Randomized Clustering Forests for Image Classification

Frank Moosmann, *Student Member*, *IEEE*, Eric Nowak, *Student Member*, *IEEE*, and Frederic Jurie, *Member*, *IEEE Computer Society*

Abstract—This paper introduces three new contributions to the problems of image classification and image search. First, we propose a new image patch quantization algorithm. Other competitive approaches require a large code book and the sampling of many local regions for accurate image description, at the expense of a prohibitive processing time. We introduce *Extremely Randomized Clustering Forests*—ensembles of randomly created clustering trees—that are more accurate, much faster to train and test, and more robust to background clutter compared to state-of-the-art methods. Second, we propose an efficient image classification method that combines *ERC-Forests* and saliency maps very closely with image information sampling. For a given image, a classifier builds a saliency map online, which it uses for classification. We demonstrate speed and accuracy improvement in several state-of-the-art image classification tasks. Finally, we show that our *ERC-Forests* are used very successfully for learning distances between images of never-seen objects. Our algorithm learns the characteristic differences between local descriptors sampled from pairs of the "same" or "different" objects, quantizes these differences with *ERC-Forests*, and computes the similarity from this quantization. We show significant improvement over state-of-the-art competitive approaches.

Index Terms—Randomized trees, image classification, object recognition, similarity measure.

1 Introduction

As the number of images in personal collections, public data sets, or the Internet is becoming immense and ever growing, it becomes crucial to imagine effective and computationally efficient methods for organizing and searching images. Existing search engines are still restricted when we come to the problem of automatically extracting underlying semantics from images. This is why content-based image organization and retrieval has emerged as an important area in computer vision.

Having a good categorization process opens the possibility of filtering out images from irrelevant classes and therefore will enhance the performance of image retrieval. The machine learning framework can be very useful in this situation. It allows for learning a semantic category associated with images based on low-level visual features of the images. Recent machine learning techniques have demonstrated their capability of identifying image categories from image features, like [11], [15], [13], [14], [19], [51], [57], [61], to mention a few. Even if it has been shown that, to some extent, categories can be discovered without any supervision, the best results are obtained when knowledge about categories is provided. This knowledge can have different forms, but the two most frequent forms are either sets of images sharing common visual concepts [11], [15], [51], [57], [61] or sets of equivalence constraints (same/different) between pairs of images [13], [14], [19]. Both cases are considered in this paper.

Many of the most popular current methods for image classification work by selecting or sampling patches from the image, characterizing them by vectors of local visual descriptors, and coding (quantizing) the vectors by using a learned *visual dictionary*, that is, a process that assigns discrete labels to descriptors, with similar descriptors having a high probability of being assigned the same label. As in text categorization, the occurrences of each label ("visual word") are then counted to build a global histogram ("bag of words"), summarizing the image contents. The histogram is fed to a classifier to estimate the image's category label (see Fig. 1). Unlike text, visual "words" are not intrinsic entities and different quantization methods can lead to very different performances.

Computational efficiency is important because a typical image yields 10^3-10^4 local descriptors and data sets often contain thousands of images. In addition, many of the descriptors generally lie on the background, not the object being classified, so the coding method needs to be able to learn a discriminative labeling, despite considerable background "noise."

This paper precisely addresses the important issue of making visual vocabularies adapted to fast computation and high discrimination. We show that (small) ensembles of trees eliminate many of the disadvantages of single-tree-based coders without losing the speed advantage of trees (also see [1]). We also show that classification trees contain a lot of valuable information about locality in the descriptor space that is not apparent in the final class labels (predicted by the leaves). One can exploit this by training them for classification, then ignoring the class labels and using them

- *F. Moosmann is with the Institut für Mess- und Regelungstechnik, Universität Karlsruhe, Engler-Bunte-Ring 21, 76131 Karlsruhe, Germany. E-mail: moosmann @mrt.uka.de.*
- *E. Nowak is with the LEAR Group-INRIA, 655 avenue de l-Europe, 38334 Saint Ismier Cedex, France. E-mail: eric.nowak@inrialpes.fr.*
- *F. Jurie is with the LEAR Group and the University of Caen, 14032 CAEN cedex, France. E-mail: frederic.jurie@unicaen.fr.*

Manuscript received 16 Mar. 2007; revised 7 Sept. 2007; accepted 27 Sept, 2007; published online 6 Nov. 2007.
Recommended for acceptance by P. Fua.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-03-0170.
Digital Object Identifier no. 10.1109/TPAMI.2007.70822.

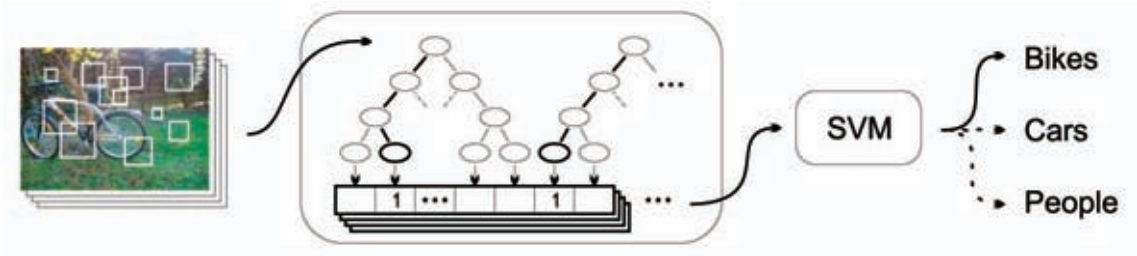

Fig. 1. Using ERC-Forests as visual code books in a bag-of-feature image classification.

as *clustering trees*, simple spatial partitioners that assign a distinct region label (visual word) to each leaf. The cluster (the leaf) contains all possible descriptors of the descriptor space that fall into that leaf. Combining these two ideas, we introduce *ERC-Forests*, ensembles of randomly created clustering trees. We show that they are robust to background clutter and that they provide much faster training and testing and more accurate results than conventional K-Means in several state-of-the-art image classification tasks.

Making visual dictionaries is not an end in itself and should be useful for achieving classification or search tasks. We demonstrate this by two further contributions.

The first one embeds the proposed *ERC-Forests* within an image classification approach. This approach can also localize objects in challenging conditions using visual attention techniques. It combines saliency maps very closely with the extraction of random subwindows for classification purposes. For a given image, a classifier builds a saliency map online and then uses it to classify the image. Thus, saliency is the core of our image classification algorithm and not an additional concept that one tries to fit into another method. Our results show that we can obtain state-of-the-art results on commonly used data sets while using only a little information and thus achieve high efficiency and short processing times.

The second demonstration uses the proposed *ERC-Forests* for learning distance between images. Specifically, we focus on learning a similarity measure to be used for comparing never-seen objects. The measure is learned from pairs of training images labeled *same* or *different*. This is far less informative than the commonly used individual image labels (for example, "car model X"), but it is cheaper to obtain. The proposed algorithm learns the characteristic differences between local descriptors sampled from pairs of *same* and *different* images. These differences are vector quantized by an ERC-Forest and the similarity measure is computed from the quantization. The proposed similarity measure is evaluated on four very different data sets and we show that it always outperforms the state-of-the-art competitive approaches.

The organization of the paper is as follows: First, the concept of ERC-Forests is presented and defined in Section 2. Then, the two other contributions, dynamic image classification and distance learning, are described in Sections 3 and 4, respectively.

2 Extremely Randomized Clustering Forests (ERC-Forests) as Visual Dictionaries

The overall goal of this section is to describe a fast and effective method for quantizing visual information. To make the description more concrete, we will consider the general context of image classification using the bag-of-words framework described in the previous section (see Fig. 1).

2.1 Quantization Methods

Several techniques have been explored to address this quantization process. The most common visual coding is based on K-Means vector quantization [11], [51]. Suggested alternatives include mean shift [28], but some recent approaches have also focused on building more discriminative code books [45], [57]. These methods give impressive results, but they are computationally expensive due to the cost of assigning visual descriptors to visual words during training and use. Indeed, the used nearest neighbor search remains hard to accelerate in high-dimensional descriptor spaces despite years of research on spatial data structures (for example, [49]). Nearest neighbor assignments may also be unstable: In high dimensions, concentration of measure [6] tends to ensure that there are many centers with similar distances to any given point. On the contrary, component-wise decision trees offer logarithmic-time coding, but individual trees can rarely compete with a good K-Means coding: Each path through the tree typically accesses only a few of the feature dimensions, so there is little scope for producing a consensus over many different dimensions. Nistér and Stewénius [41] introduced a tree coding based on hierarchical K-Means. This uses all components and gives a good compromise between speed and loss of accuracy.

Despite their popularity, K-Means codes are not the best compromise. It is difficult for a single data structure to capture the diversity and richness of high-dimensional descriptors. Therefore, we propose an ensemble approach. The theoretical and practical performance of ensemble *classifiers* is well documented [4]. Ensembles of random trees [9] seem particularly suitable for visual dictionaries due to their simplicity, speed, and performance [31]. Sufficiently diverse trees can be constructed using randomized data splits or samples [9]. *Extremely Randomized Trees* [20] take this further by randomizing both attribute choices and quantization thresholds, obtaining even better results. Compared to standard approaches such as C4.5, *ER-Tree* construction is fast, depends only weakly on the dimensionality, and requires relatively little memory.

Methods using classification trees, such as [31], [37], classify descriptors by majority voting over the tree-assigned class labels. Our method works differently after the trees are built. It uses the trees as a spatial partitioning method, not classifiers, assigning each leaf of each tree a distinct region label (visual word). Our approach is thus related to *clustering trees*, that is, decision trees whose leaves define a spatial partitioning or grouping [7], [34]. Although most visual code

books are built without external class labels, our method can use available labels to guide the tree construction. Ensemble methods, particularly forests of ER-Trees, again offer considerable performance advantages here.

Combining these ideas, we introduce *Extemely Randomized Clustering Forests* (ERC-Forests), which are ensembles of randomly created clustering trees. We show in the next section that they are robust to background clutter and that they provide much faster training and testing and more accurate results than conventional K-Means in several state-of-the-art image classification tasks.

2.2 Extremely Randomized Clustering Forests

2.2.1 Extremely Randomized Clustering Trees

As mentioned in the previous section, we developed ERC-Forests for building visual code books. ERC-Forests consist of randomized decision trees that predict class labels c from visual descriptor vectors of local regions $\mathbf{d} = (f_1, \ldots, f_D)$, where f_i, $i = 1, \ldots, D$, are elementary scalar features. Various local descriptors exist, with different degrees of geometric and photometric invariance, where the most discriminant ones tend to be high dimensional. In the experiments, we get these descriptor vectors by using either the popular SIFT descriptor [36] or a wavelet descriptor [52]. We train the trees using a labeled training set $L = \{(\mathbf{d}_n, c_n), n = 1, \ldots, N\}$, where we assume that all descriptors sampled from a given image share the same label c. However, we use the trees only for spatial coding and not for classification per se. During a query, each local descriptor $\mathbf{d}$ sampled from the query image traverses each tree from the root down to a leaf. Each tree assigns a unique leaf index to the visual descriptor and not the descriptor label c associated with the leaf during training. Thus, for each descriptor $\mathbf{d}$, the ERC-Forest returns a set of leaf indices, one for each tree, corresponding to the associated visual descriptor. The trees are built recursively in a top-down manner, as in [20] (see Algorithm 1). We start building each tree from the complete training set $L_0 = L$, which corresponds to the complete descriptor space $\mathcal{R}$.

Algorithm 1: tree growing algorithm

```
Tree(L_t): {Create a (sub)tree from labeled training set L_t}
if stopsplitting(L_t) = true then
   return createLeafNode(L_t)
else
   tries = 0
   repeat
      tries ← tries + 1
      select an attribute number i_t randomly
      select a threshold θ_t randomly
      split L_t according to test T : {f_{i_t} ≤ θ_t} and
          calculate score:
      L_l ← {f ∈ L_t | f_{i_t} < θ_t}
      L_r ← {f ∈ L_t | f_{i_t} ≥ θ_t}
      score ← Sc(L, T)
   until (score ≥ S_min) or (tries ≥ T_max)
   select i_t, θ_t that achieved highest score
   return createDecisionNode(i_t, θ_t, Tree(L_l), Tree(L_r))
end if
```

At each node t, two children l and r are created by choosing a Boolean test $\mathcal{T}_t$ that divides L_t into two disjoint subsets $L_t = L_l \cup L_r$, with $L_l \cap L_r = \emptyset$. Recursion continues with L_l and L_r until further subdivision is impossible: either all surviving training examples belong to the same class or all have identical values for all feature attributes f_{i_t}. In the algorithm, this criteria is checked by the function $stopsplitting()$. We use thresholds on elementary features as tests $\mathcal{T}_t = \{f_{i_t} \leq \theta_t\}$ for some feature index i_t and threshold θ_t. The tests are selected randomly as follows: A feature index i_t is chosen randomly, a threshold θ_t is sampled randomly from a normal distribution, and the resulting node is scored over the surviving points by using the Shannon entropy, as suggested in [20]:

$$Sc(L, \mathcal{T}) = \frac{2 \cdot I_{C,\mathcal{T}}(L)}{H_C(L) + H_{\mathcal{T}}(L)}. \tag{1}$$

H_C denotes the entropy of the class distribution in L:

$$H_C(L) = -\sum_{c \in \mathcal{C}} \frac{n_c}{n} \log_2 \frac{n_c}{n}, \tag{2}$$

where n is the size of the current set L and n_c is the number of descriptor vectors in L belonging to class c. It is maximal when all of the n_c are equal. Similarly, the split entropy $H_{\mathcal{T}}$ is defined for the test $\mathcal{T}$, which splits the data into two partitions:

$$H_T(L) = -\sum_{p=1}^{2} \frac{n_p}{n} \log_2 \frac{n_p}{n}. \tag{3}$$

The maximum is again reached when the two partitions have equal size. Based on the entropy of a given set, the impurity of a test can be calculated by the mutual information $I_{C,\mathcal{T}}$ of the split:

$$I_{C,\mathcal{T}}(L) = H_C(L) - \sum_{p=1}^{2} \frac{n_p}{n} H_C(L_p). \tag{4}$$

It is maximal when the uncertainty about the class is 0 in every subset L_i. In total, the score measure $Sc(C, \mathcal{T})$ favors trees that are as balanced as possible and, at the same time, separate the classes well.

The procedure of feature index selection and threshold selection is repeated until the score is higher than a fixed threshold $S_{\min}$ or until a fixed maximum number $T_{\max}$ of trials have been made. The test $\mathcal{T}_t$ that achieved the highest score is adopted and the recursion continues.

The parameters $(S_{\min}, T_{\max})$ control the strength and randomness of the generated trees. High values (for example $(1, D)$ for C4.5-like decision tree learning) produce highly discriminant trees with little diversity, while $S_{\min} = 0$ or $T_{\max} = 1$ produce completely random trees. The variance of the trees can be reduced afterward by pruning the trees back in a bottom up manner, recursively removing the node with the lowest score until either a specified threshold on the score or a specified number of leaves is reached.

As mentioned, we build the trees as classification trees but use them as clustering trees. One can also think about the test $\mathcal{T}$ as a hyperplane in the descriptor space $\mathcal{R}$. Hence, a given test $\mathcal{T}_t$ divides the corresponding space $\mathcal{R}_t$ into two disjoint regions, $\mathcal{R}_t = \mathcal{R}_l \cup \mathcal{R}_r$, with $\mathcal{R}_l \cap \mathcal{R}_r = \emptyset$. The split of the region, of course, then results in the split of the

training set L_t. As a tree is a hierarchical structure, one can look at it as a hierarchical clustering of the descriptor space. Thus, each leaf l_i corresponds to a hyperrectangle $\mathcal{R}_{l_i}$, with the union of all leaf hyperrectangles resulting in the complete descriptor space $\mathcal{R} = \bigcup_i \mathcal{R}_{l_i}$, with $\mathcal{R}_{l_i} \cap \mathcal{R}_{l_j} = \emptyset$ for any $\mathcal{R}_{l_i}$ and $\mathcal{R}_{l_j}$, $i \neq j$. As a clustering tree, the function $createLeafNode(L_t)$ in our case creates a leaf node and associates a unique leaf index with it and not a class label.

2.2.2 Extremely Randomized Clustering Forests

Compared to standard decision tree learning, the trees built using random decisions are larger and have higher variance. Class label variance can be reduced by voting over the ensemble of trees (for example, [37]), but here, instead of voting, we treat each leaf of each tree as a separate visual word. We also define the whole image descriptor as the set of all leaf indices reached by all of the local descriptors sampled from the image and passed through the trees, leaving the integration of votes to the final classifier (see Fig. 1).

Classifier forests are characterized by Breiman's bound on the asymptotic generalization error [9] $PE* \leq \rho(1 - s^2)/s^2$, where s measures the strength of the individual trees and ρ measures the correlation between them in terms of the raw margin. It would be interesting to optimize $S_{\min}$ and $T_{\max}$ to minimize the bound, but we have not yet tried this. Experimentally, the trees appear to be rather diverse while still remaining relatively strong, which, according to Breiman, should lead to good error bounds.

2.2.3 Computational Complexity

The worst-case complexity for building a tree is $O(T_{\max}Nk)$, where N is the size of the training set and k is the resulting number of clusters/leaf nodes. With adversarial data, the method cannot guarantee balanced trees, so it cannot do better than this. However, in our experiments on real data, we always obtained well-balanced trees at a practical complexity of around $O(T_{\max}N \log k)$. The dependence on data dimensionality D is hidden in the constant $T_{\max}$, which needs to be set large enough to filter out irrelevant feature dimensions, thus providing better coding and more balanced trees. A value of $T_{\max} \sim O(\sqrt{D})$ has been suggested [20], leading to a total complexity of $O(\sqrt{D}N \log k)$. In contrast, K-Means has a complexity of $O(DNk)$, which is more than 10^4 times larger for our 768-D wavelet descriptor, with $N = 20,000$ image patches and $k = 5,000$ clusters, not counting the number of iterations that K-Means has to perform. Our method is also faster in use and this is a useful property, given that reliable image classification requires large numbers of subwindows to be labeled [42], [57]. Labeling a descriptor with a balanced tree requires $O(\log k)$ operations, whereas k-means costs $O(kD)$.

3 Efficient Image Classification Using ERC-Forests and Adaptive Saliency Maps

3.1 Motivations

This section presents an image classification method embedding *ERC-Forests*. The key idea is to use *ERC-Forests* for both selecting informative patches and for assigning patches to visual words.

In this method (see Fig. 1), local features are extracted from the training images by sampling subwindows at random positions and scales and by coding them using a visual descriptor function (for example, the Haar Wavelet Transform [52] or SIFT [36]). An ERC-Forest is then built using the class labels and the local descriptors from the training images. To control the code book size, we grow the trees fully and then prune them back, as described in the previous section.

When calculating a bag-of-words-like histogram for an image, the trees transform each descriptor into a set of leaf node indices, with one element from each tree. Votes for each index are accumulated into a global histogram and are used for classification as in any other bag-of-features approach.

It is trivial that each of the four steps involved—patch selection, descriptor function, code book method, and final classifier—influences the resulting performance. We introduced an efficient coding method in the previous section, so we will now focus on the combined patch selection and image classification procedure.

3.2 Patch Selection Methods

Obviously, the optimal solution would be to process all possible patches in an image, but, due to computational efficiency, a subset has to be selected. Patches can be sampled on a regular grid [57], at random [37], or at selected interest points [36], [24], [29], [38]. The latter try selecting only relevant parts of the image based on some kind of importance criteria, which are usually domain independent. Two concepts can be distinguished here: visual saliency and visual search.

3.2.1 Visual Saliency

Visual saliency is a bottom-up process that selects salient regions in an image, also described by the term "pop-out." The pure bottom-up property makes them domain independent and easy to include in existing systems. They allow for summarizing images with a very small amount of local information, so it is very common to find them at the first stage of image categorization algorithms [11], [15], [30], [32]. Several kinds of salient features have been used in the literature. Keypoint-based detectors [46], [36] are generally based on cornerness criteria and can be defined to be scale, rotation, translation, and even affine invariant [38]. Entropy-based measures emphasize the information difference of regions [29], [47], [18] or the rarity of features in the feature space [23] more. Nonetheless, other approaches define salient features according to their classification capabilities [54], [53]. As pointed out by Serre et al. [48], general low specific features are not as efficient as specific learned features for recognizing objects. All of these definitions are derived from man-made criteria, although biological systems are the only full-functioning systems known. Nevertheless, biologically inspired systems such as [25] and [55] have, so far, never been able to compete with artificial systems.

3.2.2 Visual Search

Visual saliency and visual search are two related but different concepts. Although visual saliency relies on bottom-up processes only, visual search or visual attention is the process of selecting information based on saliency (bottom-up process) and on prior knowledge about objects

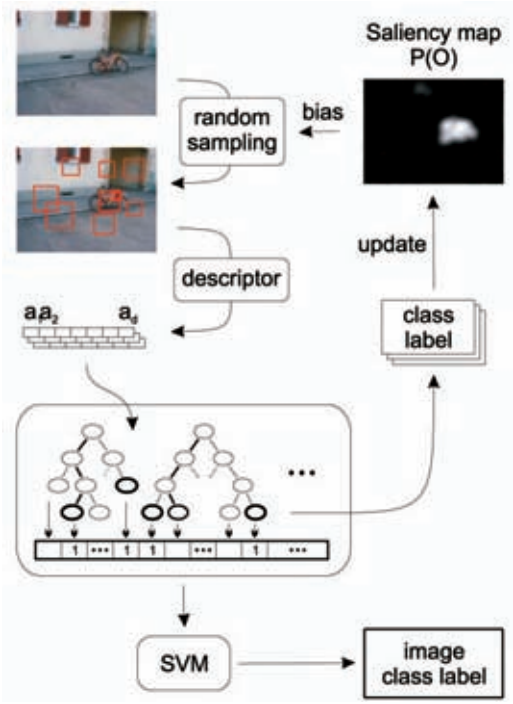

Fig. 2. Overall classification system using ERC-Forest and Adaptive Saliency Map.

(top-down process) [2], [60]. For the human visual system, such top-down processes seem to be an important component. Hence, it is not surprising that systems based on visual attention have already been proposed [40], [8], [2]. Despite the soundness of this concept, visual search is not used in any of the most noticeable object categorization systems presented before.

A comparison of sampling strategies has been carried out in [42], with the result that dense sampling outperforms sparse (salient-point-based) sampling with regard to classification performance. It was also shown that random sampling compares with grid-based sampling as the number of patches increases. Thus, summarizing the image with only a few saliency-selected features is not as good as extracting many features.

Despite these results, we believe that pure grid-based or random patch selection is not the optimal choice. As in biological systems, it should be possible to include visual search techniques that can reduce the amount of information extracted from an image without losing descriptiveness. Our work is based on random sampling, but we introduce a novel strategy to bias the random selection of patches toward domain-specific important regions of the image. We present *Adaptive Saliency Map* (ASM), an iterative algorithm based on the online estimation of the position of object parts. As shown in Fig. 2, random patches are sampled (bottom-up), which brings knowledge about the patch content, which is used back (top-down) to sample *more salient* random patches, etc. Thus, our algorithm converges to sample salient information only. Unlike most existing approaches, we define saliency as the set of attributes that distinguishes a concept (object category) the most from others; hence, it is application/domain dependent and should be learned to fit with a given context. In contrast to using the bottom-up process alone, ASM allows for reducing not only the amount of patches but also classification errors.

3.3 Saliency Map as a Probability of Finding Objects of Interest

Saliency maps contain information about where in the image interesting information can be found. These areas correspond to features considered as rare or informative, depending on the definition of saliency (see Section 3.2). High-saliency regions are likely to contain objects, while lower saliency regions are associated with background.

The classification system that we introduced extracts subwindows at random positions with random size. Sampling a subwindow at random corresponds to picking one point in the 3D scale space (position x, y of the window and its size) according to some probability density function. Similar to the idea in [59], we hence define this three-dimensional PDF as the saliency map, in which a point (subwindow) is defined as salient if it is likely to be classified as an object (as opposed to background). This PDF is defined by the probability $P(O|X)$ of having an object of interest O within the window $X = (x, y, s)$.

3.4 Building a Saliency Map by Random Sampling

To build this kind of saliency map, we take advantage of one property of ERC-Forests: Although they are used as local descriptor clusterers, they are built as local descriptor classifiers.[1] Hence, for each local descriptor, an ERC-Forest can return both region labels (leaf index) and class labels (the majority class label among local descriptors contained in the leaf during tree learning). Although the region labels are used as before in a bag-of-features representation to classify the image, we now also use the ERC-Forest to classify each descriptor for the saliency map computation.

Our method works in such a way that, first, an ERC-Forest is built from the set of training images. Afterward, this ERC-Forest is used to build the saliency map of any image. Subwindows sampled from an image are classified by the ERC Forest as a background (nonsalient) or an object (salient). To create this map efficiently, we propose the following:

1. Randomly sample points in the 3D scale space (that is, subwindows at any size and position).
2. Classify the windows by using the ERC-Forest.
3. Update the information of the saliency map based on the subwindow classes predicted by the ERC-Forest leaves. This will give us a number of sparse points whose saliency values are known.
4. From these points, propagate the saliency values to the whole 3D scale space to finally obtain $\hat{P}(O)$, which is the estimation of object locations.

Instead of finding an optimal way of propagating the certain points offline, we will instead focus on the online estimation of such a saliency map.

3.5 Active Image Classification with Adaptive Saliency Maps

So far, we have introduced a definition of saliency maps and explained how they can be used to bias the sampling of

1. The feature-threshold tests are optimized to *separate* the different classes.

subwindows. Now, we show how these maps can be built online and used at the same time by the classification process (see also Fig. 2).

To guide the sampling in such a way that enough windows are sampled on the object, we introduce a probability density function for the sampling process that combines the probability for an object to be present at a given position and scale with the probability of having already explored a given area of the image space. We use the following probabilities: $\hat{P}_n(O|X) = \hat{P}(O|X, S_{1:n}, Z_{1:n})$, the knowledge about having an object at position $X = (x, y, s)$, is the estimated saliency map that we have already talked about. It is dependent on the last n samplings $S_{1:n}$, with $S_i = (x, y, s)$, and their respective classification outcome $Z_{1:n}$, where $Z_i = +1$ for objects, and $Z_i = -1$ for the background.[2] The second "probability" $P_n(E|X) = P(E|X, S_{1:n})$ expresses the degree of need for exploration of a given position X. This *exploration map* is dependent on the last samplings $S_{1:n}$ only. In contrast to $\hat{P}_n(O|X)$, this is not an estimated quantity. In order to sample salient windows in unexplored regions, we now combine these two *probability maps* to form a PDF to draw random numbers from:

$$P(S_n = \mathbf{x}|S_{1:n-1}, Z_{1:n-1}) = \frac{\hat{P}_{n-1}(O|\mathbf{x})\, P_{n-1}(E|\mathbf{x})}{\sum_X \hat{P}_{n-1}(O|X)\, P_{n-1}(E|X)}. \quad (5)$$

Thus, we draw random numbers from the two *probability maps* multiplied with each other and normalized. Patches that have a high probability of being on an object and belonging to a region of the image not explored yet will thus be selected with a higher probability.

For the computation of $\hat{P}_n(O|X)$ and $P_n(E|X)$, we first introduce the gating function:

$$g_r(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} 1 & |x_1 - x_2| \le r \wedge |y_1 - y_2| \le r \wedge |s_1 - s_2| \le r \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

For the recursive calculation of the *exploration map* $P_n(E|X)$, we initialize the discrete PDF uniformly with $P_0(E|X) = 1$ as we need to explore the whole image. Every time we sample a window, the need for exploration in this region is reduced; thus, we set

$$P_n(E|X) = \max(0.1, P_{n-1}(E|X) - 0.05 \cdot g_r(X, S_n)). \quad (7)$$

This decreases the probabilities in a cubic neighborhood, but keeps them above zero. The radius of the neighborhood that was used during our tests was $r = 3\%$ of the image size.

The estimation of $\hat{P}_n(O|X)$ is similar to the need for exploration, but it also depends on the classification outcomes $Z_{1:n}$. We initialize the PDF uniformly, but, as we do not have any knowledge about the location of objects yet, we chose $\hat{P}_0(O|X) = 0.1$. For each window that we sample, we adjust this distribution according to the classification outcome before we sample the next window:

$$\hat{P}_n(O|X) = \min(1, \max(0.1, \hat{P}_{n-1}(O|X) + 0.05 \cdot Z_n \cdot g_r(X, S_n))). \quad (8)$$

If a window is classified as an object, this increases the PDF in the neighborhood of S_n, whereas it is decreased in the same neighborhood if Z_n holds the background label. The radius of the neighborhood used here was $r = 5\%$ of the image size. Incrementing and decrementing by a constant amount in a "cubic" region seems to be very simplified at first glance, but doing this several times and adding up the numbers, this produces smooth outcomes, as test results show.

3.6 Final Classifier

For being classified, an image is represented by a histogram of visual words, the visual words being the leaves of the *ERC-Forest* reached by all descriptors sampled from that image. Any classifier can be used. Within this work, we stick to the popular decision of using a Support Vector Machine (SVM) with linear kernel. The *histogram space* is populated sparsely enough to be (almost) linearly separable, which is why a separation can be performed very efficiently by an SVM.

3.7 Experiments on Image Classification Using Adaptive Saliency Maps and ERC-Forests

We present detailed results on the GRAZ-02 test set [44], which is available at http://www.emt.tugraz.at/~pinz/data/. Similar conclusions hold for two other sets that we tested, so we comment only briefly on these. GRAZ-02 contains three object categories—bicycles (B), cars (C), and people (P)—and negatives (N, which means that none of B, C, and P are present). It is challenging in the sense that illumination is highly variable and the objects appear at a wide range of different perspectives and scales and are sometimes partially hidden. It is also neutral with respect to the background, so it is not possible to reliably detect objects based on context alone. For example images, see Fig. 4. We tested individual object categories versus negatives N according to the test protocol in [44]. For *Setting 1* tests, we trained on the whole image, as in [44], while, for *Setting 2*, we used the segmentation masks provided with the images to train on the objects alone without background. We report results for the two hardest categories, that is, bikes and cars.

We measure performance with ROC curves and classification rates at equal error rate (EER). The method is randomized, so we report means and variances over 10 learning runs. We use $S_{\min} = 0.5$, but the exact value is not critical. In contrast, $T_{\max}$ has a significant influence on performance. We chose $T_{\max} = 50$ as a good trade-off between speed and accuracy.

Unless otherwise stated, 20,000 features (67 per image) were used to learn 1,000 spatial bins per tree for five trees and 8,000 patches were sampled per image to build the resulting 5,000-D histograms. The histograms are binarized using trivial thresholding at count 1 before being fed to the global linear SVM image classifier. We also tried histograms normalized to sum up to 1 and histograms thresholded by maximizing the mutual information of each dimension, but neither yielded better results for ERC-Forests.

We tested various visual descriptors. The best choice turns out to depend on the database. Our color descriptor uses raw

2. This does not restrict the problem to binary classification. The ERC-Forest can still be trained for a multiclass problem and the SVM can do multiclass separation afterward.

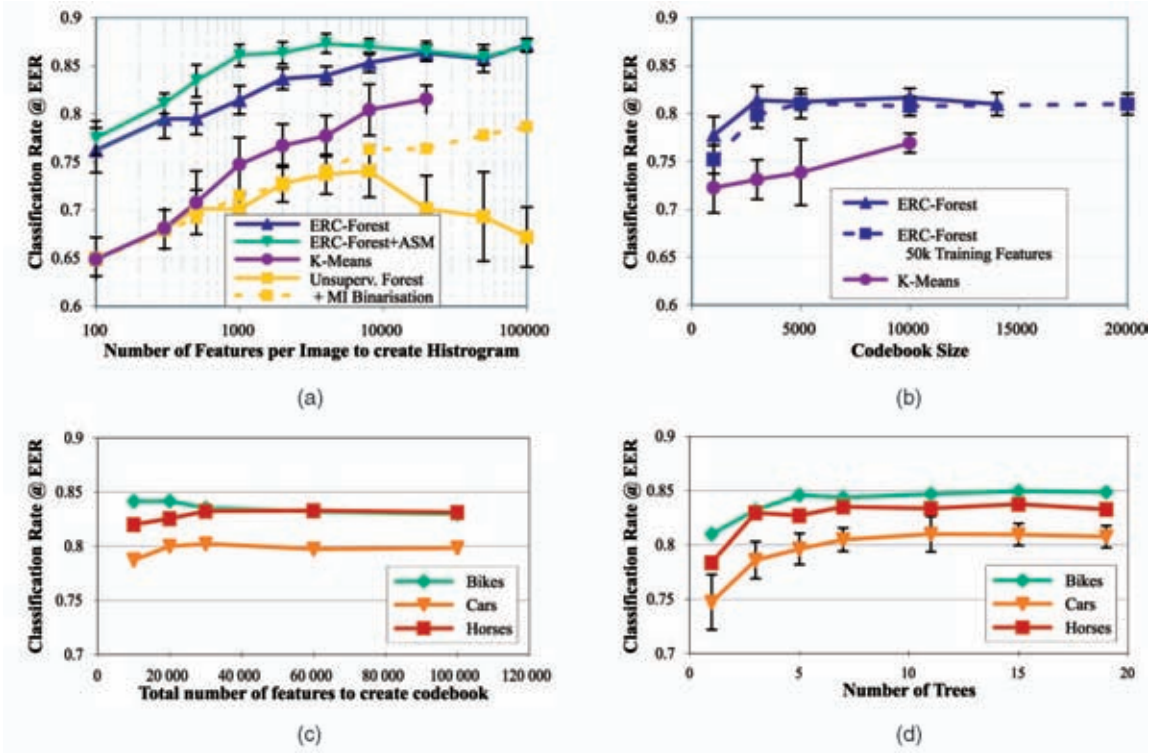

Fig. 3. Evaluation of parameters for the GRAZ02 data set in Setting 2 and the horses data set: classification rate at the EER averaged over trials. The error bars indicate standard deviations. (a) and (b) are results on the GRAZ02 bikes category (B versus N) only. See text for further explanations.

HSL color pixels to produce a 768-D feature vector (16×16 pixels $\times$ 3 colors). Our color wavelet descriptor transforms this into another 768D vector using a 16×16 Haar wavelet transform [52]. Finally, we tested the popular gray-scale SIFT descriptor [36], which returns 128-D vectors (4×4 histograms of eight orientations). For the GRAZ-02 database, the wavelet descriptors gave the best performance.

3.7.1 Comparing ERC-Forests with k-Means and KD-Clustering Trees

Fig. 3a shows the clear difference between our method and classical k-means for vocabulary construction. Note that we were not able to extend the k-means curve beyond 20,000 windows per image, owing to prohibitive execution times. The figure also shows results for "unsupervised trees," that is, ERC-Forests built without using the class labels during tree construction (but labels are used in later processing steps). The algorithm remains the same, but the node scoring function is defined as the ratio between the splits so as to encourage balanced trees similar to randomized KD-trees. If only a few patches are sampled, this is as good as k-means and is much faster. However, the spatial partitioning is so bad that, with additional test windows, the binarized histogram vectors become almost entirely filled with ones, so discrimination suffers. As the dotted line shows, using binarization thresholds that maximize the mutual information can fix this problem, but the results are still far below ERC-Forests. This comparison clearly shows the advantages of using supervision during clustering.

Fig. 3b shows that code books with around 5,000 entries (1,000 per tree) suffice for good results. Note that extending the k-means curve up to 20,000 would result in no clustering at all since we only use 20,000 training features. As illustrated in Fig. 3c, using more than 20,000 features in total (67 per image for GRAZ02) to learn the code book does not improve the performance. This exceptionally low number makes the method very fast for practical use. It also keeps the depth of the trees at around 20, resulting in a very high speed at assigning clusters to descriptors. Fig. 3d shows the clear advantage of the ensemble approach. Increasing the number of trees from 1 to 5 reduces the variance and increases the accuracy, with little improvement beyond this. Here, the number of leaves per tree was kept constant at 1,000, so doubling the number of trees effectively doubles the vocabulary size.

3.7.2 Adaptive Saliency Maps

Our algorithm's ability to produce meaningful visual words is illustrated in Fig. 4, where the brighter the region, the more salient it is, that is, the more likely it is that it contains an object. The visualization is obtained by projecting $\hat{P}(O|X)$ onto the image plane and normalizing to [black..white]. Note that, even though the ERC-Forest has been learned on entire images without object segmentation, it is discriminative enough to detect local structures in the test images that correspond to representative object fragments. The evolution of the ASM is shown in Fig. 5. The online estimation converges quickly and helps speed up the classification process. Of course, there is the danger that it will run into false-positive feedback loops (for example, in the "door" image on the top right). However, the exploration map provides a counterweight, which is also indicated in Fig. 3a. From 1,000 to 100,000 features, the curve stays approximately constant, which suggests that the false-positive rate does not

Fig. 4. Resulting Adaptive Saliency Maps for some example images. All but the lowest right picture are from the GRAZ02 database. The brightness denotes the posterior $\hat{P}(O|X)$ for the visual word at the given image position to be labeled "nonbackground." To get the lowest right picture, we trained the forest on all categories (bikes, cars, people) of the GRAZ02 database.

Fig. 5. Evolution of the ASM for the bottom left image in Fig. 4. Although the images in Fig. 4 are shown after $n = 2,000$ samplings, these images show $\hat{P}_n(O|X)$ after $n = 100, 300, 500, 1,000, 3,000$. Further sampling does not alter the PDF much.

increase. Thus, unlike popular keypoint detectors, ASM allows for reducing the amount of patches significantly while keeping or even increasing the classification rate (see [42]). For the GRAZ-02 data set, ASM allowed reducing processing times from 20 to 1.2 s per image while keeping the performance.

The comparison with Opelt et al. can be seen in Table 1. Remarkably, using segmentation masks during training does not improve the image classification performance. This suggests that the method is able to pick out relevant information from a significant amount of clutter.

3.7.3 Further Experiments

We also tested our method on the 2005 Pascal Challenge data set [12], which is available at http://www.pascal-network.org/challenges/VOC/voc2005. This contains four categories: motorbikes, bicycles, people, and cars. The goal is to distinguish each category from the others. Only 73 patches per image (50,000 in total over the 648 training images) were used to build the code book. Due to contained gray-scale images, the SIFT descriptor gave the best results for coding. The chosen forest contained four 7,500-leaf trees, producing a 30,000-D histogram. The results were similar to the frontrunners in the 2005 Pascal Challenge [12] but used less information and had much faster processing times. A 2.8 GHz Pentium 4 took around 20 minutes to build the code book. Building the histograms for the 684 training and 689 test images, with 10,000 patches per image, took only a few hours (compared to several days of the front-runners). All times include both feature extraction and coding.

We also compared our results with those of Marée et al. [37]. They use the same kind of tree structures to classify images directly, without introducing the vocabulary layer that we propose. Our EERs are consistently 5 to 10 percent better than theirs.

Finally, we tested the horse database from http://pascal.inrialpes.fr/data/horses. The task is difficult because the images were taken randomly from the Internet and are highly variable regarding subject size, pose, and visibility. Using SIFT descriptors, we get an EER classification rate of 85.3 percent, which is significantly better than the other methods that we are aware of [27]. We used 100 patches per image to build a code book with four trees. We used 10,000 patches per image for testing.

TABLE 1
Experimental Results: Classification Rate at EER

	GRAZ-02			
	setting 1		setting 2	
	B vs. N	C vs. N	B vs. N	C vs. N
Opelt *et al.* [44]	76.5%	70.7%	-	-
ERC-Forests	84.4%	79.9%	84.1%	79.8%

	Pascal-05 Competition 1, test 1			
	1.1	1.2	1.3	1.4
Challenge Best	**97.7%**	**93%**	91.7%	**96.1%**
Challenge Worst	72.2%	68.9%	57.1%	54.8%
ERC-Forest	95.8%	90.1%	**94.0%**	96.0%

3.8 Discussions on the Image Classification Experiments

Image classifiers based on bag of local descriptors give state-of-the art results but require the quantization of a large number of high-dimensional image descriptors into many label classes. *ERC-Forests* provide a rapid and highly discriminative solution that outperforms K-Means-based coding in training time and memory, testing time, and classification accuracy. The method works as well with segmented and unsegmented images. It is also robust to background clutter, giving relatively clean segmentation of foreground classes, even when trained on images that contain significantly more background features than foreground ones. Although they are trained as classifiers, the trees are used as descriptor-space quantization rules, with the final classification being handled by a separate SVM trained on the leaf indices. By using the ERC-Forest as classifier, *ASM* provides the possibility of estimating a category-dependent saliency map online during the classification process. We showed that this not only allows for reducing the number of patches by a factor of more than 20 but also simultaneously increases the classification performance slightly. This seems to be a promising approach for visual recognition and may be beneficial in other areas such as object detection and segmentation.

4 Learning Visual Similarity Measures Using Extremely Randomized Clustering Forests

4.1 Motivations

In this section, we use *ERC-Forests* to build an efficient similarity measure, that is, we use them to compare images of never-seen objects. Comparing two images—more generally comparing two examples—heavily relies on the definition of a good similarity function. Standard functions (for example, the euclidean distance in the original feature space) are often too generic and fail to encode *domain-specific knowledge*. This is why we propose learning a similarity measure from example images that embeds such domain-specific knowledge. Moreover, we propose learning this measure from *equivalence constraints*. The equivalence constraints considered in this section are pairs of training examples representing similar or different objects. A pair of images is not labeled "car model X and car model Y" but only "same" or "different." The latter is much more difficult because it contains less information: Same or different pairs can be produced from fully labeled examples, not vice versa. For many applications, equivalence information is cheaper to obtain than labels, for example, for retrieval systems. It is indeed easier to know whether two documents are similar or not rather than to obtain their true labels because the space of potential labels is very large (for example, all car models) and is difficult to define.

We use this similarity measure for *visual identification of never-seen objects*. Given a training set of pairs labeled "same" or "different," we have to decide if two never-seen objects are the same or not (see Fig. 6).

4.2 Similarity Measures

Learning effective functions to compare examples is an active topic that has received much attention during recent years. Most of the contributions consist of finding a function mapping the feature space into a target space such that a simple distance can eventually be used in the target space.

This function is generally inspired by the Mahalanobis distance $d(x,y) = (x-y)^t A(x-y)$, as in [35] or, more recently, [58], [22], [50], [3], [21], [56]. Various optimization schemes are possible for estimating A, depending on the objective function to be satisfied. The objective function plays a key role in the definition of the metric. In [58], [22], the objective function tries to collapse all examples of the same class and pushes examples in other classes. In [22], a stochastic variant of the leave-one-out k-NN score is maximized. In [56], the objective function tries separating examples from different classes by a large margin in a k-NN framework. In [50], the margin between positive pairs and negative pairs is to be maximized. In [3], A is directly computed from the so-called *chunklets*, which are the sets of equivalence relations provided as training data. The mapping can also be learned without explicit functions, as in [10], where a convolutional network is used for its robustness to geometric distortions. When considering distance between images, more specific functions can be used, embedding expected deformations of object appearances [39], [16], [33].

Unfortunately, none of these methods is perfectly suited for visual identification in images. Unlike traditional pattern recognition problems, information included in images is subject to complex transformations such as occlusions, pose, and scale changes that cannot be modeled easily by any kind of linear, quadratic, or other polynomial

Fig. 6. Given pairs labeled "same" or "different," can we learn a similarity measure that decides if two images represent the same object? The similarity measure should be robust to modifications in pose, background, and lighting conditions and, above all, should deal with *never-seen* objects.

transformations. Furthermore, the objects to be identified cover only a fraction of the image.

The usual way of facing these problems is to represent images as a collection of loose scale invariant local information (gray-scale patches, SIFT descriptors, or others). As in the previous section, this leaves at least several parts of the image not being affected by these transformations. In this kind of strategy, already used in [14], [13], the key idea is to learn what characterizes features (local descriptors) that are informative in distinguishing one object instance from another. We also have to mention the interesting approach based on *chopping* [17]. However, this approach, which relies on random binary splits chosen to keep images of the same object together, requires having all the training images fully labeled and, therefore, it is not usable in our context.

Inspired by the work proposed in these related approaches, more particularly in [14], we propose a new learning method for measuring similarities between two images of never-seen objects by using information extracted from pairs of similar and different objects of the same generic category.

4.3 Quantizing Local Differences

The proposed *ERC-Forest* is the cornerstone of this new image distance. As we mentioned earlier, the measure is learned by detecting characteristic differences between local descriptors sampled from pairs of images, labeled as *same* or *different*. These differences are vector quantized by an ERC-Forest and the similarity measure is computed from the quantization. All recent image classification approaches based on local representation and clustering deal with local descriptor quantization, but, to the best of our knowledge, our approach is the first to quantize local descriptor *differences*.

We evaluate our innovative similarity measure on four very different data sets and always outperform the state-of-the-art competitive approaches.

4.4 Building a Similarity Measure from Patch Correspondences

The computation of the similarity measure of two images is a three-step process illustrated in Fig. 7. In Fig. 7a, several pairs of corresponding local regions (patch pairs) are sampled from a pair of images. In Fig. 7b, each patch pair is assigned to several clusters by an ERC-Forest. In Fig. 7c, the cluster memberships are combined to make a global decision about the pair of images. The steps are detailed as follows:

1. *Sampling corresponding patch pairs*. A pair of images is reduced to the set of corresponding patch pairs sampled from it, each of which is produced as follows (see Fig. 8): A patch p_1 of a random size is chosen at a random position (x, y) in the first image I_1. The best normalized cross correlation (NCC) match p_2 is looked for in the second image I_2 in the neighborhood of (x, y), for example, a search region twice as big as the patch.
2. *Clustering the space of patch pair differences*. Each patch pair sampled from an image pair is assigned to several clusters via an ERC-Forest, analogously to the previous section. Thus, an image pair is transformed into a binary vector x (of size similar to the total number of leaves), each dimension indicating if a patch pair sampled from the image pair has reached the corresponding leaf (see Fig. 7).

   The forest is constructed as in Section 2.2, except that we use Boolean tests on patch pairs (detailed later in Section 4.5) instead of feature-threshold tests on single patches, as we are now dealing with the difference between two patches and not with the appearance of a single patch.
3. *The similarity measure*. The learned trees can perfectly discriminate the patch pairs that they were trained on, as they were trained to do so. However, we are again not using the trees as classifiers; instead, we consider which leaves are reached by the patch pairs. Indeed, as in the previous sections, trees are used as clusterers and not as classifiers.

   The similarity measure is defined as a linear combination of the binary feature vector indicating cluster membership, that is, $S_{lin}(I_1, I_2) = \omega^\top x$, where ω contains weights optimized such that high values of S_{lin} correspond to similar images. In practice, ω is the hyperplane normal of a binary linear SVM trained on positive and negative image pair representations.

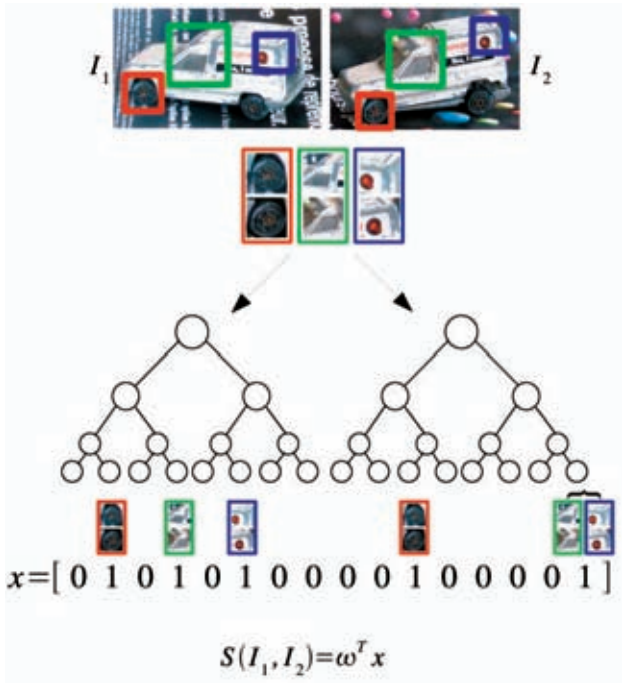

Fig. 7. Similarity computation. (a) Detect corresponding patch pairs. (b) Affect them to clusters via Extremely Randomized Clustering Forest (encoded in x). (c) The similarity is a linear combination of the cluster memberships.

4.5 A Multimodal Algorithm

The Boolean tests evaluated in each internal tree node are simple tests on a pair of patches. Unlike in Sections 2 and 3, we do not use the test $(f_i < \theta)$ in the nodes anymore, but we define two kinds of Boolean tests based on different feature types (SIFT descriptors and geometry). That is, when constructing a decision node, we also have to select which kind of test to apply within this node. Thus, we have to enhance the algorithm in Section 2.2 to deal with this new degree of freedom. We do this by proposing a multimodal

Fig. 8. Patch pairs sampled on a toy car data set. Each image pair shows a random patch in image 1, the search region in image 2, and the best match in the search region. All image pairs are positive (same object), except for the last one (different objects).

algorithm which automatically selects the test type and its parameters most adapted to a given node.

4.5.1 SIFT-Based Test Type

Given the SIFT descriptors S_1 and S_2 of two patches, the test is true if

$$k(S_1(i) - d) > 0 \ \wedge \ k(S_2(i) - d) > 0, \tag{9}$$

where i, d, and k are parameters. i is the SIFT dimension under observation, d is a threshold, and $k = 1$ or $k = -1$ encodes whether the measured value should be higher or lower than the threshold.

4.5.2 Geometry-Based Test Type

Given the position and scale (x, y, s) of the patch from the first image, the split condition is true if

$$k_x(x - d_x) > 0 \ \wedge \ k_y(y - d_y) > 0 \ \wedge \ k_s(s - d_s) > 0, \tag{10}$$

where d_x, d_y, d_s, k_x, k_y, and k_s are parameters. k_x, k_y, and k_s are equal to 1 or -1 and encode whether the values should be above or below the thresholds d_x, d_y, and d_s. This split condition can encode complex concepts. For example, a large patch in the bottom left corner of an image may be

$$\begin{aligned} -1 * (x - 0.25) > 0 \ \wedge \ 1 * (y - 0.75) \\ > 0 \ \wedge \ 1 * (s - 0.5) > 0. \end{aligned} \tag{11}$$

4.5.3 Test Type Combination

For each tree node, we generate random Boolean tests of any type (SIFT or geometry). First, we randomly draw a type and, then, we randomly draw the parameters that it requires. The information gain is computed for all of these Boolean tests and the best one is assigned to the node.

4.6 Experiments on Learning a Similarity Measure

We evaluate our similarity measure on four different data sets, a small data set of toy cars[3] and three other publicly available data sets, making comparison with competitive approaches possible. For each data set, the objects of interest fully occupy the images and we have pairs marked as positive (same object) or negative (different objects). Those sets are split into a training set and a test set. Obviously, the test set does not contain any image from the training set, but it does not contain any object of the training set either. The similarity measure is evaluated only on *never-seen* objects (not just never-seen images).

To evaluate the performance, we compute a Precision-Recall Equal Error Rate score on the similarity measure evaluated on the test set image pairs. For each test set image pair, a similarity score S_{lin} is computed. A threshold t is defined to decide if the two images represent the same object instance or not. $S_{lin} > t$ means the "same" object, while $S_{lin} \leq t$ means "different" objects. The Precision-Recall curve is obtained by varying the threshold t.

The four data sets contain different types of images:

- The toy car data set contains 225 images of 14 different objects (cars and trucks). The training set contains 1,185 positive and 7,330 negative image pairs of seven different objects. The test set contains 1,044 positive and 6,337 negative image pairs of the seven other (new) objects. The background of the images has been selected to increase the difficulty of the problem. As it was moved between any two shots, it is noninformative (the background is not correlated with any car model). Moreover, as the same background image is used for all car models, many patch pairs sampled from "different" images look similar. Therefore, the algorithm has to ignore that kind of similarity and should focus on other (more relevant) similarities.
- The Ferencz et al. cars data set [14] contains 2,868 training pairs (180 positive and 2,688 negative) and the test set contains 2,860 pairs.
- The Jain et al. faces data set [26] is a subset of "Faces in the news"[4] [5] and contains 500 positive and 500 negative pairs of faces. We measure our accuracy as the authors did, by a 10-fold cross validation. That data set is built from faces sampled "in the news"; hence, there are very large differences in resolution, light, appearance, expression, pose, noise, etc.
- The Coil-100 data set used by Fleuret and Blanchard [17] has 10 different configurations, each configuration using 1,000 positive and 1,000 negative pairs from 80 objects for training and 250 positive and 250 negative pairs from the remaining 20 (new) objects for testing. This data set is highly heterogeneous as it contains object categories as different as bins, tomatoes, boxes, medicine, puppets, mugs, bottles, etc.

In all of the experiments, gray-scale and color images are all considered gray scale. All data sets have image pairs of slightly different orientations, except for Coil-100, which

3. http://lear.inrialpes.fr/people/nowak/dwl/toycarlear.tar.gz.

4. We thank the authors for providing us the precise subset.

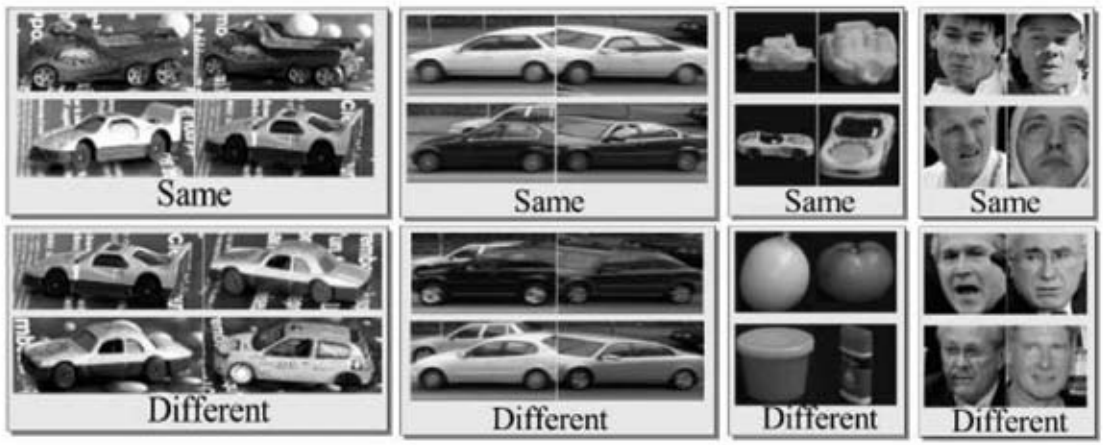

Fig. 9. "Different" and "Same" pairs from our (a) toy car data set, (b) the Ferencz et al. cars data set, (c) the Fleuret and Blanchard Coil-100 data set, and (d) the Jain et al. faces data set. Although "Different" pairs may look similar, "Same" pairs may look different, and the test set objects are never seen, our similarity measure obtains a very high performance on all of these data sets (see Section 4.6.2).

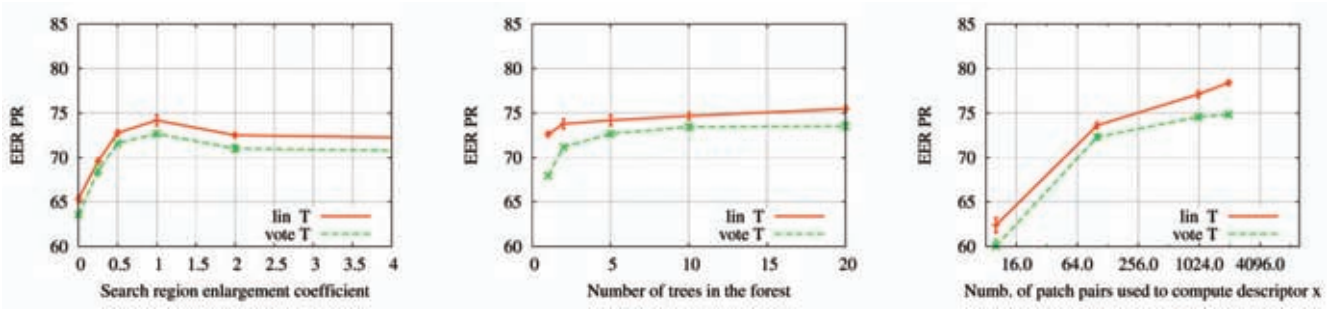

Fig. 10. Precision-Recall Equal Error Rate for the toy car data set, with a simple similarity measure S_{vote} and our linear similarity measure S_{lin}. All parameters are fixed to their standard value, except for the one that is studied.

has very different orientations. The data sets are illustrated in Fig. 9.

4.6.1 Parametric Evaluation

We have evaluated the influence of all of the parameters involved in the similarity measure on the toy car data set. Fig. 10 shows the effect of influential parameters.

Each experiment plots the Precision-Recall Equal Error Rate (EER-PR) with respect to a parameter, all other parameters being fixed. We plot the EER-PR of the linear similarity measure S_{lin} and also the EER-PR of a simple vote-based similarity measure S_{vote}. The measure S_{vote} simply counts the number of times that positive tree leaves are reached during the description of an image pair.

We first notice that the linear similarity measure S_{lin} always outperforms the simple similarity S_{vote}, which proves that the use of weights is necessary. Second, let us discuss the different parameters one by one. The first curve shows that the search region size should not be too small (otherwise, it is impossible to find the expected match) nor too large (leading to too many misleading matches). Moreover, if the second patch is selected randomly, the performance is very bad (42.7 percent, which is not shown on the graph). This shows how crucial the computation of good patch pairs is. The second curve shows that the more trees there are, the higher the performance is. This is because we obtain more clusters and because the clusters are not correlated due to the randomness of the tree computation (also compare Section 3.7). The third curve shows that the more the patch pairs sampled in an image pair, the higher the performance is, which is also consistent with the results in Section 3.7. We first believed that the reason for better results through sampling more windows is because it increases chances to sample *relevant* information. However, if this were true, only S_{lin} would progress because it is able to separate relevant and irrelevant information. However, S_{vote} also increases and that measure makes no difference between relevant and irrelevant information. This means that any "weak" information also improves the performance, which confirms the conclusion in [42] about sampling.

4.6.2 Performance and Comparison with State-of-the-Art Competitive Approaches

We report the performance of our algorithm and compare our results with state-of-the-art methods on the four data sets. For each data set, the experiments are carefully performed with the same protocol as the one used in the method compared to. The results are summarized in Table 2.

For all of these experiments on the different data sets, we use the same parameters: The number of positive and negative patch pairs sampled to learn the trees is set to 10^5, the number of random split conditions, among which the best one is selected, is 10^3, the number of trees in the forest is 50, the number of patch pairs sampled to compute the similarity is 10^3, the second patch search region size is increased by 1 times the size of the patch in all directions, leading to a search region nine times larger than the

TABLE 2
Comparison with the State of the Art

Method	Toy cars	Ferencz	Faces	Coil 100
Others	-	84.9 [13]	70.0 [26]	$88.6_{\pm 4}$ [17]
Ours	$85.9_{\pm 0.4}$	$91.0_{\pm 0.6}$	$84.2_{\pm 3.1}$	$93.0_{\pm 1.9}$
Gain	-	6.1	14.2	4.4

Method	Recall 40%	Recall 60%	Recall 80%
Jain [26]	$93.0_{\pm 6.3}$	$78.9_{\pm 8.2}$	$60.1_{\pm 7.0}$
Ours	$99.0_{\pm 1.9}$	$97.8_{\pm 2.5}$	$86.3_{\pm 7.6}$
Gain	6	18.9	26.2

(Top) PR-EER on different data sets. (Bottom) Precision on the face data set for given recall. Our method clearly outperforms the others.

original patch, the minimum patch sampling size is set to 15×15 pixels, and the maximum is set to $1/2$ the image height. On average, the produced trees have 20 levels.

Toy car data set. On our toy car data set, we obtain an EER-PR of 85.9 percent $\pm$ 0.4 measured on five runs. Using a simple NCC as a similarity measure leads to an EER-PR of 51.1 percent. This is a new data set for which no other results are published yet. Fig. 11 shows a 2D representation of the similarity measures via a multidimensional scaling. It computes the 2D projection, preserving, as much as possible, all pairwise image similarities (the Sammon mapping). On the top, a simple bag-of-words representation is used for the images and a chi-square distance is used to compare images. On the bottom, we use the proposed similarity measure. It is surprising to see how different views of the same objects are extremely well grouped together, despite large intraclass variations, high interclass similarities, and the fact that these cars are very different from the ones used to learn the distance.

Ferencz et al. data set. On the Ferencz et al. data set, we obtain an EER-precision of 91.0 percent $\pm$ 0.6, whereas Ferencz et al. [13] get 84.9 percent. Fig. 12 shows the pairs of cars ranked by similarity.

Faces-in-the-news data set. The Jain et al. data set [26] greatly outperforms the top performer in the FERET face recognition competition on their face data set and we largely outperform their results. The EER-PR reported in Table 2, top, (70 percent) is approximate since we have estimated it from a curve in their paper; thus, we also provide a comparison with the same metric as Jain et al. used, that is, the precision score for given recall values (see Table 2 (bottom)). We always outperform their results and are even 26.2 percent better for an 80 percent recall. Moreover, Jain et al. are working on face images rectified to the frontal pose, whereas we are working on the original face images.

Coil-100 data set. On the Coil-100 data set, we have an EER-PR of 93.0 percent $\pm$ 1.9, whereas Fleuret and Blanchard [17] have 88.6 percent $\pm$ 4. Moreover, the method of Fleuret and Blanchard uses the information of the real object categories during training, whereas we only know if two images belong to the same category or not. Obdrzalek et al. [43] have obtained very good results on the Coil-100 data set (more than 99 percent) but with totally different experimental settings. They are using eight training images per category, whereas we are using zero training images per category. Indeed, 80 object categories are used for training and the remaining 20 (new) categories are used for testing (as mentioned at the beginning of Section 4.6).

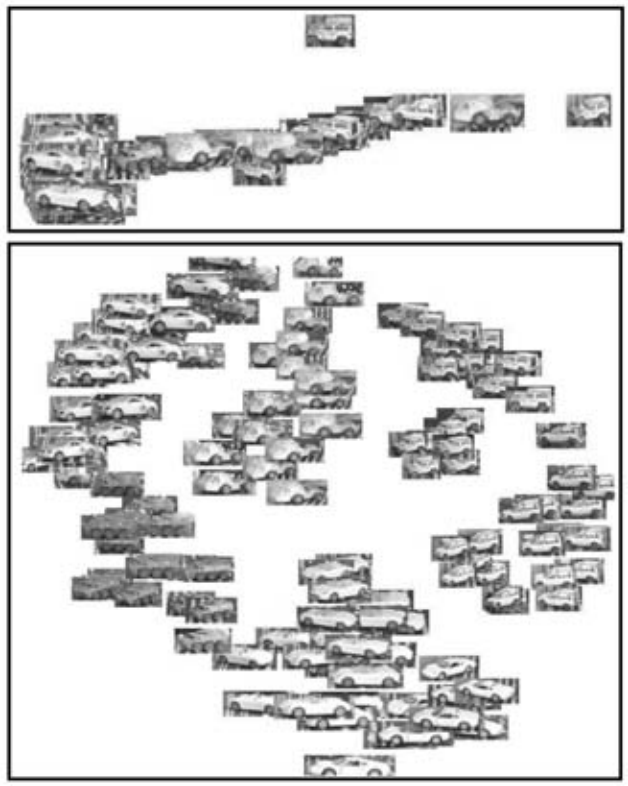

Fig. 11. Two-dimensional multidimensional scaling representation of the similarities between never-seen toy cars. Top: Similarity measure based on bag-of-words representation and Chi square distance. Bottom: Our similarity measure, which nicely groups the different views of the same object.

4.6.3 Computational Time

In our work, 95 percent of the computation time is dedicated to the computation of matching patch pairs. This is clearly the bottleneck of the algorithm. The process is very slow because, for each patch sampled from the first image, the best matching patch has to be searched for by NCC in the second image.

It takes approximately 2 s to compute the similarity between two images (whatever the data set) on a 900 MHz Itanium 2 running a C++ program. This involves the computation of 250 patch pairs, their SIFT descriptor computation, the image pair description x (via the trees), and the similarity measure computation (via the scalar product $\omega^\top x$).

4.7 Discussion

We address the problem of predicting how similar two images of never-seen objects are given a set of similar and different training object pairs. We propose an original method consisting of 1) finding corresponding local regions in a pair of images, 2) clustering their *differences* with an *ERC-Forest*, and 3) combining the cluster memberships of the local region pairs to compute a global similarity measure between the two images.

Our algorithm automatically selects and combines the feature types (SIFT descriptors and geometry information) that are the most appropriate to a data set.

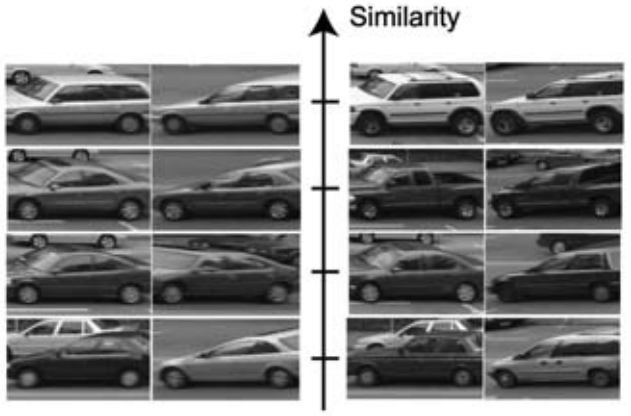

Fig. 12. Test set image pairs from the Ferencz et al. data set ranked from the most different (bottom) to the most similar (top) by our similarity measure.

Our experiments show that our approach gives excellent results on the four data sets used for evaluation. We outperform the latest results of Ferencz et al. [13], Jain et al. [26], and Fleuret and Blanchard [17] significantly and obtain a high accuracy on our own data set.

5 Conclusions

Image classification and image search are topics of high interest because of the rapid growth of digital image and video collections. Classification and search algorithms heavily depend on image information extraction (which should focus on valuable information) and image representation (which should lead to efficient algorithms).

In this paper, we have proposed three key contributions. First, we introduced a new clustering scheme, the so-called *Extremely Randomized Clustering Forest*, which is very efficient for vector quantization of local visual information. Its randomized process makes it very fast to build, while its tree structure allows very fast feature vector quantization. *ERC-Forests* outperform k-means-based coding in training time, memory usage, testing time, and classification accuracy. The method is robust to background clutter and provides relatively clean foreground class segmentations, despite being trained on very cluttered unsegmented images. Although they are trained as classifiers, the trees are used as descriptor-space quantizers and the final classification is handled by a separate SVM trained on the leaf indices.

The soundness of *ERC-Forests* has been proven in two different contexts: image classification and image comparison. The proposed image classification method combines *ERC-Forests* with visual search mechanisms so that only useful visual information is processed. The ASM computed online significantly reduces the amount of computation and increases the classification accuracy. In the context of learning the distance between images, we apply *ERC-Forests* to pairs of registered image patches, which leads to a very efficient similarity function for image comparison. On four different data sets, the method gives better results than state-of-the-art competitive methods.

ERC-Forests seem to be a promising approach for visual recognition and may be beneficial in other areas such as object detection and segmentation.

Acknowledgments

The authors are thankful to Bill Triggs for his useful comments and discussions. This work is supported in part by the PASCAL Network of Excellence.

References

[1] Y. Amit, D. Geman, and K. Wilder, *Joint Induction of Shape Features and Tree Classifiers,* 1997.
[2] T. Avraham and M. Lindenbaum, "Dynamic Visual Search Using Inner Scene Similarity: Algorithms and Inherent Limitations," *Proc. Eighth European Conf. Computer Vision,* 2004.
[3] A. Bar Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a Mahalanobis Metric from Equivalence Constraints," *J. Machine Learning Research,* vol. 6, pp. 937-965, 2005.
[4] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning J.,* vol. 36, nos. 1-2, pp. 105-139, 1999.
[5] T.L. Berg, A.C. Berg, J. Edwards, M. Maire, R. White, Y.-W. Teh, E. Learned-Miller, and D.A. Forsyth, "Names and Faces in the News," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 848-854, 2004.
[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When Is Nearest Neighbors Meaningful," *Proc. Seventh Int'l Conf. Database Theories),* pp. 217-235, 1999.
[7] H. Blockeel, L. De Raedt, and J. Ramon, "Top-Down Induction of Clustering Trees," *Proc. 15th Int'l Conf. Machine Learning,* pp. 55-63, 1998.
[8] J. Bonaiuto and L. Itti, "Combining Attention and Recognition for Rapid Scene Analysis," *Proc. Third Int'l Workshop Attention and Performance in Computational Vision,* 2005.
[9] L. Breiman, "Random Forests," *Machine Learning J.,* vol. 45, no. 1, pp. 5-32, 2001.
[10] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 539-546, 2005.
[11] G. Csurka, C. Dance, L. Fan, J. Williamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," *Proc. ECCV Workshop Statistical Learning in Computer Vision,* pp. 59-74, 2004.
[12] M. Everingham, "The 2005 PASCAL Visual Object Classes Challenge," *Proc. First PASCAL Challenges Workshop,* 2006.
[13] A. Ferencz, E.G. Learned Miller, and J. Malik, "Building a Classification Cascade for Visual Identification from One Example," *Proc. 10th IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 286-293, 2005.
[14] A.D. Ferencz, E.G. Learned-Miller, and J. Malik, "Learning Hyper-Features for Visual Identification," *Proc. Ann. Conf. Neural Information Processing Systems,* pp. 425-432, 2005.
[15] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning Object Categories from Google's Image Search," *Proc. 10th IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 1816-1823, 2005.
[16] A.W. Fitzgibbon and A. Zisserman, "Joint Manifold Distance: A New Approach to Appearance-Based Clustering," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 26-33, 2003.
[17] F. Fleuret and G. Blanchard, "Pattern Recognition from One Example by Chopping," *Proc. Ann. Conf. Neural Information Processing Systems,* pp. 371-378, 2005.
[18] G. Fritz, C. Seifert, L. Paletta, and H. Bischof, "Entropy-Based Saliency Maps for Object Recognition," *Proc. Workshop Early Cognitive Vision,* 2004.
[19] A. Frome, Y. Singer, and J. Malik, "Image Retrieval and Classification Using Local Distance Functions," *Proc. Ann. Conf. Neural Information Processing Systems,* 2006.
[20] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," *Machine Learning J.,* vol. 63, no. 1, 2006.
[21] A. Globerson and S. Roweis, "Metric Learning by Collapsing Classes," *Proc. Ann. Conf. Neural Information Processing Systems,* 2005.
[22] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood Components Analysis," *Proc. Ann. Conf. Neural Information Processing Systems,* 2004.
[23] D. Hall, B. Leibe, and B. Schiele, "Saliency of Interest Points under Scale Changes," *Proc. 13th British Machine Vision Conf.,* 2002.

[24] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf.,* 1988.
[25] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1254-1259, Nov. 1998.
[26] V. Jain, A. Ferencz, and E.G. Learned Miller, "Discriminative Training of Hyper-Feature Models for Object Identification," *Proc. 17th British Machine Vision Conf.,* vol. 1, pp. 357-366, 2006.
[27] F. Jurie and C. Schmid, "Scale-Invariant Shape Features for Recognition of Object Categories," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 90-96, 2004.
[28] F. Jurie and B. Triggs, "Creating Efficient Codebooks for Visual Recognition," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.
[29] T. Kadir and M. Brady, "Saliency, Scale and Image Description," *Int'l J. Computer Vision,* vol. 45, no. 2, Nov. 2001.
[30] B. Leibe and B. Schiele, "Interleaved Object Categorization and Segmentation," *Proc. 14th British Machine Vision Conf.,* 2003.
[31] V. Lepetit, P. Lagger, and P. Fua, "Randomized Trees for Real-Time Keypoint Recognition," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 775-781, 2005.
[32] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons," *Int'l J. Computer Vision,* vol. 43, no. 1, pp. 29-44, June 2001.
[33] F.F. Li, R. Fergus, and P. Perona, "One-Shot Learning of Object Categories," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 594-611, Apr. 2006.
[34] B. Liu, Y. Xia, and P.S. Yu, "Clustering through Decision Tree Construction," *Proc. Ninth ACM Int'l Conf. Information and Knowledge Management,* pp. 20-29, 2000.
[35] D.G. Lowe, "Similarity Metric Learning for a Variable-Kernel Classifier," *Neural Computation,* vol. 7, no. 1, pp. 72-85, 1995.
[36] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, 2004.
[37] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Random Subwindows for Robust Image Classification," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 34-40, 2005.
[38] K. Mikolajczyk and C. Schmid, "Scale and Affine Invariant Interest Point Detectors," *Int'l J. Computer Vision,* vol. 60, no. 1, pp. 63-86, 2004.
[39] E.G. Miller, N.E. Matsakis, and P.A. Viola, "Learning from One Example through Shared Densities on Transforms," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 464-471, 2000.
[40] V. Navalpakkam and L. Itti, "Sharing Resources: Buy Attention, Get Recognition," *Proc. First Int'l Workshop Attention and Performance in Computational Vision,* 2003.
[41] D. Nistér and H. Stewénius, "Scalable Recognition with a Vocabulary Tree," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* 2006.
[42] E. Nowak, F. Jurie, and B. Triggs, "Sampling Strategies for Bag-of-Features Image Classification," *Proc. Ninth European Conf. Computer Vision),* 2006.
[43] S. Obdrzlek and J. Matas., "Sub-Linear Indexing for Large Scale Object Recognition," *Proc. 16th British Machine Vision Conf.,* 2005.
[44] A. Opelt and A. Pinz, "Object Localization with Boosting and Weak Supervision for Generic Object Recognition," *Proc. 14th Scandinavian Conf. Image Analysis,* 2005.
[45] F. Perronnin, C. Dance, G. Csurka, and M. Bressan, "Adapted Vocabularies for Generic Visual Categorization," *Proc. Ninth European Conf. Computer Vision,* 2006.
[46] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of Interest Point Detectors," *Int'l J. Computer Vision,* vol. 37, no. 2, pp. 151-172, June 2000.
[47] N. Sebe and M.S. Lew, "Comparing Salient Point Detectors," *Pattern Recognition Letters,* vol. 24, nos. 1-3, pp. 89-96, Jan. 2003.
[48] T. Serre, M. Riesenhuber, J. Louie, and T. Poggio, "On the Role of Object-Specific Features for Real-World Object Recognition in Biological Vision," *Proc. 13th British Machine Vision Conf.,* 2002.
[49] U. Shaft, J. Goldstein, and K. Beyer, "Nearest Neighbor Query Performance for Unstable Distributions," Technical Report TR 1388, Dept. of Computer Science, Univ. of Wisconsin, 1998.
[50] S. Shalev-Shwartz, Y. Singer, and A.Y. Ng, "Online and Batch Learning of Pseudo-Metrics," *Proc. 21st Int'l Conf. Machine Learning,* 2004.
[51] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 1470-1477, Oct. 2003.
[52] E. Stollnitz, T.D. DeRose, and D.H. Salesin, "Wavelets for Computer Graphics: A Primer—Part 1," *IEEE Computer Graphics and Applications,* vol. 15, no. 3, pp. 76-84, 1995.
[53] M. Vidal-Naquet and S. Ullman, "Object Recognition with Informative Features and Linear Classification," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 281-288, 2003.
[54] K.N. Walker, T.F. Cootes, and C.J. Taylor, "Locating Salient Object Features," *Proc. 13th British Machine Vision Conf.,* 1998.
[55] D. Walther, U. Rutishauser, C. Koch, and P. Perona, "On the Usefulness of Attention for Object Recognition," *Proc. Eighth European Conf. Computer Vision,* 2004.
[56] K. Weinberger, J. Blitzer, and L. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *Proc. Ann. Conf. Neural Information Processing Systems,* 2006.
[57] J. Winn, A. Criminisi, and T. Minka, "Object Categorization by Learned Universal Visual Dictionary," *Proc. 10th IEEE Int'l Conf. Computer Vision,* pp. 1800-1807, 2005.
[58] E.P. Xing, A.Y. Ng, S. Jordan, and M.I. Russell, "Distance Metric Learning, with Application to Clustering with Side Information," *Proc. Ann. Conf. Neural Information Processing Systems,* 2002.
[59] Y. Ye and J.K. Tsotsos, "Where to Look Next in 3D Object Search," *Proc. IEEE Int'l Symp. Computer Vision,* 1995.
[60] A. Zaharescu, A.L. Rothenstein, and J.K. Tsotsos, "Towards a Biologically Plausible Active Visual Search Model," *Proc. Third Int'l Workshop Attention and Performance in Computational Vision,* pp. 133-147, 2004.
[61] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision,* 2006.

Frank Moosmann received the MS degree in computer science from the University of Karlsruhe, Germany, in 2006. For his master's thesis, he joined the LEAR Research Group, INRIA, Rhone-Alpes. He is currently working toward the PhD degree at the Institute of Measurement and Control, University of Karlsruhe. His research interests include computer vision, machine learning, and autonomous vehicles. He is a student member of the IEEE and the IEEE Computer Society.

Eric Nowak received the MS degree in computer science from the University of Paris VI and the engineering school EFREI. He is currently working toward the PhD degree. His research interests include computer vision (object and object class recognition) and machine learning. He is a student member of the IEEE and the IEEE Computer Society. In his free time, he enjoys photography, travels, and reading. As there is plenty of free space, he is happy to greet all his friends here.

Frederic Jurie received the PhD degree in computer science from the University of Clermont. He is a professor at the University of Caen and an associate researcher at INRIA, France. He joined the Lear Group, Institut National de Recherche en Informatique et en Automatique (INRIA), in 2003. He has served on a variety of workshop and conference program committees such as the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), IEEE International Conference on Computer Vision (ICCV), and European Conference on Computer Vision (ECCV). His research interests include movement and object detection, visual tracking, and image understanding, especially image classification and object recognition. He is the author of more than 50 publications in computer vision and related fields. He is a member of the IEEE Computer Society.