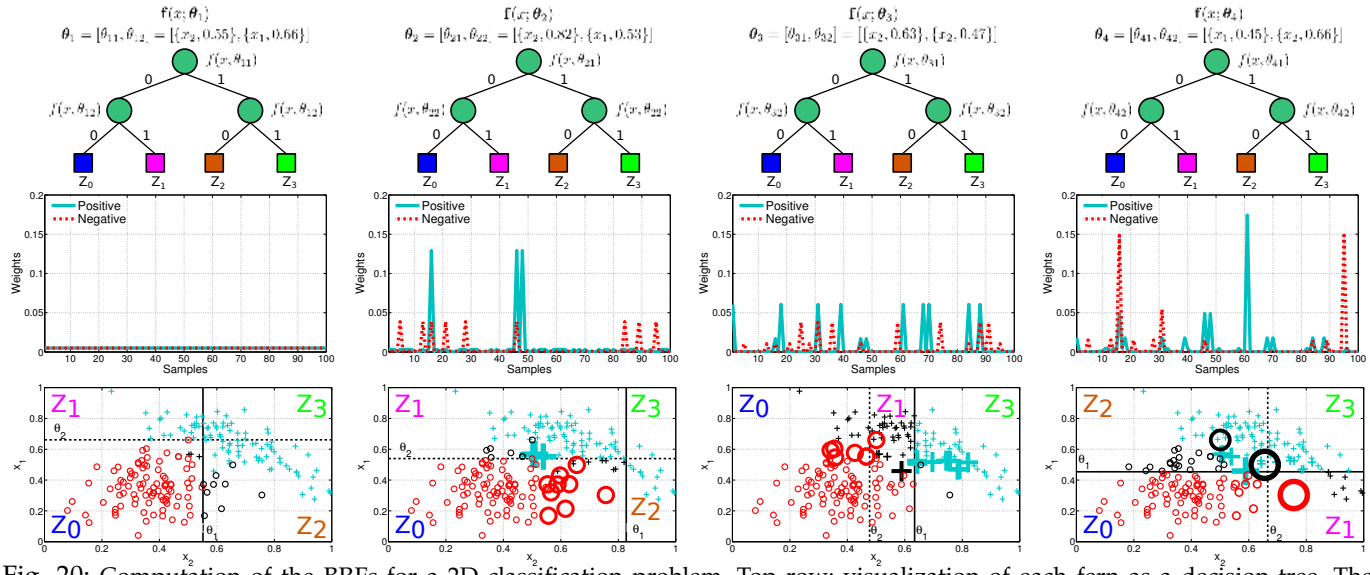

Fig. 20: Computation of the BRFs for a 2D classification problem. Top row: visualization of each fern as a decision tree. The parameters for the i-th fern are $\boldsymbol{\theta}_i = \{\theta_{i1}, \theta_{i2}\}$, where θ_{ij} corresponds to a pair $\{\text{axis}, \tau\}$ chosen from a large pool of random values. Middle row: Weight associated to the 200 training samples (100 positive and 100 negative) after each boosting step. Bottom row: Classification response of each weak learner. Black circles and crosses represent misclassified samples, and the size of each sample is proportional to its associated weight. Additionally, we indicate the spatial coverage of each output $\{z_1, z_2, z_3, z_4\}$, i.e, the samples within the region z_0 fill end-up on the z_0 leaf of the classification tree.

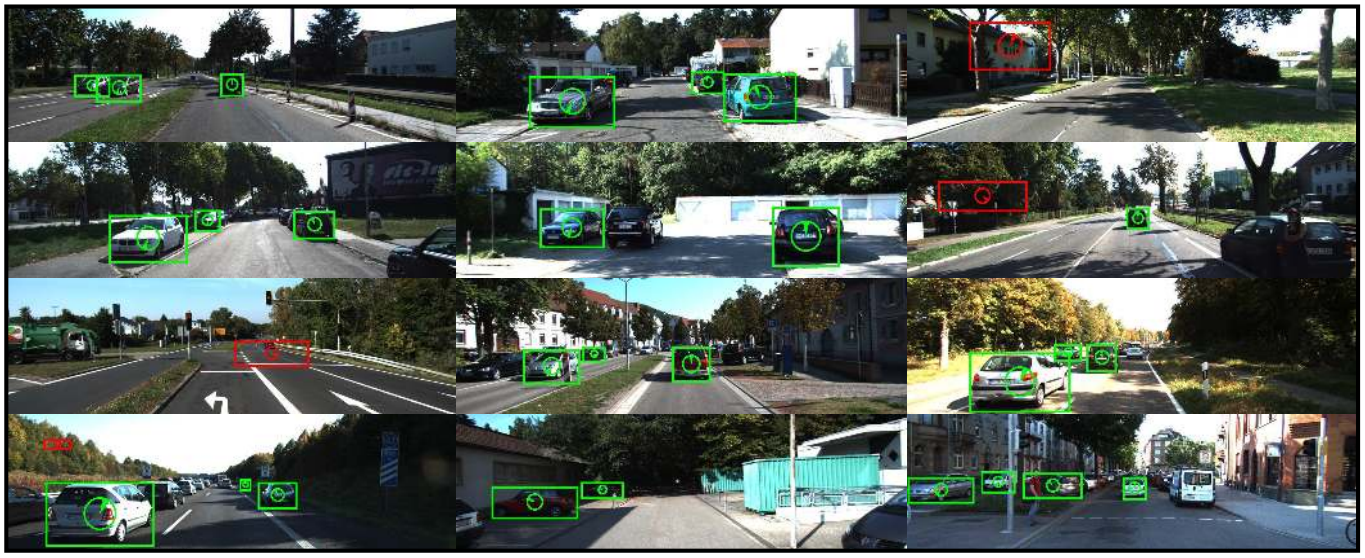

Fig. 21: Detection results obtained with the Boosted Random Ferns on the KITTI Car dataset [17].

confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[44] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 30(7):1270–1281, 2008.

[45] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1–8, 2008.

[46] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3D CAD data. In *British Mach. Vision Conf.*, pages 106.1–106.11, 2010.

[47] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *Int. Conf. on Comp. Vision*, pages 213–220, 2009.

[48] D. Tang, Y. Liu, and T-K. Kim. Fast pedestrian detection by cascaded random forest with dominant orientation templates. In *British Mach. Vision Conf.*, pages 1–11, 2012.

[49] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Analylis and Machine Intelligence*, 29(5):854–869, 2007.

[50] A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 950–957, 2010.

[51] M. Villamizar, J. Andrade-Cetto, A. Sanfeliu, and F. Moreno-Noguer. Bootstrapping boosted random ferns for discriminative and efficient object classification. *Pattern Recognition*, 45(9):3141–3153, 2012.

[52] M. Villamizar, H. Grabner, J. Andrade-Cetto, A. Sanfeliu, L. Van Gool, and F. Moreno-Noguer. Efficient 3D object detection using multiple pose-specific classifiers. In *British Mach. Vision Conf.*, pages 20.1–20.10, 2011.

[53] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Efficient rotation invariant object detection using boosted random ferns. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1038–1045, 2010.

[54] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Shared random ferns for efficient detection of multiple categories. In *Int. Conf. on Pattern Recognition*, 2010.
[55] M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto. Computation of rotation local invariant features using the integral image for real time object detection. In *Int. Conf. on Pattern Recognition*, 2006.
[56] M. Villamizar, A. Sanfeliu, and F. Moreno-Noguer. Fast online learning and detection of natural landmarks for autonomous aerial robots. In *Int. Conf. on Robotics and Automation*, 2014.
[57] M. Villamizar, A. Sanfeliu, and F. Moreno-Noguer. Interactive multiple object learning with scanty human supervision. *Comp. Vision and Image Underst.*, 149:51–64, 2016.
[58] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 511–518, 2001.
[59] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
[60] X. Wang, T.X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *Int. Conf. on Comp. Vision*, pages 32–39, 2009.
[61] B. Wu and R. Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *Int. Conf. on Comp. Vision*, pages 1–8, 2007.
[62] Y. Xiang and S. Savarese. Estimating the aspect layout of object categories. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 3410–3417, 2012.
[63] P. Yarlagadda, A. Monroy, and B. Ommer. Voting by grouping dependent parts. In *Europ. Conf. on Computer Vision*, pages 197–210, 2010.
[64] J. Zhang, K. Huang, Y. Yu, and T. Tan. Boosted local structured HOG-LBP for object localization. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1393–1400, 2011.
[65] Z. Zhang, J. Warrell, and P. Torr. Proposal generation for object detection using cascaded ranking SVMs. In *Conf. on Comp. Vision and Pattern Recogn.*, pages 1497–1504, 2011.

Alberto Sanfeliu received the Engineer and PhD degrees from the Universitat PolitÃ¨cnica de Catalunya (UPC), Spain, in 1978 and 1982 respectively. He joined the faculty of UPC in 1981 and is full professor of Computational Sciences and Artificial Intelligence. He is coordinator of the research Artificial Vision and Intelligent System Group (VIS) and head of the research line Mobile Robotics of (IRI) Institut de Robòtica i Informàtica Industrial (UPC-CSIC). He is former director of the Institut de Robòtica i Informàtica Industrial, UPC-CSIC, former director of the UPC Automatic Control Department, and past president of AERFAI, (Spanish Association for Pattern Recognition). He has worked on various theoretical aspects on pattern recognition, computer vision and robotics and on applications on vision defect detection, tracking, recognition, robot vision, SLAM and autonomous vehicles. He has several patents on quality control based on computer vision. He has authored books in pattern recognition and SLAM, and published more than 260 papers in international journals and conferences. He has lead and participated in 45 RD projects, 16 of them funded by the European Commission, and he was the coordinator of the European project URUS (Ubiquitous Networking Robotics in Urban Areas). He has worked in autonomous driving in the European projects (CargoAnts and Logimatic) and with Volkswagen Research in a CARNET project. He is member of editorial boards of top scientific journals in computer vision and pattern recognition and has been the General Co- Chairmen of the top international congresses of Computer Vision (ICCV 2011) and Pattern Recognition (ICPR 2000). He received the prize to the Technology given by the Generalitat de Catalonia and is Fellow of the International Association for Pattern Recognition.

Michael Villamizar Vergel is a postodoctoral researcher at the IDIAP Research Institute in Martigny (Switzerland). Before, he was a postdoctoral researcher at the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, in Barcelona (Spain). He obtained his PhD in computer vision and robotics from the Universitat Politècnica de Catalunya in 2012. He also received the BSc degree in mechatronics engineering from San Buenaventura University (Colombia) in 2004. Michael has participated in diverse national and European projects and has worked with other research groups in Europe. His research interests are focused on object detection and categorization, robust visual tracking, and real-time robotics applications.

Francesc Moreno-Noguer received the MSc degrees in industrial engineering and electronics from the Universitat Politècnica de Catalunya and the Universitat de Barcelona in 2001 and 2002, respectively, and the PhD degree from UPC in 2005. From 2006 to 2008, he was a postdoctoral fellow at the computer vision departments of Columbia University and the École Polytechnique Fédérale de Lausanne. In 2009, he joined the Institut de Robòtica i Informàtica Industrial in Barcelona as an associate researcher of the Spanish National Research Council. His research interests include retrieving rigid and nonrigid shape, motion, and camera pose from single images and video sequences. He received UPC's Doctoral Dissertation Extraordinary Award for his work.

Juan Andrade-Cetto is Associate Researcher of the Spanish National Research Council and Director of the Institut de Robòtica i informàtica Industrial CSIC-UPC. His research addresses state estimation and computer vision problems with applications to mobile robotics. He obtained the BSEE degree from CETYS Universidad, Mexico, in 1993, the MSEE degree from Purdue University, USA, in 1995, and the PhD degree in Systems Engineering from the Universitat Politècnica de Catalunya, Spain, in 2003, and is the recipient of the EURON Georges Giralt Best PhD Award.