Interactive Multiple Object Learning with Scanty Human Supervision

Michael Villamizar, Anaís Garrell, Alberto Sanfeliu and Francesc Moreno-Noguer

Institut de Robòtica i Informàtica Industrial, CSIC-UPC
{mvillami,agarrell,sanfeliu,fmoreno}@iri.upc.edu

Abstract

We present a fast and online human-robot interaction approach that progressively learns multiple object classifiers using scanty human supervision. Given an input video stream recorded during the human-robot interaction, the user just needs to annotate a small fraction of frames to compute object specific classifiers based on random ferns which share the same features. The resulting methodology is fast (in a few seconds, complex object appearances can be learned), versatile (it can be applied to unconstrained scenarios), scalable (real experiments show we can model up to 30 different object classes), and minimizes the amount of human intervention by leveraging the uncertainty measures associated to each classifier.

We thoroughly validate the approach on synthetic data and on real sequences acquired with a mobile platform in indoor and outdoor scenarios containing a multitude of different objects. We show that with little human assistance, we are able to build object classifiers robust to viewpoint changes, partial occlusions, varying lighting and cluttered backgrounds.

Keywords: Object recognition, interactive learning, online classifier, human-robot interaction.

1. Introduction

Over the last decade we have witnessed the enormous progress in the field of object recognition and classification in images and video sequences. At present, there are methods that produce impressive results in a wide variety of challenging scenarios corrupted by lighting changes, cluttered backgrounds, partial occlusions, viewpoint and scale changes, and large intra-class variations [4, 15, 25, 33, 45, 48, 51]

This progress in object recognition has had a positive impact in many application fields such as robotics, where computer vision algorithms have been used for diverse robotics tasks such as object recognition and grasping [3, 5], detection and tracking of people in urban settings [10, 34, 39], human-robot interaction [40, 47], and robot localization and navigation [11, 16, 26].

The standard method for recognizing objects in images consists in computing object specific classifiers during an offline and time-consuming training step where large amounts of annotated data are used to build discriminative and robust object detectors [15, 33]. However, there are situations in which offline learning is not feasible, either because the training data is obtained continuously, or because the size of the training data is very cumbersome, and a batch processing becomes impractical. This is particularly critical in some robotics applications, specially those related to human-robot interaction, where the robots need to compute object detectors on the fly, in real time, and with very little training data.

In these cases, online learning methods which use their own predictions to compute and update a classifier have been proposed [20, 21, 35]. Yet, although these approaches have shown great adaptation capabilities, they are prone to suffer from drifting when the classifier is updated with wrong predictions. This has been recently addressed by combining offline and online strategies [17, 28], semi-supervised boosting methods [22], or by using human intervention during learning so as to assist the classifier in uncertain classification cases [52, 57].

In preliminary versions of this work, we already proposed online object detection approaches in which the human assistance is integrated within the learning loop in an active and efficient manner [19, 52, 53]. In [19, 52] the proposed approach was focused for single object detection, and further extended in [53] to multiple instances using an adaptive uncertainty classification threshold that reduces the human supervision.

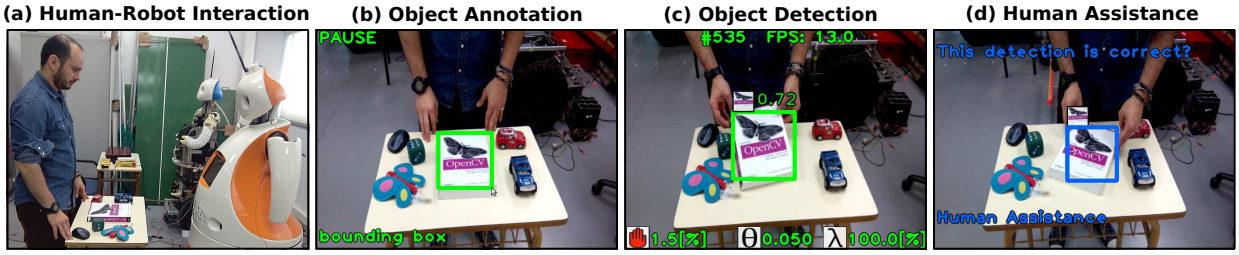


Figure 1: Interactive and real-time approach for learning and detecting multiple objects through human-robot interaction. (a) Interplay between the robot and the human user. (b) Object annotation using a bounding box provided by the user (green box). (c) Output of the proposed detection method (green box). (d) Human intervention for unknown and difficult cases. The user provides the label for current object prediciton (blue box).

In this paper, we unify the formulation of these previous works and perform a more in-depth analysis of the method presented in [53] through additional experiments in synthetic and real scenarios, while providing more comparisons against competing approaches.

More precisely, we propose a fast and online approach that interactively models several object appearances on the fly, using as few human interventions as possible, and still keeping the real-time efficiency [53]. At the core of our approach, there is a randomized tree classifier [12, 37, 38] that is progressively computed using its own detection predictions. Yet, to avoid feeding the classifier with false positive samples (i.e, drifting), we propose an uncertainty-based active learning strategy [32, 46] that gradually minimizes the amount of human supervision and keeps high classification rates. Note that this issue is critical in order to maintain long-term interactions with robots, as if the robot keeps asking for annotating images insistently, people tend to quickly give up the interaction [19, 41].

To make the proposed approach scalable for various object instances, multiple object specific classifiers are computed in parallel, but sharing the same features in order to maintain the efficiency of the method and to reduce the computational complexity in run time [54].

As an illustrative example, Fig. 1 shows the operation of the proposed interactive method to learn and detect multiple object instances through human-robot interaction (Fig. 1-a). Each time the human user seeks to model a new object of interest, he/she marks a bounding box around the object in the input image, via a mouse, keyboard or touchscreen (see Fig. 1-b). The robot initializes a model for this new object and runs a detector on subsequent frames for this, and the rest of objects in the database (Fig. 1-c). When the robot is not confident enough about the detections and class predictions, it requests the human assistance to provide the true class labels, which, in turn, are used to update the classifier, observe Fig. 1-d. This procedure is performed continuously, and at each iteration, the performance and confidence of the classifier is increased whereas the degree of human intervention is reduced significantly.

The remainder of the paper is organized as follows: Sec. 2 describes the related work and our contributions while Sec. 3 explains the proposed approach with all its main ingredients. Sec. 4 describes the experiments conducted to evaluate the proposed learning approach. We report results using synthetic an real data. The former are used to thoroughly assess the limits of the method in terms of number of classes it can handle or classification rate. Real experiments demonstrate the applicability of the method for diverse perception tasks in challenging scenarios.

2. Related Work and Contributions

In this section, we show and discuss our contributions along with the related work on the three main topics concerned with the proposed approach: human-robot interaction, the computation of online classifiers, and interactive learning techniques:

2.1. Human-Robot Interaction

Computer vision techniques for human-robot interaction have been mainly focused on recognizing people in urban scenarios [10, 34, 39] as well as identifying human gestures and activities [13, 36] to establish contact with people and perform particular robotics tasks such as guiding people in museums and urban areas [18, 42, 49], providing information in shopping malls [23], or recognizing human emotions through classifying facial gestures [9]. Although these techniques have endowed the robot with remarkable interaction skills, they are commonly computed offline and using a potentially large training time. As a result, the robot is limited to perform tasks only for which it has been trained previously, missing the opportunity to learn and improve its perception skills through the interaction with humans.

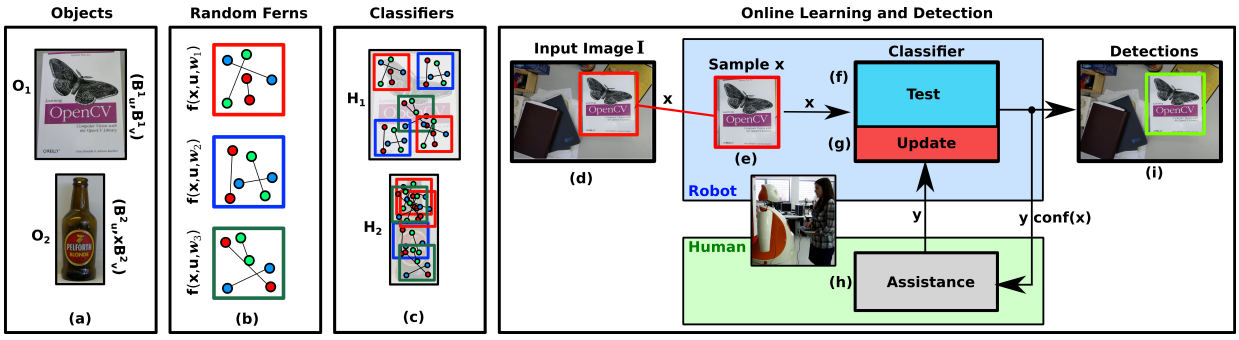


Figure 2: General schemes of the proposed interactive learning approach for online object recognition. (a) Two object instances with different sizes. (b) Computation of random ferns where each one is specific set of pixel-intensity comparisons (pairs of colored dots). (c) Two different object classifiers computed by combining ferns at multiple image locations. (d) Online learning using human-robot interactions. The human assists the robot when it is not certain about its sample class prediction.

Conversely, in this work we propose a very efficient interactive approach that combines human assistance and robot's detection predictions so as to build object detectors which can be applied for a wide range of robotics tasks. The approach exploits the interplay between robots and humans in order to compute and improve progressively the robot's perception capabilities.

2.2. Online Classifiers

Despite showing impressive results, standard methods for object detection compute the classifiers using intensive and offline learning approaches applied to large annotated datasets [15, 33, 37, 51]. Therefore, most of these offline approaches are not suitable for some particular applications requiring computing the classifier on the fly, either because the training data is obtained continuously, or because the size of the training data is so large that it needs to be loaded progressively. To handle these situations, several online alternatives allowing to sequentially train the classifiers have been proposed [6, 8, 21, 24, 44].

In this work, the classifier we use is based on an online random ferns formulation [28, 31, 37, 52, 53], which has been showing excellent results, both in terms of classification rates as computational efficiency. In Fig. 2 we show the overall schemes of the proposed interactive method and the online classifier. In essence, this classifier computes several sets of intensity-based pixel comparisons (see Fig. 2-b) to build the randomized trees which are then used to estimate the posterior class probabilities.

Most previous online versions are focused to single object modeling and tracking [6, 8, 21, 52]. In order to learn multiple models we propose computing simultaneously and in parallel multiple classifiers, one for each object class (Fig. 2-a), and with specific configurations like the spatial distribution of ferns or the particular object size (observe Fig. 2-c). This also differs from other state of the art classifiers, that when applied to multiclass problems they require objects with constant aspect ratios and to know the number of object classes in advance [50]. In this respect, our method scales better for multiple objects since each object is learned by separated at the time in which the user selects a new object model in the video stream. This allows learning and detecting up to 30 object classes in an efficient and dynamic manner.

2.3. Interactive Learning

Active learning techniques have been extensively used in computer vision to reduce the number of training samples that need to be annotated when building a classifier [46]. Approaches such as "query by committee" [1, 27], and "uncertainty-based sampling" [32] close the learning loop using human assistance. In these works, the human user acts as an oracle that annotates/labels those samples that the classifier is not quite confident about their class prediction.

In this paper, we propose an interactive learning strategy in which the robot plays a more active role, that is, the discriminative classifiers are built using a combination of the robot predictions with the human assistance (see Fig. 2-f,g,h). Additionally, we also propose a methodology based on an adaptive uncertainty threshold that progressively reduces the amount of human assistance, making a more "enjoyable" human-robot interaction. This is also another difference with respect to our own previous works [19, 52]. As it will be shown in the experimental section, using an adaptive threshold we can learn and detect several object instances without decrementing the intra-class classification rates.

Robot Utterances	
Type of utter	**Example**
Greeting	Nice to meet you Can you teach me to detect faces/objects?
Assistance	Is your face inside the green rectangle? Is this object detection correct?
No detection	I can't see you, move a little bit. Can you stand in front of me?
Farewell	Thank you for your help, nice to meet you I hope I see you soon.

Table 1: Sample phrases uttered by the robot.

After having discussed the related work, we can summarize the main *contributions* of our approach as follows: (1) Proposing an online approach to learn and detect multiple object instances in images; (2) Designing an interactive learning strategy that incrementally improves the discrimination power of the classifiers using human assistance; (3) An adaptive learning scheme to reduce gradually the human interventions; and (4) A real-time implementation of the algorithm, which can cope with multiple objects at several frames per second.

3. Interactive Object Learning and Recognition

In this section, the main components of the proposed learning and detection strategy are described in more detail. Fig. 2 shows an schematic of how these elements are related.

3.1. Human-Robot Interaction

To perform online learning of object instances, we consider a scenario in which the classifiers are learned using a computer onboard a mobile robot, equipped with devices such as a keyboard, mouse, and a screen that enable the interaction with the human. We refer the reader to Fig. 1 for an illustration. Specifically, the keyboard and mouse are used to annotate the object of interest that the user wants to learn, and also to attend the robot in situations where the robot is not confident in its predictions. On the other hand, the touch screen is used to display the output of the detector and to show the performance of the detection system.

In those situations where the robot is uncertain about its detection hypotheses, the robot will formulate to the user a set of concise questions, that expect for a 'yes' or 'not' answer. Table 1 shows a few examples of such questions that are used to label the detection hypotheses and to update the classifiers with them. Note that the classifiers are computed only with difficult samples which require human assistance (i.e, active learning). In the experiments section, we will show that this strategy improves the classification performance using less human annotations.

In order to make the human-robot interaction efficient and dynamic, the robot has been programmed with behaviors that avoid having large latency times (Table 1), specially when the human does not know exactly how to proceed. Strategies for approaching the person in a safe and social manner, or attracting people's attention have been designed for this purpose [14, 19, 52, 56].

3.2. The Online Classifier

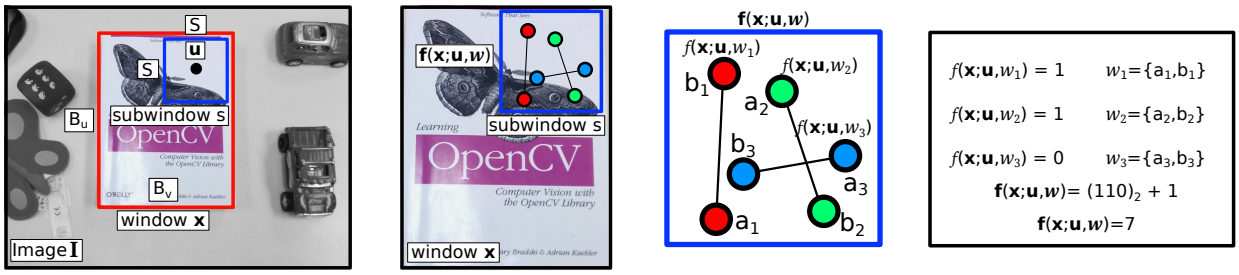
The proposed approach performs object detection by scanning a fixed-size sliding window over an input image I, observe Fig. 2-d. At every image location, the classifier is tested over an image sample $\mathbf{x}$ (local image window defined by the object size $B_u \times B_v$) and returns the probability that such window contains a particular object instance (Fig. 2-e,f). The size of the object is defined at the time of selecting the object of interest by the user using the computer mouse. This scanning procedure is carried out over the entire image, and once it is finished non-maximal neighborhood suppression is applied to remove multiple overlapped detections. Observe that Fig. 2-i shows just a single detection (green bounding box). Additionally, the scanning process is repeated for different window sizes so as to deal with scale changes.

Next, we describe each of the main ingredients of the online classifier used for object detection: the computation of random ferns on pixel intensities, the shared pool of ferns parameters to reduce the computational complexity of the method, the building of the classifier and the process to update this classifier with new samples.

3.2.1. Random Ferns

We compute object classifiers using a particular version of the extremely randomized trees [12, 37, 38], which are the so-called random ferns [28, 31, 37, 52]. Specifically, random ferns consist of sets of random and simple binary features computed over image pixel intensities [37]. Fig. 2-b shows for instance three different random ferns, each with three binary features (i.e., pairs of colored dots).

More formally, each feature f corresponds to a Boolean comparison of intensity image values at two pixel locations a and b within a square subwindow s of size $S \times S$ where the fern is computed. Refer to Fig. 3 for a descriptive example showing the computation of



(a) Window x (b) Subwindow s (c) Random fern (d) Fern output

Figure 3: Computation of a single random fern containing three binary features. (a) Scanning window $\mathbf{x}$ in the image I and location and size of subwindow s. (b) Subwindow s where the fern $\mathbf{f}(\mathbf{x};\mathbf{u},\omega)$ is subsequently evaluated. (c) Fern features (binary comparisons) within the subwindow s. (d) Fern response determined by the co-occurence of binary features outputs.

a random fern and its binary features. Then, a binary feature can be written as:

$$f(\mathbf{x};\mathbf{u},\omega) = \mathbb{I}(\mathbf{x}(a) > \mathbf{x}(b)) \quad (1)$$

where $\mathbb{I}(e)$ is the indicator function[1], $\mathbf{x}$ is the image sample or window, $\mathbf{u}$ is the center position of the subwindow s inside the image window, $\omega = \{a, b\}$ are two randomly chosen pixel locations, and $\mathbf{x}(a)$ indicates the intensity-pixel value at location a.

Following the same spirit of the random ferns [37], we define a fern $\mathbf{f}$ as the combination of M different binary features in the image subwindow s. This can be formulated as,

$$\mathbf{f}(\mathbf{x};\mathbf{u},\omega) = [f(\mathbf{x};\mathbf{u},\omega_1), f(\mathbf{x};\mathbf{u},\omega_2), \ldots, f(\mathbf{x};\mathbf{u},\omega_M)] \quad (2)$$

where $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_M\}$ is the set of parameters (pixel locations) for M different binary features.

The fern output $\mathbf{f}(\mathbf{x};\mathbf{u},\omega)$ is an M-dimensional binary vector, which at the implementation level, is represented by an integer value $z \in \{1, \ldots, 2^M\}$. Fig. 3-c,d shows an example of how a fern is computed on a subwindow s centered at $\mathbf{u}$. In this case, $M = 3$ intensity-based pixel comparisons are considered, with individual outputs 1,1,0. The combination of these Boolean features responses determines the fern output $z = (110)_2 + 1 = 7$.

3.2.2. Shared Ferns Parameters

In order to compute efficiently multiple online object classifiers from scratch, we propose to use recursively a small set of fern features among all classifiers. By doing this, the computation of the fern features, which is the most computation costly part of the algorithm, is shared by all classifiers. This provides a remarkable speed up compared to when we train each classifier with a different subset of ferns, while classification rates are shown to remain high [54, 51] .

To this end, a small set of R random ferns parameters, $\Omega = \{\omega_1, \ldots, \omega_R\}$, is computed in advance, such that each object classifier can then be computed as a combination of ferns evaluated at different image locations but sharing the same parameters. Again in Fig. 2-c, we show an example where two different object classifiers are trained for two object instances. Note that each classifier is computed using five random ferns with only $R = 3$ features parameters, but with a specific spatial distribution and ferns probabilities which makes it discriminative for that object class.

Since both classifiers use the same fern features parameters Ω, the feature computation is shared and done in advance to testing the object classifiers. This results in a $U \times V \times R$ lookup table with a dense sampling of all possible fern responses, being $U \times V$ the size of the input image I. Consequently, the sharing strategy makes the overall computational cost to be just a function of the number of ferns parameters R (a typical value is $R = 10$), and to be independent on the number of classifiers (K) and the amount of ferns in each classifier (J). 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 $\frac{KJ}{R}$-fold speed-up achieved by the sharing scheme [54]. With this, two classifiers with $J = 200$ ferns and $R = 10$ distinct features parameters, yields speed-ups of up to 40×.

3.2.3. Building the Classifier

The classifier $H_k(\mathbf{x})$ for an object instance k is built by random sampling with replacement of J random ferns, from the shared pool Ω, computed at multiple image locations. The response of this classifier $H_k(\mathbf{x})$ over the

[1]The indicator function is defined by: $\mathbb{I}(e) = 1$ if e is true, and 0 otherwise.

sample $\mathbf{x}$ is:

$$H_k(\mathbf{x}) = \begin{cases} +1 & \text{if } \text{conf}_k(\mathbf{x}) > \beta \\ -1 & \text{otherwise,} \end{cases} \quad (3)$$

where $\text{conf}_k(\mathbf{x})$ is the confidence of the classifier on predicting that $\mathbf{x}$ belongs to the object k, and β is a confidence threshold whose default value is 0.5. Thus, if the output of the classifier is $H(\mathbf{x}) = +1$, the sample $\mathbf{x}$ is considered as an object or positive sample. Otherwise, this sample is assigned to the background or negative class.

The confidence of the classifier is defined according to the following posterior:

$$\text{conf}_k(\mathbf{x}) = p(y = +1|\mathbf{F}_k(\mathbf{x}), \boldsymbol{\eta}_k), \quad (4)$$

where $\mathbf{F}_k(\mathbf{x}) = \{\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j)\}_{j=1}^{J}$, with $\boldsymbol{\omega}_j \in \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_R\}$ and $\mathbf{u}_j \in \{\mathbf{u}_1, \ldots, \mathbf{u}_L\}$, makes reference to the set of J ferns for the classifier k, $y = \{+1, -1\}$ indicates the class label, and $\boldsymbol{\eta}_k$ are parameters of the classifier. The set $\{\mathbf{u}_1, \ldots, \mathbf{u}_L\}$ corresponds to all possible 2D pixel coordinates within a window $\mathbf{x}$ where a fern can be tested. For each fern j, the binary features parameters $\boldsymbol{\omega}_j$ and fern location $\mathbf{u}_j$ are chosen at random and kept constant during the learning and run-time steps.

In turn, the posterior probability $p(y = +1|\mathbf{F}_k(\mathbf{x}), \boldsymbol{\eta}_k)$ is computed by combining the posterior of the J ferns:

$$p(y = +1|\mathbf{F}_k(\mathbf{x}), \boldsymbol{\eta}_k) = \frac{1}{J} \sum_{j=1}^{J} p(y = +1|\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j) = z, \eta_k^{j,z}), \quad (5)$$

where z is the fern output and $\eta_k^{j,z}$ is the probability that the sample $\mathbf{x}$ belongs to the positive class in the k-th classifier with output z in the fern j. Since these posterior probabilities follow a Bernoulli distribution, $p\left(y|\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j) = z, \eta_k^{j,z}\right) \sim \text{Ber}(y|\eta_k^{j,z})$, we can write that

$$p\left(y = +1|\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j) = z, \eta_k^{j,z}\right) = \eta_k^{j,z}. \quad (6)$$

The parameters of these distributions are computed through a Maximum Likelihood Estimate (MLE) over the input samples and their corresponding labels, provided by the human user during the interaction with the robot. That is,

$$\eta_k^{j,z} = \frac{N_{k,+1}^{j,z}}{N_{k,+1}^{j,z} + N_{k,-1}^{j,z}} \quad (7)$$

where $N_{k,+1}^{j,z}$ is the number of positive (object) samples with output z for fern j. Similarly, $N_{k,-1}^{j,z}$ corresponds to the number of negative samples for the fern j with output z in the classifier k.

Algorithm 1: Online Random Ferns

Input: Previous online classifier $H_k(\mathbf{x})$
Input: Input sample $(\mathbf{x}, y)$, with $y = \{+1, -1\}$
Output: Updated online classifier $H_k(\mathbf{x})$

1 **for** $r = 1, \ldots, R$ **do**
2 **for** $l = 1, \ldots, L$ **do**
3 Test the fern $\mathbf{f}(\mathbf{x}; \mathbf{u}_l, \boldsymbol{\omega}_r)$ for every location $\mathbf{u}_l$ of the input sample $\mathbf{x}$ to compute a list of fern outputs $\mathbf{z}$. (Eq. 2)
4 **for** $j = 1, \ldots, J$ **do**
5 Using the fern parameters $\mathbf{u}_j$ and $\boldsymbol{\omega}_j$, retrieve the fern output z from the previously computed outputs $\mathbf{z}$. (lines 1-3)
6 Update the estimate $\eta_k^{j,z}$ of the fern j using the fern output z and label y, (Eq. 7)
7 **if** $y = +1$ **then**
8 $N_{k,+1}^{j,z} = N_{k,+1}^{j,z} + 1$
9 **else**
10 $N_{k,-1}^{j,z} = N_{k,-1}^{j,z} + 1$
11 Assemble the online classifier $H_k(\mathbf{x}) = \frac{1}{J} \sum_{j=1}^{J} p(y = +1|\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j) = z, \eta_k^{j,z})$. (Eq. 3)

3.2.4. Updating the Classifier

During the learning stage, once a sample has been labeled by the human user, this sample is used to recompute the probabilities $\eta_k^{j,z}$ of Eq. 7, and update the online classifier. For instance, let us assume that a sample $\mathbf{x}$ is labeled as $y = +1$, and it activates the output z of the fern $\mathbf{f}(\mathbf{x}; \mathbf{u}_j, \boldsymbol{\omega}_j)$. We will then update the classifier by adding one unit to the z-th bin of the histogram of $N_{k,+1}^{j,z}$. This process is repeated for all J ferns of the classifier k. With these new distributions, we can recompute the priors $\eta_k^{j,z}$ and update the classifier.

The computation of the online classifier is summarized in Algorithm 1. Note that the first part of the algorithm (lines 1-3) corresponds to convolve the ferns, with the shared pool of fern features parameters Ω, on the input sample $\mathbf{x}$. This process is done in advance to updating the classifier, and hence, it reduces drastically the computational cost since the size of Ω is much lower than the number of ferns ($R << J$).

3.3. Interactive Learning

The online learning strategy to train a specific classifier k is shown again in Fig. 2-d. As mentioned before, the object detection is carried out using a sliding window approach [55], where the classifier $H_k(\mathbf{x})$ is tested at every image location and multiple scales over an input image I. At each location, the image sample $\mathbf{x}$ is evaluated on all J ferns of the classifier to obtain the confidence $\text{conf}_k(\mathbf{x})$ (Eq. 4). Subsequently, the class la-

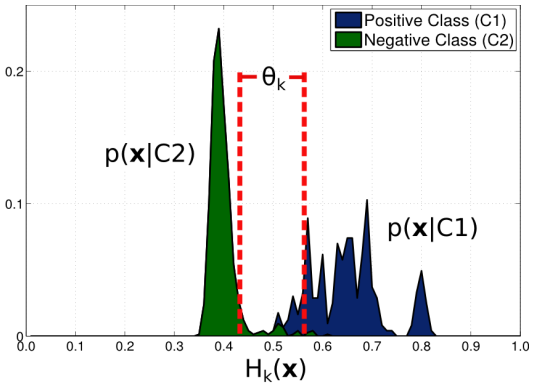


Figure 4: Uncertainty classification threshold θ_k for an online classifier $H_k(\mathbf{x})$. This threshold defines a region of high classification uncertainty between the positive and negative class distributions.

bel for this sample, $y = \{+1, -1\}$, is estimated according to the classifier response and the threshold β (Eq. 3).

To reduce the number of false positives and avoid drifting problems (produced when updating the classifier with erroneously labeled samples), frequent in non- and semi-supervised learning approaches [6, 21], we use an uncertainty-based active learning strategy [32, 46] that reduces gradually the amount of human assistance in combination with an adaptive uncertainty threshold. Active learning minimizes the risk of misclassification by updating the classifier only with samples which have been annotated/labeled by the user.

Therefore, in situations where the classifier is not certain about the class estimate y, because the confidence over the sample $\mathbf{x}$ is ambiguous (close to the threshold β), the system opts for requiring the human help so as annotate the true class of the sample. This request q can be written as:

$$q(\mathbf{x}) = \mathbb{I}(\beta + \theta_k/2 > \text{conf}_k(\mathbf{x}) > \beta - \theta_k/2), \qquad (8)$$

where θ_k corresponds to the uncertainty threshold for the classifier k. Fig. 4 displays a clarifying example. If $q(\mathbf{x})$ is true the system asks for human assistance. Otherwise, this sample is discarded and not used to update the classifier. Note that by doing this we are just feeding the classifier with labeled samples that are close to the decision boundary, improving thus, its discriminability power.

With the aim of adapting the human assistance in accordance to the performance of the classifier, we define an adaptive threshold that depends on the incremental classification rate over the requested samples. That is,

$$\theta_k = 1 - \xi\lambda_k, \qquad (9)$$

where ξ is a sensitivity parameter assigned by the user, and λ_k measures the performance of the classifier k. In turn, this performance rate can be computed by

$$\lambda_k = M_k^c / M_k^q \qquad (10)$$

being M_k^q and M_k^c the numbers of requested samples and correctly classified samples, respectively. A sample $\mathbf{x}$ is correctly classified when the class label y coming from the classifier agrees with the true class label given by the user.

Algorithm 2 shows the proposed interactive learning approach to compute online classifiers with small human supervision. Observe that each classifier is only computed with difficult samples falling close to the decision boundary β and which require human assistance.

Algorithm 2: Interactive Learning

Input: Online classifier $H_k(\mathbf{x})$
Input: Input sample $\mathbf{x}$
Input: Sensitivity parameter ξ
Input: Numbers of learning samples M_k^c and M_k^q
Input: Uncertainty classification threshold θ_k
Output: Updated classifier $H_k(\mathbf{x})$
Output: Updated numbers of learning samples M_k^c and M_k^q
Output: Updated uncertainty classification threshold θ_k

```
Test the classifier H_k on the input sample x to compute the
confidence of the classifier conf_k(x). (Eq. 4)
Estimate the sample label y using the response of the
classifier H_k(x)). (Eq. 3)
Assess if the input sample x requires human assistance
q(x) = I(β + θ_k/2 > conf_k(x) > β − θ_k/2). (Eq. 8)
if q(x) = 1 then
    Request human assistance to obtain the true sample
    label y* ∈ {+1, −1}. (Sec. 3.1)
    Update the online classifier H_k using the input sample x
    and label y*. (Algorithm 1)
    Update the number of requested samples M_k^q = M_k^q + 1
    Update the number of correctly classified samples
    M_k^c = M_k^c + I(y = y*)
    Update the classifier performance λ_k = M_k^c / M_k^q. (Eq. 10)
    Update the uncertainty threshold θ_k = 1 − ξλ_k. (Eq. 9)
```

4. Experimental Results

In this section the proposed approach is evaluated and analyzed using diverse synthetic and real experiments. Synthetic experiments are used to accurately assess the limits of the learning approach in terms of the potential number of classes it can handle or the amount of human assistance required by ours and alternative learning strategies. Real experiments in indoor and outdoor scenarios are used in order to demonstrate the applicability of the system in our robotic platform. The system is used to perform some particular robotics tasks such as detecting multiple objects in urban environments and recognizing faces during human-robot interaction.

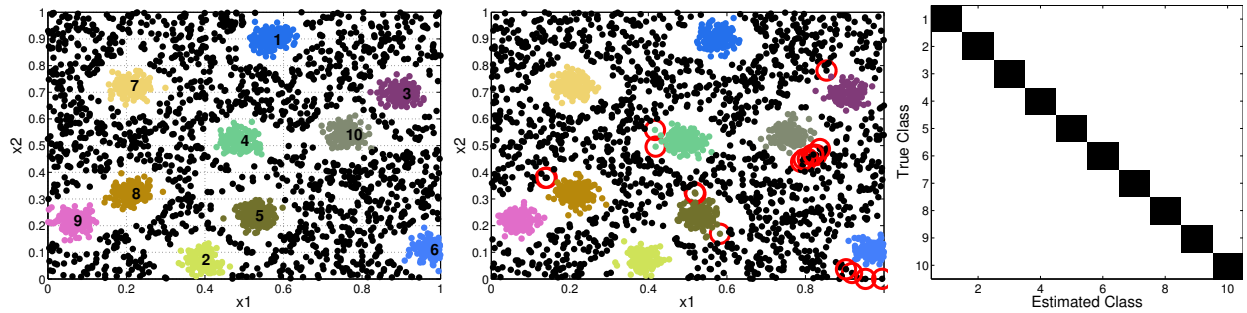


Figure 5: Performance of the proposed approach for the 2D classification problem. Left: 1500 positive and negative training samples used to compute the online classifiers. Colored samples represent positive samples, whereas black samples spread out over the feature space correspond to negative samples. Center: Test samples used to evaluate generalization of the online classifiers. Samples with red circles indicate that they have been misclassified by the proposed approach. Right: Confusion matrix showing the intra-class classification.

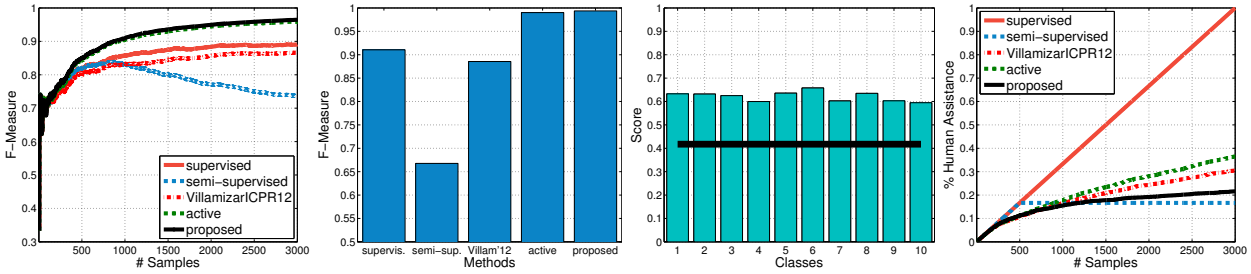


Figure 6: Classification results for the 2D problem. Left: Incremental classification rates for different learning approaches. Center·Left: Classification rates on the test samples. Center·Right: Average classification scores across classes. Right: Degree of human assistance.

4.1. Synthetic Data - 2D Classification Problem

We initially analyze the performance of the proposed online method on a synthetic 2D classification problem, which reveals the influence of certain parameters or different learning strategies on the classification results and on the number of samples that need to be manually annotated.

Fig. 5-left shows an example of a 2D classification problem where 10 positive classes (colored points) and one negative class (black points spread out over the feature space) are randomly and sequentially fed to the online learning system in order to compute the classifiers. In this particular scenario, the classifiers are computed using individual 2D decision stumps as binary features (Eq. 1). That is,

$$f(\mathbf{x};\omega) = \mathbb{I}(\mathbf{x}_i > \phi), \tag{11}$$

where $i \in \{1,2\}$ is the feature axis, $\phi \in (0,1)$ is threshold defining the space partition, and $\omega = \{i,\phi\}$ are the parameters of the binary feature.

The classification performance of the proposed approach is shown in Fig. 5-center where the classifiers computed using the training samples (left side) are evaluated on a set of test samples in order to measure the generalization capability. It can be seen that most samples are correctly classified and only a small fraction of them are misclassified, indicated in the figure through red circles. Correctly classified samples are depicted using the same color code as the training samples (left side) and without plotting red circles. Quantitatively speaking, the method obtains a F-measure[2] rate of 0.994 to distinguish positive samples from negative ones (two-class separability). Fig. 5-right shows the confusion matrix in specifically recognizing each of the 10 positive classes (using ground-truth sample labels). We see that the method achieves high classification rates both to separate the positive and negative classes and to correctly classify the positive subclasses.

In Fig. 6-left is shown the incremental classification performance of the method as the training samples are given sequentially to the online classifiers during the learning step. Here, five different learning alternatives are evaluated:

Supervised: The classifiers are trained with all samples and using human labels. That is, for each sample $\mathbf{x}$ the human user provides its class label y. They are used to train/update the classifiers.

[2]F-measure is the harmonic mean of precision and recall:

$$F = 2\frac{recall \times precision}{recall + precision}$$

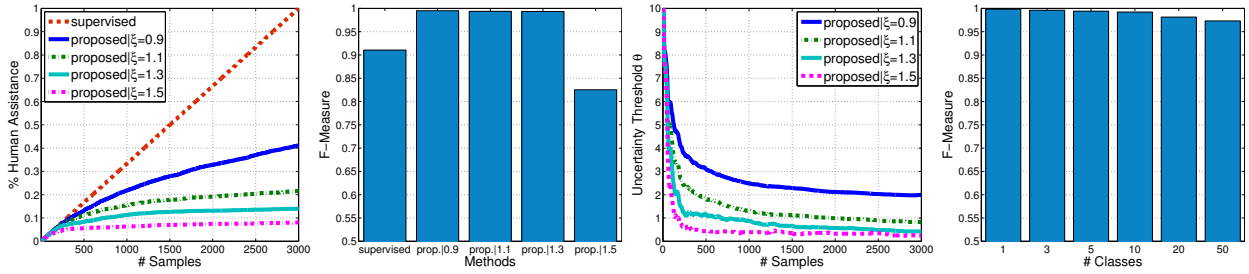


Figure 7: Classification results for different degrees of human intervention. Left: Percentages of human labels. Center·Left: Classification rates on the test samples. Center·Right: Adaptive uncertainty threholds. Right: Classification rates in terms of the number of classes.

Semi-supervised: The first n samples are labeled by the human (human labels), whereas the rest ones are labeled using the classifier confidence (machine label). The latter is computed for a sample $\mathbf{x}$ according to $y = H(\mathbf{x})$ (see Eq. 3). In this work we use a value of $n = 500$.

Active: The classifiers are only trained/updated in cases of high uncertainty in their predictions. The human resolves the ambiguity by providing the sample label. Here, we use a fixed uncertainty threshold of $\theta = 0.2$ and the assistance criterion is defined in Eq. 8. Hence, samples falling within the range $[\beta - \theta/2, \beta + \theta/2]$ are requested for human annotation.

Villamizar ICPR2012 [52]: This approach combines active and semi-supervised learning strategies. Active learning for uncertain samples and self-learning for certain samples. Similar to the previous case, active learning uses a fixed threshold ($\theta = 0.2$) and the assistance criterion (Eq. 8), while the self-learning estimates the sample label using the classifier output (Eq. 3).

Proposed [53]: The classifiers use active learning in combination with an adaptive uncertainty threshold θ defined in Eq. 9.

It can be observed that all learning approaches start with low classification scores, but then they begin to improve progressively as more samples are provided to the classifiers. However, note that at some point, the semi-supervised learning deteriorates the classifier performance. This is because the self-learning suffers from drifting problems, making the classifier to be constantly updated with erroneously labeled samples. Specifically, this classifier deteriorates after $n = 500$ samples, which is the number of human labels considered in this experiment. Conversely, our proposed method and the active learning obtain the best performance since the classifiers are computed with highly informative samples (uncertain samples) and human labels. This focuses the classifier mainly on the decision boundary and makes it more discriminative that using all training samples (supervised method).

Similarly, Fig. 6-center·left shows the final classification rates (F-measure) calculated on the set of test samples (refer to Fig. 5-center). We see again that the proposed method, together with the active learning, achieves the best classification performance and generalization capability. This is evidenced in Fig. 6-center·right where the average classification scores across the different classes are plotted. Notice that all scores are above the classification threshold β, whereas the average score for the negative class is indicated in the figure by the black thick line.

As regards to the amount of human intervention, Fig. 6-right displays the percentages of human labels during the learning step as a function of the number of incoming samples. The semi-supervised learning just uses labels for the first 500 samples. Note also that the supervised learning uses all human labels (represented by a diagonal line) to compute the classifiers, whereas the proposed method reduces considerably the amount of human assistance. More precisely, the classifiers are computed by only using 22% of the training samples. By contrast, the active learning continues requiring human assistance until reaching about 38% of the samples, while the learning method presented in [52] reports an annotation rate of 32%, since this method uses a fixed uncertainty threshold θ. This shows that the proposed adaptive uncertainty threshold reduces the human intervention without deteriorating the classification rates.

This behavior is observed in Fig. 7 where our approach is evaluated in terms of the adaptive uncertainty threshold. Fig. 7-left shows the human assistance percentages for four different values of the sensitivity parameter ξ. We observe how the number of required human annotations decreases as the sensitivity parameter gets larger until obtaining less than 10% of the training samples. However, this at the expense of an important reduction in the classification rates, see Fig. 7-center·left. In Fig. 7-center·right we can see the adaptive threshold values through the incoming samples. As a general trend, the threshold decreases rapidly as the classifiers get more confident in their predictions.

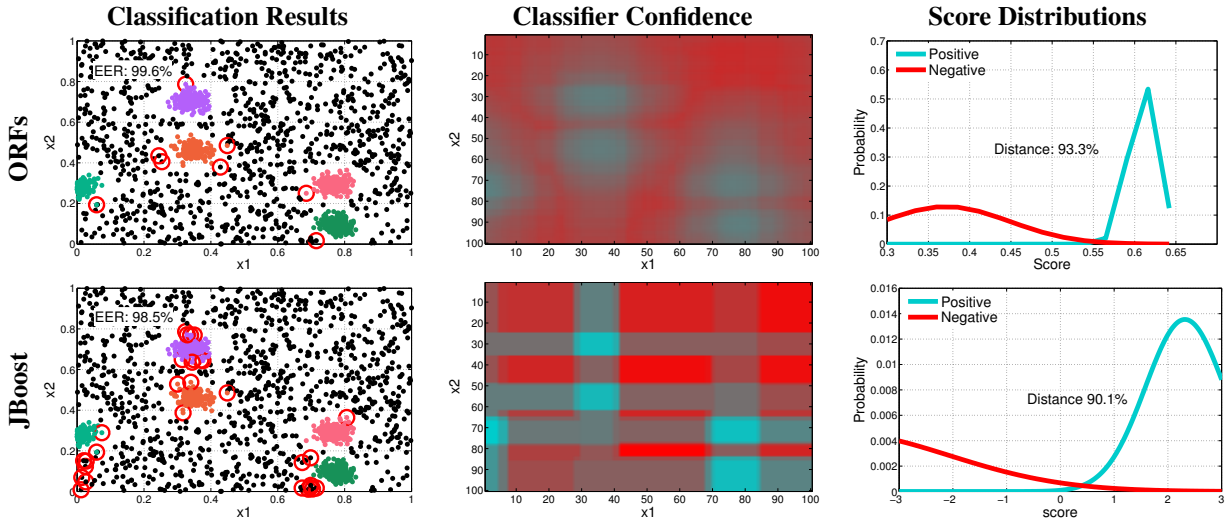


Figure 8: Classification results of the proposed method (ORFs) and the JointBoosting classifier (JBoost) using axis-aligned features. Left: Classification output of the classifiers over the test samples. Center: Confidence of the classifiers on the feature space. Right: Score class distributions.

2D Classification Performance													
	Equal Error Rate [%]			Hellinger Distance [%]			Training Time [sec.]			Run Times [msec.]			J/WLs
ORFs	97.9	97.8	96.1	92.2	87.2	79.6	0.41	0.50	0.62	0.05	0.06	0.07	10
	99.1	99.0	98.7	96.1	92.4	89.0	0.49	0.59	0.79	0.07	0.09	0.13	25
	99.5	99.5	99.3	96.7	93.6	91.1	0.60	0.80	1.00	0.12	0.16	0.22	50
JBoost	92.9	77.4	57.5	67.9	32.9	25.8	0.23	0.27	0.87	0.72	2.27	18.06	10
	98.9	96.2	81.4	93.0	83.0	57.1	0.32	0.47	1.94	0.83	2.35	18.08	20
	99.4	98.6	97.1	95.0	91.1	85.7	0.46	0.89	3.60	0.98	2.62	17.84	50
# Classes	3	5	8	3	5	8	3	5	8	3	5	8	

Figure 9: Classification performance of the proposed method (ORFs) against JointBoosting classifier (JBoost) for varying numbers of addressed classes, and weak learners (WLs) or random ferns (J). 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.

We plot in Fig. 7-right the classification rates on the test samples for varying numbers of classes. Note that increasing the number of classes in the learning phase, produces a small drop in the classification rates. This is because we are considering a large number of classes for such a small feature space (2D).

With regards to other state-of-the-art classifiers, the proposed approach is evaluated against the JointBoosting classifier (JBoost), introduced in [50], using the current 2D classification problem. That is, the classification of multiple positive classes with respect to a negative class (Fig. 8-left). For JBoost, we directly use an implementation publicly available for 2D classification problems[3]. We compare against JBoost since it is an efficient and multi-class classifier that shares weak learners across different classes, reducing thus, the overall number of features.

In Fig. 9 we show the classification performance of our approach (ORFs) and JBoost in the sets of test samples. We base our analysis on two metrics: the Equal Error Rate (EER) on the precision-recall curve[4]; and the Hellinger distance[5] 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 training classes, weak learners (WLs) for JBoost, and random ferns (J) for the proposed method. Note that the method we propose achieves remarkable classification rates (EER) and larger separability between the positive and negative classes. This is especially outstanding for large values of training classes and small numbers of

[3] `http://web.mit.edu/torralba/www/`

[4] The equal error rate is the point in the precision-recall curve where precision=recall.

[5] 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 $k_3 = (\mu_P - \mu_Q)^2$.

Figure 10: Detection results for a single object. Top Row: Detection output of the proposed method. Bottom Row: Output of the approach proposed in [52]. Green rectangles indicate object hypotheses whereas red ones are background hypotheses labeled by the user during the assistance.

random ferns. In general, the proposed approach outperforms to JBoost while keeps high efficiency. We see that the method is trained in less that one second (using 1000 positive and negative samples) and that it can be tested in the order of microseconds per test sample. This contrasts to JBoost which is very efficient, but only for small amounts of classes. Additionally, JBoost is an offline and fully supervised classifier that requires knowing the number of classes in advance so as to train all classes together.

Finally, Fig. 8 shows the classification response of ORFs and JBoost using five training classes and 50 weak learners/ferns. The left column in the figure depicts the classification output of both classifiers. We can see that our method produces a small rate of misclassified samples (indicated again with red circles). The center column shows the confidence of the classifiers on the entire feature space, where bright areas correspond to high classification scores. Here, we attribute the color cyan to the positive class, whereas red regions make reference to regions with high probability to belong to the negative class. Notice that both methods show certain correspondence with the samples in the left column. Fig. 8-right plots the score class distributions and the Hellinger distance between the positive and negative samples. Both methods obtain large distance values between classes.

4.2. Experiments with Real Data

The proposed approach is evaluated in this section over different object recognition scenarios and compared against some standard works devoted to object detection and tracking in video sequences[6].

Unless otherwise stated, all classifiers are computed using the same parameters. The number of ferns used to build each classifier has been set to $J = 500$, while the amount of shared ferns parameters has been set to $R = 10$. We have utilized $M = 9$ binary features to compute each random fern. Moreover, the size of the subwindow s is $S = 8$, whereas all objects are normalized by height to 30 pixels, but maintaining the aspect ratio of objects. By default, we have set a conservative sensitiviy parameter of $\xi = 0.95$.

In order to increase the robustness of the classifier and speed up the learning of object models with varying appearance, we update the classifiers not only with the object hypotheses (image windows $\mathbf{x}$) given by itself, and annotated then by the human user, but that additionally we create a set of new positive and negative samples that enlarge the training data. This is done by applying small distortions like image shifts on those detection windows. For this work, we consider 10 new samples per window.

Incremental Object Learning. This first experiment consists in learning and detecting one single object in a cluttered scene. Fig. 10 shows some examples. Particularly, this experiment has over 2,000 images containing the object (beer bottle) at diverse locations and viewpoints. The user initializes the classifier with the first frame. The top row of Fig. 10 corresponds to the output of the proposed method (for $\xi = 1.1$). This is compared in the bottom row with the detection results obtained by the approach [52], that combines active learning with self-learning. Results show that both methods are able to learn and detect the object through the whole sequence. This is indicated via the green rectangles around the object. Red rectangles are detection hypotheses that the user has manually labeled as incorrect during the interaction. Yet, despite the good performance of both methods, the proposed approach has the benefit that the amount of human assistance is significantly reduced. This is shown in Fig. 12-left, where the percentages of human assistance are displayed together with the active learning approach. Our method obtains the lowest rate of human assistance while correctly detects the object in all frames.

[6]The code for the proposed method is available at `http://www.iri.upc.edu/people/mvillami/code.html`.

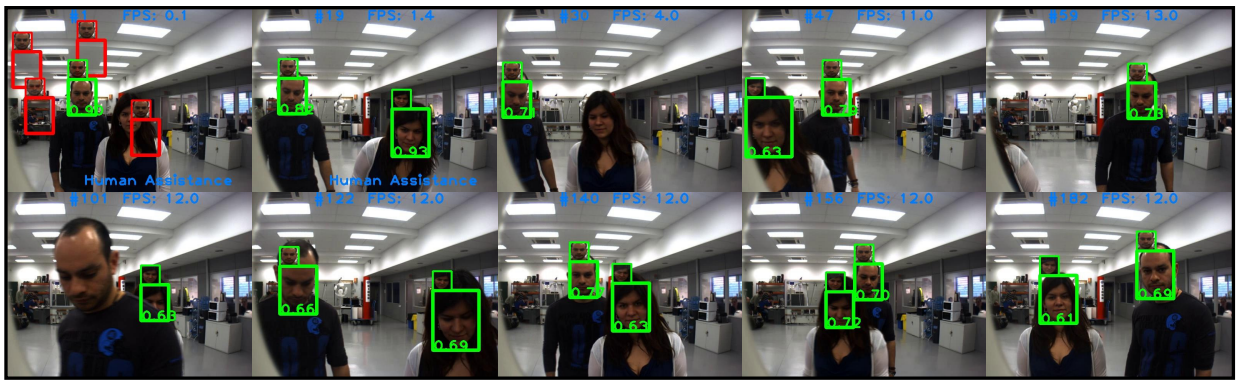


Figure 11: Online face recognition. The presented method performs online learning and detection of faces via human-robot interaction.

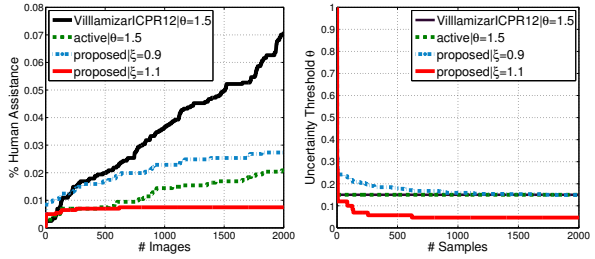


Figure 12: Performance comparison of different learning approaches in terms of the amount of human assistance (left) and the uncertainty threshold (right).

Figure 13: Online learning and detection of objects while the robot navigates in urban areas.

Additionally, Fig. 12-right shows the uncertainty thresholds for the aforementioned learning approaches. Observe that our method gradually reduces the uncertainty threshold during the learning step while the active learning and [52] maintain a constant value ($\theta = 0.15$).

Interactive Face Recognition. In this second experiment, the proposed method is focused on the detection and identification of faces for two persons interacting with our robotic platform. Fig. 11 shows some detection results. Similar to the previous experiment, the classifiers are interactively trained using human assistance. In this case, the method learns and detects two people simultaneously while they interact with the robot. This contrasts to [52] where the classifiers are independently computed and one at a time. The displayed sequence snapshots show that our method can effectively and simultaneously learn multiple faces and detect them in the subsequent frames. Besides, the approach retrieves people identity since each classifier is specialized in one person. This issue is shown by the small images beside the detection boxes. Observe that in the first video frame (at top-left of the figure), there exist several incorrect hypotheses (red boxes) that require the human intervention to label them as false positives. This occurs because the classifier is initialized in this frame and thus its confidence is very low. This issue is removed in future frames as the confidence of the classifier gets larger.

Furthermore, the method runs in real time, except for the assistance periods, where the system stops the video capture waiting for human answer. If after a while the robot does not receive response, the system resumes the video capture but without updating the classifier. Technically speaking, the method runs on about 13 fps using a C++ implementation[6]. It is noteworthy that the proposed method does not use neither temporal information nor tracking techniques that help to speed up the object detection.

Object Recognition in Urban Settings. In our third experiment, the method is evaluated for learning and detecting multiple objects in urban scenarios. Fig. 13 shows two example images where the robot is moving in urban areas. During the tour the user teaches the robot some objects of interest in order to be learned and detected.

Fig. 14 depicts some sample images showing the performance of the classifiers and the ability of the proposed method to learn several urban objects in real time and interactively when the robot navigates within the environment. We can see that objects like cars, doors and buildings are easily learned and recognized by the system. They are represented in the images by colored

Figure 14: Learning and detection results of the proposed method in urban scenarios. The method is able to compute efficiently multiple classifiers, each one specialized in an object (colored boxes). Black boxes are detection hypotheses assigned to the background class by the human user, while red circles denote false positive detections committed by the classifiers.

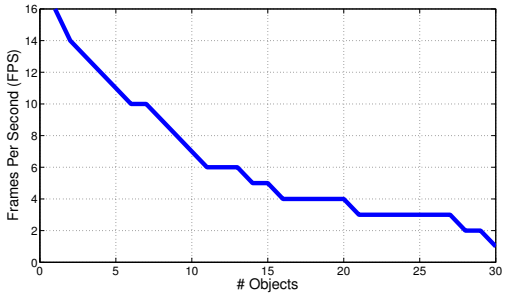


Figure 15: Computational efficiency of the proposed approach according to the number of objects.

boxes and small patches at the right side. The images also contain the percentage of human supervision (indicated by the red hand), the uncertainty threshold θ, and the incremental performance of the classifier λ. Notice that the proposed method is capable of learning and detecting multiple objects, with a few false positives (indicated by red circles), and scarce human assistance.

With regards to the computational time, Fig. 15 plots the running times (in frames per second) of the proposed method as a function of the number of object classifiers. Note that the computational cost increases as the number of objects gets larger. However, learning and detect 20 objects at four frames per second is a remarkable and promising result, especially for current robotic tasks involving online learning and real-time performance. The reason that the method slows down its recognition speed, in spite that all classifiers share the same features, is because all classifiers must be evaluated using the sliding window approach. This process depends of the number of classifiers and involves the combination of fern probabilities per each classifier (see Eq. 4).

Object Tracking. In this section, the approach is evaluated on public datasets focused on tracking objects under varying conditions such as illumination changes, occlusions and rotations. The objective is to compare the method with other approaches, even knowing that our method is not specifically designed for object tracking. Our method for example does not use temporal information to reduce the search space and to remove false positives [21, 28]. However, we consider that object tracking is a good scenario to show the importance of the human intervention to cope with drifting problems.

First, we evaluate our approach in the BoBot object dataset [30] which contains various daily objects such as cups and toys. Here, we consider just five objects

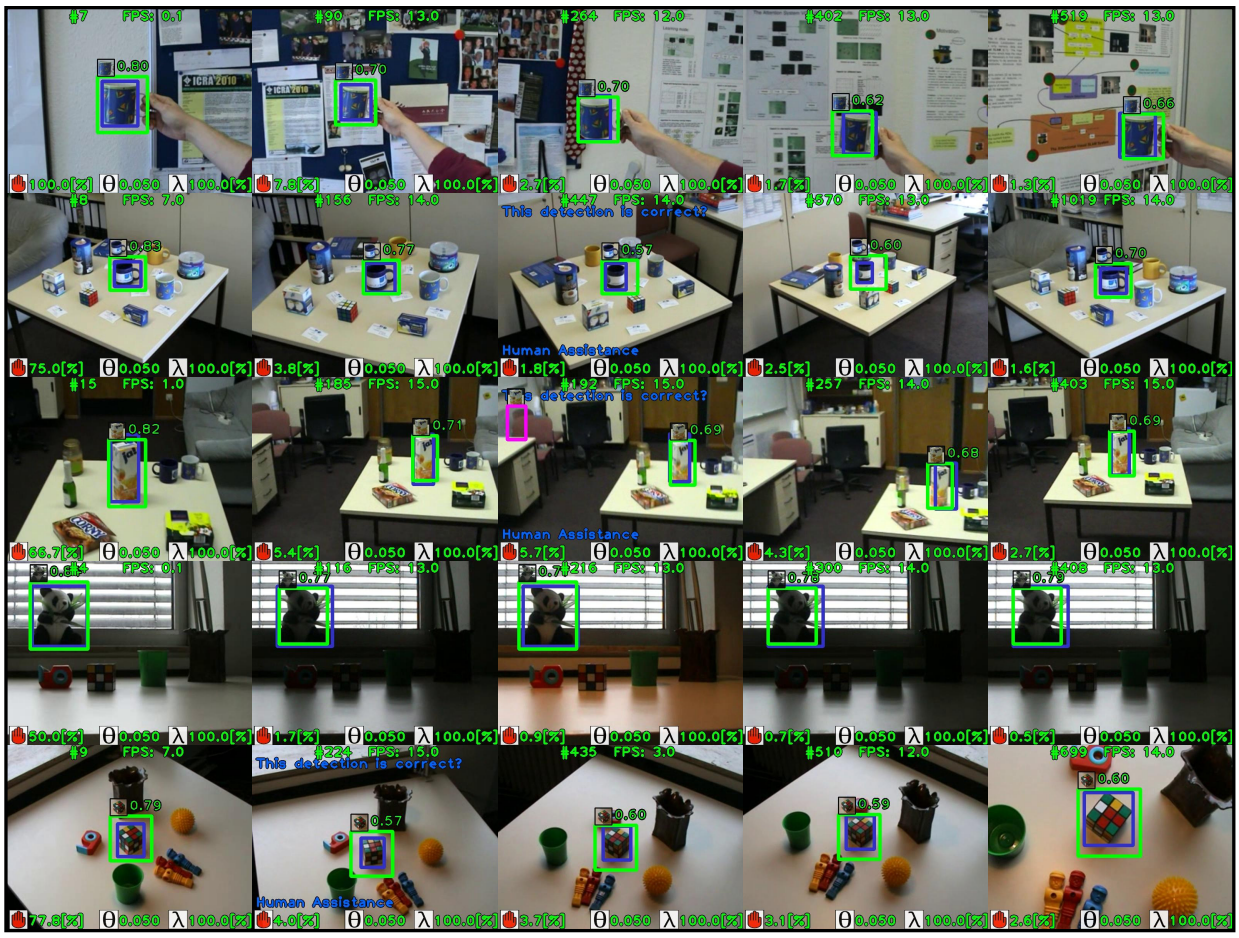


Figure 16: Online learning and detection results of the proposed method over the BoBot dataset.

Object Detection Rates					
Object	**EER**	**TPs**	**FPs**	**Assist.**	**# Fram.**
Cup 1	100	629	0	1.27	629
Cup 2	85.5	872	148	1.56	1020
Juice	100	404	0	2.72	404
Panda	100	412	0	0.48	412
Rubik.	89.3	642	74	2.45	716

Table 2: Object detection results provided by our method in the BoBoT object dataset [30]. The method is evaluated in terms of EER rates, amounts of true positives (TPs) and false positives (TPs), and percentages of human assistance.

from the entire dataset: two cups, a rubikscube, a stuffed panda and a juice box. Refer to Fig. 16 to observe these objects under varying imagining conditions.

The detection results of our method in this dataset are summarized in Table 2, where we can see that the method performs remarkably well. In most cases, the method achives a detection rate of 100% using the Equal Error Rate (EER) over the precision-recall plot. The table also includes the amounts of true positives (TPs) and false positives (FPs), the number of frames, and the percentages of human assistance in each case. Notice that these values are really small.

In the sequences of cup 2 and rubikscube, the method obtains lower detection rates because the classifiers are updated with samples including small portions of background. This occurs mainly because the object is seen from multiple views and the classifier must be adapted to new and unknown object appearance.

Fig. 16 shows some sample images with the output of the proposed method in each video sequence. Blue boxes correspond to the ground truth while green rectangles are the response of the method. Magenta rectangles make reference to hypotheses considered by the human as background. We see that the method is able to learn and detect the objects with a few human-labeled samples and that it is robust to cluttered backgrounds, camera viewpoint changes and lighting conditions (as

Figure 17: Online detection performance of the proposed method together other standard approaches on the PROST database [44]. The methods are tested for the *box* (top row) and *lemming* (bottom row) object sequences. The response of our method is indicated by black boxes. The output of the trackers PROST [44], MILTrack [7], FragTrack[2], ORF[43], and GRAD[29] are shown by red, blue, green, yellow and magenta boxes, respectively.

Object Detection Accuracy				
	Box		**Lemming**	
Method	Score	Dist.	Score	Dist.
PROST[44]	90.6	13.0	70.5	25.1
MILTrack[7]	24.5	104.6	83.6	14.9
FragTrack[2]	61.4	57.4	54.9	82.8
ORF[43]	28.3	145.4	17.2	166.3
GRAD[29]	91.8	13.2	78.0	28.4
Proposed	95.3	15.29	88.1	28.31

Table 3: Accuracy of the proposed method on the PROST dataset [44], and comparison with standard tracking approaches.

shown in the stuffed panda sequence).

The proposed method has also been evaluated on the PROST dataset [44]. This dataset contains four different objects under difficult conditions such as occlusions, blurring, and 3D rotations. In this experiment, we evaluate the proposed approach on two object sequences: *box* and *lemming*. Some instances of these objects are shown in Fig. 17.

Table 3 shows the detection results of our method using the percentage of correct detections (PASCAL score) and the average euclidean distance between the detection boxes and the ground truth [44]. The table also provides an extensive comparison with various recognized approaches in the state of the art for tracking purposes. Note that the proposed method outperforms these works on both object sequences. This is because our method uses human assistance to cope with the drifting problem. As a result, the classifier continues learning the object appearance with correct hypotheses and using little human intervention. Specifically, the method needs human assistances of 3.7% and 9.4% for the *box* and *lemming* objects, respectively.

Fig. 17 illustrates the online detection performance of our method and conventional trackers in the literature (see Table 3). Observe that our method detects the objects in almost all frames (black boxes). The PROST [44] and GRAD [29] trackers also achieve high recognition rates on these sequence. However, these methods are devoted to tracking single objects while our approach is able to compute multiple object detectors on the fly. With respect to the other tracking methods, lower recognition rates are reported since they are prone to drifting, and thus, miss object instances in some frames. This is particularly evident for the *lemming* sequence that exhibits object rotations and occlusions.

People Recognition under Shadows. Here, the proposed approach is evaluated for people detection under difficult illumination conditions. More precisely, the method is tested on a video sequence with 348 images that include a person walking in an urban scenario with varying shadows and background. Fig. 18 shows some video frames. For this experiment, we use a normalized object height of 50 pixels.

For comparison purposes, we have evaluated the TLD tracker[7] on this sequence. This tracker has shown excellent results in recent years for online learning and detection [28]. The recall and precision rates of this tracker and the proposed method are indicated in the Table 4. We see that our method achieves perfect recognition rates ($\xi = 0.95$) but at the expense of a higher degree of human intervention. On the other hand, the TLD tracker provides small detection rates, especially in terms of recall, since the classifier is not able to adapt to the abrupt

[7]Code available at `http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html`.

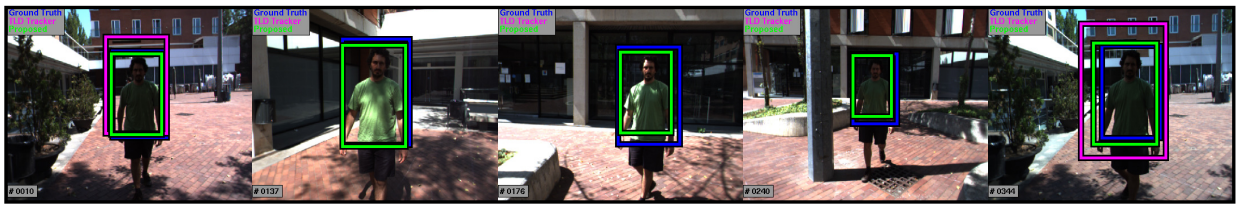


Figure 18: Detection results provided by the proposed approach (green boxes) for people recognition under varying shadows and illuminations. Blue boxes indicate the ground truth. Magenta boxes are the output of the TLD tracker [28] for comparison purposes.

People Detection Rates			
Method	Recall	Precision	Assist.
Prop. (ξ = 1.00)	0.68	0.98	0.9
Prop. (ξ = 0.95)	1.00	1.00	13.0
TLD tracker [28]	0.18	0.89	-

Table 4: Detection rates of the TLD tracker and the proposed approach for people detection under strong shadows.

changes in the image caused by shadows and the illumination conditions. This is observed in Fig. 18 where the tracker only detects the person at the beginning and final of the video sequence (when the loop is closed). By contrast, our approach is able to detect the person in most video frames thanks to the human assistance is used to remove false positives and update the classifier.

Online Learning with Occlusions. Finally, the method is evaluated in a standard video used for object tracking under occlusions [8]. This video has 886 images containing a woman under multiple occlusions with different degrees of difficulty. This experiment shows the adaptability of the method and the robustness to drifting. Fig. 19 shows some snapshots of this sequence and our detection results. At the top row, we see the output of the proposed approach (green boxes) along with the output of the MILTrack method [8], indicated by blue boxes. Here, we see that the latter method is prone to suffer of drifting, as shown in the fourth image. The bounding box has been displaced from the face's center. By contrast, our method is robust to drifting since the human assistance resolves the ambiguous cases. In the middle row of the figure, the confidence of the classifier is plotted. Note that the confidence changes according to the level of difficulty. In cases with occlusions, the confidence is low and requests human intervention (snapshots 1 and 3). At the bottom row, we plot the percentage of human supervision through the sequence. Again, we see that the method reduces gradually the overall amount of human intervention.

5. Conclusions

In this work, we have presented a novel approach for human robot interaction to interactively learn the appearance model of multiple objects in real time using scanty human supervision. The proposed method uses efficient and reliable random trees classifiers to compute object detectors on the fly and which are progressively refined with the human assistance. The proposed method also includes an uncertainty-based active learning strategy that reduces the amount of human intervention while it maintains high recognition rates. The method has been evaluated extensively in both synthetic and real-life different scenarios such as 2D classification, face recognition, and the learning and detection of contextual objects in urban settings using an autonomous mobile robot.

6. Acknowledgments

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness under projects ERA-Net Chistera project ViSen PCIN-2013-047, RobInstruct TIN2014-58178-R, ROBOTINT-COOP DPI2013-42458-P, and by the EU project AEROARMS H2020-ICT-2014-1-644271.

References

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *ICML*, 1998.

[2] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006.

[3] G. Alenyà, S. Foix, and C. Torras. Using tof and rgbd cameras for 3d robot perception and manipulation in human environments. *Intelligent Service Robotics*, pages 1–10, 2014.

[4] K. Ali and K. Saenko. Confidence-rated multiple instance boosting for object detection. In *CVPR*, pages 2433–2440, 2014.

[5] A. Amor-Martinez, A. Ruiz, F. Moreno-Noguer, and A. Sanfeliu. On-board real-time pose estimation for uavs using deformable visual contour registration. In *ICRA*, pages 2595–2601, 2014.

[6] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271, 2007.

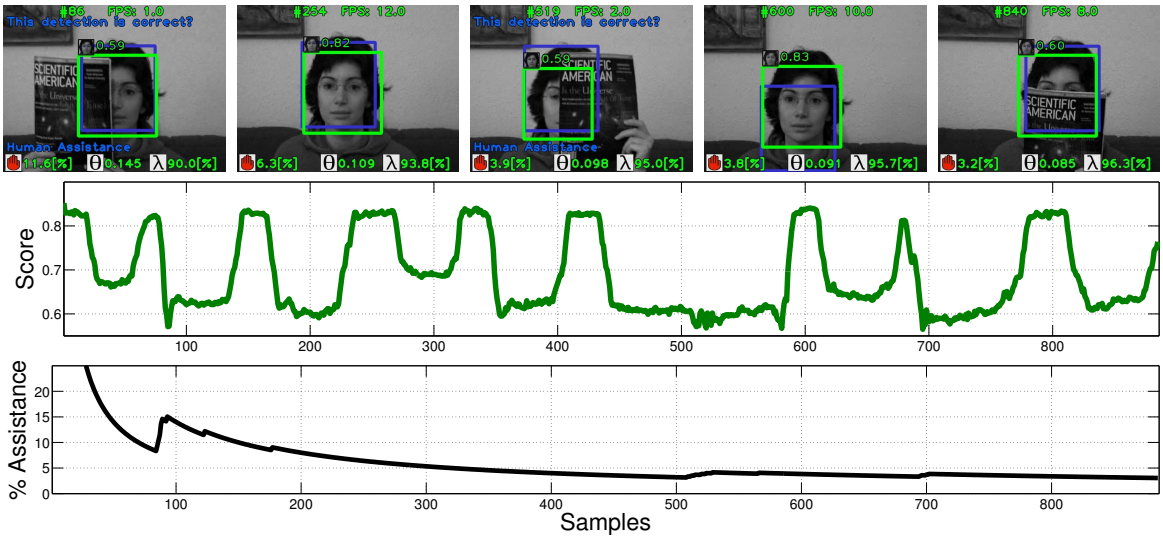


Figure 19: Online detection peformance of the proposed method under several and different occlusions.

[7] B. Babenko, M. H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, pages 983–990, 2009.

[8] B. Babenko, M. H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619–1632, 2011.

[9] M. S. Bartlett, G. Littlewort, I. Fasel, and J. R. Movellan. Real time face detection and facial expression recognition: Development and applications to human computer interaction. pages 53–53, 2003.

[10] N. Bellotto and H. Hu. Multisensor-based human detection and tracking for mobile service robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(1):167–181, 2009.

[11] A. Corominas, J.M. Mirats-Tur, and A. Sanfeliu. Efficient active global localization for mobile robots operating in large and cooperative environments. In *ICRA*, pages 2758–2763, 2008.

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

[13] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss. Real-time 3d hand gesture interaction with a robot for understanding directions from humans. In *RO-MAN*, pages 357–362, 2011.

[14] D. Feil-Seifer and M.J. Mataric. Defining socially assistive robotics. In *ICRA*, pages 465–468, 2005.

[15] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[16] G. Ferrer, A. Garrell, and A. Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *IROS*, pages 1688–1694, 2013.

[17] J. Gall, N. Razavi, and L. Van Gool. On-line adaption of class-specific codebooks for instance tracking. In *British Machine Vision Conference*, 2010.

[18] A. Garrell and A. Sanfeliu. Local optimization of cooperative robot movements for guiding and regrouping people in a guiding mission. In *IROS*, pages 3294–3299, 2010.

[19] A. Garrell, M. Villamizar, F. Moreno-Noguer, and A. Sanfeliu. Proactive behavior of an autonomous mobile robot for human-assisted learning. In *RO-MAN*, pages 107–113, 2013.

[20] M. Godec, C. Leistner, A. Saffari, and H. Bischof. On-line random naive bayes for tracking. In *ICPR*, 2010.

[21] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, 2006.

[22] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[23] H. M. Gross, A. Koenig, H. J. Boehme, and C. Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *IROS*, pages 256–262, 2002.

[24] D. Hall and P. Perona. Online, real-time tracking using a category-to-individual detector. In *ECCV*, 2014.

[25] S. Hinterstoisser, C. Cagniart, S. Holzer, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *CVPR*, 2011.

[26] A. Hornung, K. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *IROS*, 2010.

[27] M. Opper H.S. Seung and H. Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, 1992.

[28] Z. Kalal, J. Matas, and K. Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010.

[29] D. Klein and A. B. Cremers. Boosting scalable gradient features for adaptive real-time tracking. In *ICRA*, pages 4411–4416, 2011.

[30] Dominik A. Klein, Dirk Schulz, Simone Frintrop, and Armin B. Cremers. Adaptive real–time video–tracking for arbitrary objects. In *IROS*, 2010.

[31] E. Krupka, A. Vinnikov, B. Klein, A.B. Hillel, D. Freedman, and S. Stachniak. Discriminative ferns ensemble for hand pose recognition. In *CVPR*, pages 3670–3677, 2014.

[32] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[33] T. Malisiewicz, A. Gupta, and A.A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, pages 89–96, 2011.

[34] L. Merino, A. Gilbert, J. CapitÃ¡n, R. Bowden, J. Illingworth, and A. Ollero. Data fusion in ubiquitous networked robot systems for urban services. *annals of telecommunications-annales des télécommunications*, 67(7–8):355–375, 2012.

[35] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras. Dependent multiple cue integration for robust tracking. *PAMI*, 30(4):670–685, 2008.

[36] K. Nickel and R. Stiefelhagen. Visual recognition of pointing gestures for human–robot interaction. *Image and Vision Computing*, 25(12):1875–1884, 2007.

[37] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *PAMI*, 2010.

[38] D. Ernst P. Geurts and L. Wehenkel. Extremely randomized trees. *Machine learning, 63.1*, pages 3–42, 2006.

[39] J. Portmann, S. Lynen, M. Chli, and R. Siegwart. People detection and tracking from aerial thermal views. In *ICRA*, pages 1794–1800, 2014.

[40] M. Ragaglia, L. Bascetta, P. Rocco, and A.M. Zanchettin. Integration of perception, control and injury knowledge for safe human-robot interaction. In *ICRA*, pages 1196–1202, 2014.

[41] P. Rani, C. Liu, N. Sarkar, and E. Vanman. An empirical study of machine learning techniques for affect recognition in human–robot interaction. *Pattern Analysis and Applications*, 9(1):58–69, 2006.

[42] M. G. Rashed, R. Suzuki, A. Lam, Y. Kobayashi, and Y. Kuno. Toward museum guide robots proactively initiating interaction with humans. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 1–2, 2015.

[43] A. Saffari, C.Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV Workshops*, pages 1393–1400, 2009.

[44] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST Parallel Robust Online Simple Tracking. In *CVPR*, 2010.

[45] S. Schulter, C. Leistner, P. Wohlhart, P.M. Roth, and H. Bischof. Accurate object detection with joint classification-regression random forests. In *CVPR*, pages 923–930, 2014.

[46] B. Settles. Active learning literature survey. *University of Wisconsin, Madison 52*, pages 55–66, 2010.

[47] Y. Tamura, S. Yano, and H. Osumi. Visual attention model for manipulating human attention by a robot. In *ICRA*, pages 5307–5312, 2014.

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

[49] S. Thrun. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.

[50] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 19(5):854–869, 2007.

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

[52] M. Villamizar, A. Garrell, A. Sanfeliu, and F. Moreno-Noguer. Online human-assisted learning using random ferns. In *ICPR*, 2012.

[53] M. Villamizar, A. Garrell, A. Sanfeliu, and F. Moreno-Noguer. Modeling robot's world with minimal effort. In *ICRA*, 2015.

[54] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. Shared random ferns for efficient detection of multiple categories. In *ICPR*, 2010.

[55] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[56] D.M. Wilkes, R.T. Pack, A. Alford, and K. Kawamura. Hudl, a design philosophy for socially intelligent service robots. In *American Association for Artificial Intelligence Conference*, 1997.

[57] A. Yao, J. Gall, C. Leistner, and L. Van Gool. Interactive object detection. In *CVPR*, pages 3242–3249, 2012.