Fast Keypoint Recognition Using Random Ferns

Mustafa Özuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua, *Senior Member*, *IEEE*

Abstract—While feature point recognition is a key component of modern approaches to object detection, existing approaches require computationally expensive patch preprocessing to handle perspective distortion. In this paper, we show that formulating the problem in a naive Bayesian classification framework makes such preprocessing unnecessary and produces an algorithm that is simple, efficient, and robust. Furthermore, it scales well as the number of classes grows. To recognize the patches surrounding keypoints, our classifier uses hundreds of simple binary features and models class posterior probabilities. We make the problem computationally tractable by assuming independence between arbitrary *sets* of features. Even though this is not strictly true, we demonstrate that our classifier nevertheless performs remarkably well on image data sets containing very significant perspective changes.

Index Terms—Image processing and computer vision, object recognition, tracking, image registration, feature matching, naive Bayesian.

✦

---

1 INTRODUCTION

IDENTIFYING textured patches surrounding keypoints across images acquired under widely varying poses and lightning conditions is at the heart of many computer vision algorithms. The resulting correspondences can be used to register different views of the same scene, extract 3D shape information, or track objects across video frames. Correspondences also play a major role in object category recognition and image retrieval applications.

In all of these cases, since patch recognition is one of the first and most critical stages in the algorithmic pipeline, reliability and speed are key to the overall success and practicality of the corresponding applications. Achieving both is difficult because surface appearance around a keypoint can vary drastically depending on how the images were captured. The standard approach to addressing this problem is to build affine-invariant descriptors of local image patches and to compare them across images. This usually involves fine-scale selection, rotation correction, and intensity normalization [23], [22]. It results in a high computational overhead and often requires handcrafting the descriptors to achieve insensitivity to specific kinds of distortion.

In earlier work [20], we have shown that casting the matching problem as a more generic classification problem leads to solutions that are much less computationally demanding. This approach relies on an offline training phase during which multiple views of the patches to be matched are used to train randomized trees [2] to recognize them based on a few pairwise intensity comparisons. This

yields both fast runtime performance and robustness to viewpoint and lighting changes.

Here, we show that the trees can be profitably replaced by nonhierarchical structures that we refer to as *ferns* to classify the patches. Each one consists of a small set of binary tests and returns the probability that a patch belongs to any one of the classes that have been learned during training. These responses are then combined in a naive Bayesian way. As before, we train our classifier by synthesizing many views of the keypoints extracted from a training image as they would appear under different perspectives or scales. The ferns are just as reliable as the randomized trees but much faster and simpler to implement. The code that implements patch evaluation can be written in 10 lines of C++ code, which highlights the simplicity of the resulting implementation and the complete software package is available [29].[1]

The binary tests we use as classifier features are picked completely at random, which puts our approach firmly in the camp of techniques that rely on randomization to achieve good performance [1]. We will show that this is particularly effective for applications such as real-time 3D object detection and *Simultaneous Localization and Mapping* (SLAM) that require scale and perspective invariance, involve a very large number of classes, but can tolerate significant error rates since we use robust statistical methods to exploit the information provided by the correspondences. Furthermore, our approach is particularly easy to implement, does not overfit, does not require ad hoc patch normalization, and allows fast and incremental training.

2 RELATED WORK

Due to its robustness to partial occlusions and computational efficiency, recognition of image patches extracted around detected keypoints is crucial for many vision problems. As a result, two main classes of approaches have

● *The authors are with the Ecole Polytechnique Fédérale de Lausanne (EPFL)/IC/ISIM/Computer Vision Laboratory (CVLab), CH-1015 Lausanne, Switzerland. E-mail: {mustafa.oezuysal, michael.calonder, vincent.lepetit, pascal.fua}@epfl.ch.*

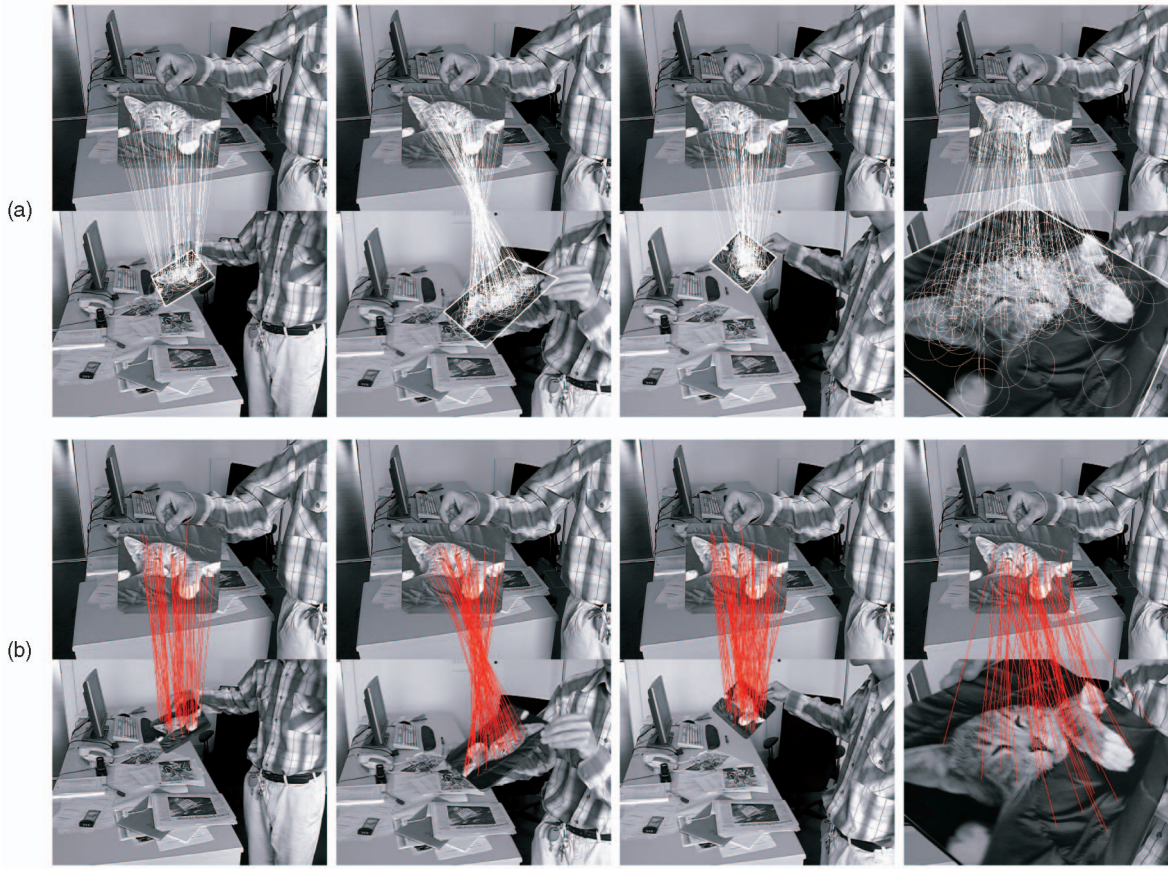1. We make the code available at http://cvlab.epfl.ch/software.

Fig. 1. Matching a mouse pad in a 1,074-frame sequence against a reference image. The reference image appears in (a) and the input image from the video sequence in (b). (a) Matches obtained using ferns in a few frames. (b) Matches obtained using SIFT in the same frames.

been developed to achieve robustness to perspective and lighting changes.

The first family relies on local descriptors designed to be invariant, or at least robust, to specific classes of deformations [30], [22]. These approaches usually rely on the fine scale and rotation estimates provided by the keypoint detector. Among these, the SIFT vector [22], computed from local histograms of gradients, has been shown to work remarkably well and we will use it as a benchmark for our own approach. It has also been shown that keypoints can be used as visual words [31] for image retrieval in very large image databases [27]. Keypoints are labeled by hierarchical k-means [5] clustering based on their SIFT descriptors. This makes it possible to use very many visual words. However, performance is measured in terms of the number of correctly retrieved documents rather than the number of correctly classified keypoints, which is the important criterion for applications such as pose estimation or SLAM.

A second class relies on statistical learning techniques to compute a probabilistic model of the patch. The one-shot approach of [13] uses PCA and Gaussian Mixture Models but does not account for perspective distortion. Since the set of possible patches around an image feature under changing perspective and lightning conditions can be seen as a class, we showed that this problem can be overcome by training a set of Randomized Trees [2] to recognize feature points independently of pose [20]. This is done using a database of patches obtained by warping those found in a training image by randomly chosen homographies.

This approach is fast and effective to achieve the kind of object detection depicted in Figs. 1 and 2. Note that, unlike in traditional classification problems, a close-to-perfect method is not required, because the output of the classifier can be filtered by a robust estimator. However, the classifier should be able to handle many classes—typically more than
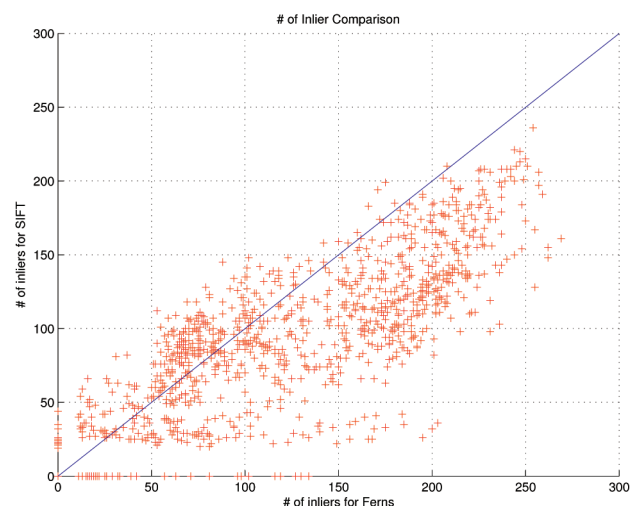

Fig. 2. Scatter plot showing the number of inliers for each frame for the experiment in Fig. 1. The values on the axes give the number of inliers for Ferns and SIFT. Most of the time, the Ferns match at least as many points as SIFT and, often, even more, as can be seen from the fact that most of the points lie below the diagonal.

200—simultaneously without compromising on performance or speed. We will demonstrate that the approach we propose is even faster than the trees [20], more scalable, and just as reliable.

Local Binary Patterns [28] also rely on binary feature statistics by describing the underlying texture in terms of histograms of binary features over all pixels of a target region. While such a description is appropriate for texture classification, it is not directly suitable for keypoint characterization since, if histograms are built in this way, it will lose the spatial information between the features. By contrast, we compute the statistics of the binary features over example patches seen from different viewpoints and use independence assumptions between groups of features, hence using many more features centered on the keypoint location so as to improve the recognition rate.

The Real-Time SLAM method of [35] also extends the Randomized Trees (RTs) into lists of features (similar to our Ferns), but the full posterior distribution over binary features are replaced by a single bit. This design choice is aimed at significantly reducing the memory requirements while correctly matching the sparse set of visible landmarks at a time. For maximum performance, we model the full joint probability. Memory requirements can be tackled by using fixed-point representations that require fewer bits than the standard floating-point representation.

Trees and Ferns have recently been used for image classification as a replacement for a multiway Support Vector Machine [6]. The binary features are computed on shape and texture descriptors, hence gaining invariance to local deformations. The distributions are computed over different instances of the same class, but unlike our approach, the posteriors from different trees and ferns are combined by averaging. The results match our own observations that using either fern or tree structures leads to similar classification performance.

3   A SEMINAIVE BAYESIAN APPROACH TO PATCH RECOGNITION

As discussed in Section 2, image patches can be recognized on the basis of very simple and randomly chosen binary tests that are grouped into decision trees and recursively partition the space of all possible patches [20]. In essence, we treat the set of possible appearances of a keypoint as classes and the RTs embody a probability distribution over these classes. In practice, no single tree is discriminative enough when there are many classes. However, using a number of trees and averaging their votes yields good results because each one partitions the feature space in a different way.

In this section, we will argue that, when the tests are chosen randomly, the power of the approach derives not from the tree structure itself but from the fact that combining groups of binary tests allows improved classification rates. Therefore, replacing the trees by our nonhierarchical ferns and pooling their answers in a naive Bayesian manner yields better results and scalability in terms of number of classes. As a result, we can combine many more features, which is the key to improved recognition rates.

We first show that our nonhierarchical ferns fit nicely into a naive Bayesian framework and then explain the training protocol which is similar to the one used for RTs.

3.1   Formulation of Feature Combination

As discussed in Section 2, we treat the set of all possible appearances of the image patch surrounding a keypoint as a class. Therefore, given the patch surrounding a keypoint detected in an image, our task is to assign it to the most likely class. Let c_i, $i = 1, \ldots, H$, be the set of classes and let f_j, $j = 1, \ldots, N$, be the set of binary features that will be calculated over the patch that we are trying to classify. Formally, we are looking for

$$\hat{c}_i = \underset{c_i}{\mathrm{argmax}} \; P(C = c_i \mid f_1, f_2, \ldots, f_N),$$

where C is a random variable that represents the class. Bayes' formula yields

$$P(C = c_i \mid f_1, f_2, \ldots, f_N)$$
$$= \frac{P(f_1, f_2, \ldots, f_N \mid C = c_i) P(C = c_i)}{P(f_1, f_2, \ldots, f_N)}.$$

Assuming a uniform prior $P(C)$, since the denominator is simply a scaling factor that is independent from the class, our problem reduces to finding

$$\hat{c}_i = \underset{c_i}{\mathrm{argmax}} \, P(f_1, f_2, \ldots, f_N \mid C = c_i). \qquad (1)$$

In our implementation, the value of each binary feature f_j only depends on the intensities of two pixel locations $\mathbf{d}_{j,1}$ and $\mathbf{d}_{j,2}$ of the image patch. We therefore write

$$f_j = \begin{cases} 1, & \text{if } I(\mathbf{d}_{j,1}) < I(\mathbf{d}_{j,2}), \\ 0, & \text{otherwise}, \end{cases}$$

where I represents the image patch. Since these features are very simple, we require many ($N \approx 300$) for accurate classification. Therefore, a complete representation of the joint probability in (1) is not feasible since it would require estimating and storing 2^N entries for each class. One way to compress the representation is to assume independence between features. An extreme version is to assume complete independence, that is,

$$P(f_1, f_2, \ldots, f_N \mid C = c_i) = \prod_{j=1}^{N} P(f_j \mid C = c_i).$$

However, this completely ignores the correlation between features. To make the problem tractable while accounting for these dependencies, a good compromise is to partition our features into M groups of size $S = \frac{N}{M}$. These groups are what we define as *ferns* and we compute the joint probability for features in each fern. The conditional probability becomes

$$P(f_1, f_2, \ldots, f_N \mid C = c_i) = \prod_{k=1}^{M} P(F_k \mid C = c_i), \qquad (2)$$

where $F_k = \{f_{\sigma(k,1)}, f_{\sigma(k,2)}, \ldots, f_{\sigma(k,S)}\}$, $k = 1, \ldots, M$, represents the kth fern and $\sigma(k, j)$ is a random permutation function with range $1, \ldots, N$. Hence, we follow a Seminaive Bayesian [36] approach by modeling only some of the dependencies between features. The viability of such an approach has been shown by [18] in the context of image retrieval applications.

This formulation yields a tractable problem that involves $M \times 2^S$ parameters, with M between 30 and 50. In practice, as will be shown in Section 4, $S = 11$ yields good results. Therefore, $M \times 2^S$ is of the order of 80,000, which is much smaller than 2^N with $N \approx 450$ that the full joint probability representation would require. Our formulation is also flexible since performance/memory trade-offs can be made by changing the number of ferns and their sizes.

Note that we use randomization not only in feature selection but also in grouping. An alternative approach would involve selecting feature groups to be as independent from each other as possible. This is routinely done by semi-naive Bayesian classifiers based on a criteria such as the mutual information between features [8]. However, in practice, we have not found this to be necessary to achieve good performance. We have, therefore, chosen not to use such a strategy to preserve the simplicity and efficiency of our training scheme and to allow for incremental training.

3.2 Training

We assume that at least one image of the object to be detected is available for training. We call any such image as a *model* image. Training starts by selecting a subset of the keypoints detected on these model images. This is done by deforming the images many times, applying the keypoint detector, and keeping track of the number of times the same keypoint is detected. The keypoints that are found most often are assumed to be the most stable and retained. These stable keypoints are assigned a unique class number.

The training set for each class is formed by generating 10,000 sample images with randomly picked affine deformations by sampling the deformation parameters from a uniform distribution, adding Gaussian noise to each sample image, and smoothing with a Gaussian filter of size 7×7. This increases the robustness of the resulting classifier to runtime noise, especially when there are features that compare two pixels on a uniform area.

The training phase estimates the class-conditional probabilities $P(F_m \mid C = c_i)$ for each fern F_m and class c_i, as described in (2). For each fern F_m, we write these terms as:

$$p_{k,c_i} = P(F_m = k \mid C = c_i), \qquad (3)$$

where we simplify our notations by considering F_m to be equal to k if the base 2 number formed by the binary features of F_m taken in sequence is equal to k. With this convention, ferns can take $K = 2^S$ values, and for each one, we need to estimate the p_{k,c_i}, $k = 1, 2, \ldots, K$, under the constraint

$$\sum_{k=1}^{K} p_{k,c_i} = 1.$$

The simplest approach would be to assign the maximum likelihood estimate to these parameters from the training samples. For parameter p_{k,c_i}, it is

$$p_{k,c_i} = \frac{N_{k,c_i}}{N_{c_i}},$$

where N_{k,c_i} is the number of training samples of class c_i that evaluates to fern value k and N_{c_i} is the total number of
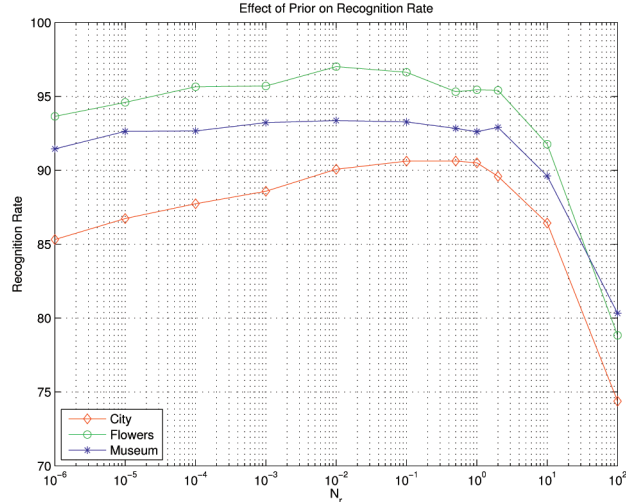

Fig. 3. Recognition rate as a function of $\log(N_r)$ using the three test images of Section 4. The recognition rate remains relatively constant for $0.001 < N_r < 2$. For $N_r < 0.001$, it begins a slow decline, which ends in a sudden drop to about 50 percent when $N_r = 0$. The rate also drops when N_r is too large because too strong a prior decreases the effect of the actual training data, which are around 10,000 samples for this experiment.

samples for class c_i. These parameters can therefore be estimated for each Fern independently.

In practice, however, this simple scheme yields poor results because if no training sample for class c_i evaluates to k, which can easily happen when the number of samples is not infinitely large, both N_{k,c_i} and p_{k,c_i} will be zero. Since we multiply the p_{k,c_j} for all ferns, it implies that, if the fern evaluates to k, the corresponding patch can *never* be associated to class c_i, no matter the response of the other ferns. This makes the ferns far too selective because of the fact that $p_{k,c_i} = 0$ may simply be an artifact of the necessarily limited size of the training set. To overcome this problem, we take p_{k,c_i} to be

$$p_{k,c_i} = \frac{N_{k,c_i} + N_r}{N_{c_i} + K \times N_r},$$

where N_r represents a regularization term, which behaves as a uniform Dirichlet prior [4] over feature values. If a sample with a specific fern value is not encountered during training, this scheme will still assign a nonzero value to the corresponding probability. As illustrated by Fig. 3, we have found our estimator to be insensitive to the exact value of N_r and we use $N_r = 1$ in all our experiments. However, it is strictly essential to have N_r greater than zero. This tallies with the observation that combining classifiers in a naive Bayesian fashion can be unreliable if improperly done [19].

In effect, our training scheme marginalizes over the pose space since the class-conditional probabilities $P(F_m \mid C = c_i)$ depend on the camera poses relative to the object. By densely sampling the pose space and summing over all samples, we marginalize over these pose parameters. Hence, at runtime, the statistics can be used in a pose-independent manner, which is the key to real-time performance. Furthermore, the training algorithm itself is very efficient since it only requires storing the N_{k,c_i} counts
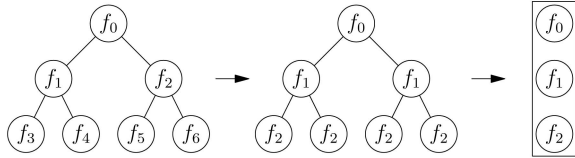
Fig. 4. Ferns versus trees. A tree can be transformed into a fern by performing the following steps. First, we constrain the tree to systematically perform the same test across any given hierarchy level, which results in the same feature being evaluated independently of the path taken to get to a particular node. Second, we do away with the hiearchical structure and simply store the feature values at each level. This means applying a sequence of tests to the patch, which is what ferns do.

for each fern while discarding the training samples immediately after use, which means that we can use arbitrarily many if need be.

4   COMPARISON WITH RANDOMIZED TREES

As shown in Figs. 4 and 5, Ferns can be considered as simplified trees. Whether or not this simplification degrades their classification performance hinges on whether our randomly chosen binary features are still appropriate in this context. In this section, we will show that they are indeed. In fact, because our naive Bayesian scheme outperforms the averaging of posteriors used to combine the output of the decision trees [20], the ferns are both simpler and more powerful.

To compare RTs and ferns, we experimented with the three images of Fig. 6. We warp each image by repeatedly applying random affine deformations and detect Harris corners in the deformed images. We then select the most stable 250 keypoints per image based on how many times they are detected in the deformed versions to use in the following experiments and assign a unique class id to each of them. The classification is done using patches that are 32×32 pixels in size.

Ferns differ from trees in two important respects: The probabilities are multiplied in a naive Bayesian way instead of being averaged and the hierarchical structure is replaced by a flat one. To disentangle the influence of these changes, we consider four different scenarios:
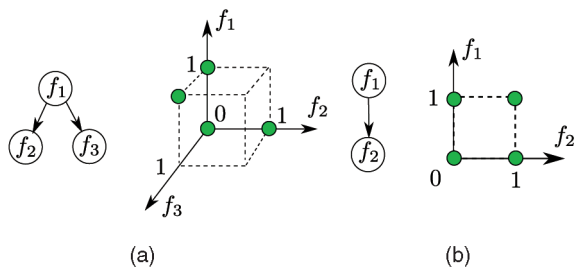

Fig. 5. The feature spaces of trees and ferns. Although the space of tree features seems much higher dimensional, it is not because only a subset of features can be evaluated. (a) For the simple tree on the left, only the four combinations of feature values denoted by green circles are possible. (b) The even simpler fern on the left also yields four possible combinations of feature values but a much simpler structure. As shown below, this simplicity does not entail any performance loss.

Fig. 6. The recognition rate experiments are performed on three images that show different texture and structures. Image size is 640×480 pixels.

- using randomized trees and averaging of class posterior distributions, as in [20];
- using randomized trees and combining class-conditional distributions in a naive Bayesian way;
- using ferns and averaging of class posteriors;
- using ferns and combining class-conditional distributions in a naive Bayesian way, as we advocate in this paper.

The trees are of depth 11 and each fern has 11 features, yielding the same number of parameters for the estimated distributions. Also, the number of features evaluated per patch is equal in all cases.

The training set is obtained by randomly deforming images of Fig. 6. To perform these experiments, we represent affine image deformations as 2×2 matrices of the form

$$R_\theta R_{-\phi} \mathrm{diag}(\lambda_1, \lambda_2) R_\phi,$$

where $\mathrm{diag}(\lambda_1, \lambda_2)$ is a diagonal 2×2 matrix and R_γ represents a rotation of angle γ. Both to train and to test our ferns, we warped the original images using such deformations computed by randomly choosing θ and ϕ in the $[0{:}2\pi]$ range and λ_1 and λ_2 in the $[0.6{:}1.5]$ range. Fig. 7 depicts patches surrounding individual interest points first in the original images and then in the warped ones. We
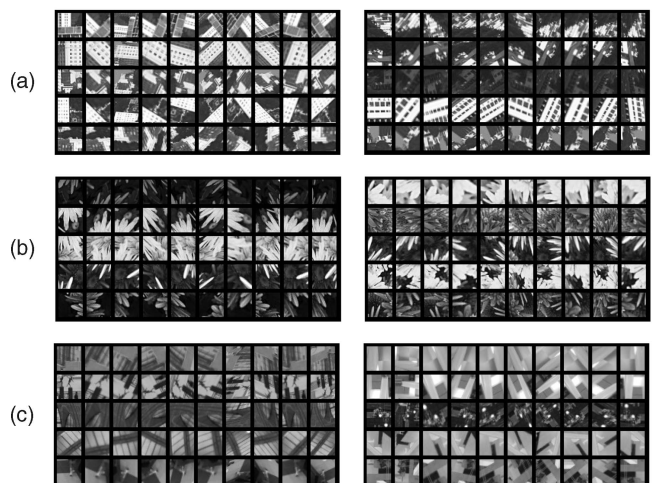

Fig. 7. Warped patches from the images of Fig. 6 show the range of affine deformations that we considered. In each line, the leftmost patch is the original one and the others are deformed versions of it. (a) Sample patches from the *City* image. (b) Sample patches from the *Flowers* image. (c) Sample patches from the *Museum* image.
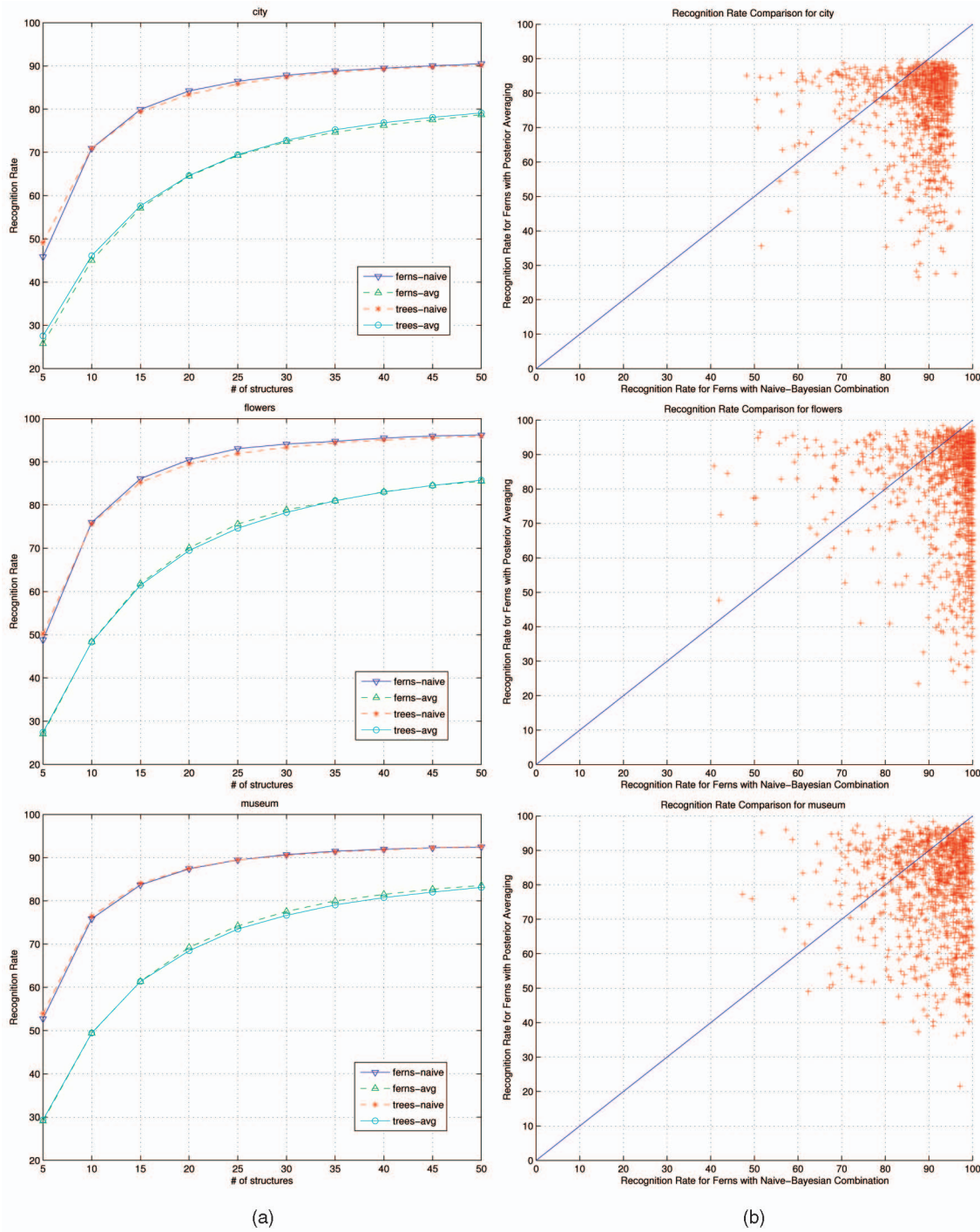
Fig. 8. (a) The average percentage of correctly classified image patches over many trials is shown as the number of trees or ferns is changed. Using the naive Bayesian assumption gives much better rates at reduced number of structures, while the fern and tree structures are interchangeable. (b) The scatter plots show the recognition rate over individual trials with 50 ferns. The recognition rates for the naive Bayesian combination and posterior averaging are given in the x and y axes, respectively. The naive Bayesian combination of features usually performs better, as evidenced by the fact that most points are below the diagonal, and only very rarely produce recognition rates below 80 percent. By contrast, the averaging produces rates below 60 percent.

used 30 random affine deformations per degree of rotation to produce 10,800 images. As explained in Section 3.2, we then added Gaussian noise with zero mean and a large variance—25 for gray levels ranging from 0 to 255—to these warped images to increase the robustness of the resulting ferns. Gaussian smoothing with a mask of 7×7 is applied to both training and test images.

The test set is obtained by generating a separate set of 1,000 images in the same affine deformation range and adding noise. Note that we simply transform the original keypoint locations and therefore we ignore the keypoint detector's repeatability in the tests and measure only the recognition performance. In Fig. 8, we plot the results as a function of the number of trees or ferns being used.

TABLE 1
Variance of the Recognition Rate

Number of Structures	10	15	20	25	30	35	40	45	50
Fern-Naive	0.59	0.24	0.11	0.06	0.05	0.07	0.06	0.08	0.05
Fern-Avg	0.33	0.21	0.29	0.24	0.35	0.39	0.35	0.41	0.30
Tree-Naive	0.37	0.19	0.13	0.07	0.06	0.04	0.04	0.02	0.01
Tree-Avg	0.30	0.22	0.13	0.17	0.16	0.17	0.15	0.17	0.20

We first note that using either flat Fern or hierarchical tree structures does not affect the recognition rate, which was to be expected as the features are taken completely at random. By contrast, the naive Bayesian combination strategy outperforms the averaging of posteriors and achieves a higher recognition rate even when using relatively few structures. Furthermore, as the scatter plots of Fig. 8 show, for the naive Bayesian combination, the recognition rate on individual deformed images never falls below an acceptable rate. Since the features are taken randomly, the recognition rate changes and the variance of the recognition rate is given as Table 1. As more ferns or trees are used, the variance decreases and more rapidly for the naive combination. If the number of ferns or trees is below 10, the recognition rate starts to change more erratically and entropy-based optimization of feature selection becomes a necessity.

To test the behavior of the methods as the number of classes is increased, we have trained classifiers for matching up to 1,500 classes. Fig. 9 shows that the performance of the naive Bayesian combination does not degrade rapidly and scales much better than averaging posteriors. For both methods, the required amounts of memory and computation times increase linearly with the number of classes since we assign a separate class for each keypoint.

So far, we have used 11 features for each fern, and we now discuss the influence of this number on recognition performance and memory requirements.

Increasing the fern size by one doubles the number of parameters, and hence, the memory required to store the distributions. It also implies that more training samples should be used to estimate the increased number of parameters. It has, however, negligible effect on the runtime speed and larger ferns can, therefore, handle more variation at the cost of training time and memory but without much of a slowdown.

By contrast, adding more ferns to the classifier requires not only a linear increase in memory but also in computation time. Since the training samples for other ferns can be reused, it only has a negligible effect on training time. As shown in Fig. 10, for a given amount of memory, the best recognition rate is obtained by using many relatively small ferns. However, this comes at the expense of runtime speed, and when sufficient memory is available, a fern size of 11 represents a good compromise, which is why we have used this value in the experiments.

Finally, we evaluate the behavior of the classifier as a function of the number of training samples. Initially, we use 180 training images that we generate by warping the images of Fig. 6 by random scaling parameters and deformation angle ϕ, while the rotation angle θ is uniformly sampled at every two degrees. We then increase the number of training samples by 180 at each step. The graphs depicted in Fig. 11 show that the naive Bayesian combination performs consistently better than the averaging, even when only a small number of training samples are used.

5 EXPERIMENTS

We evaluate the performance of fern-based classification for both planar and fully 3D object detection and SLAM-based camera tracking. We also compare our approach against SIFT [22], which is among the most reliable descriptors for patch matching.
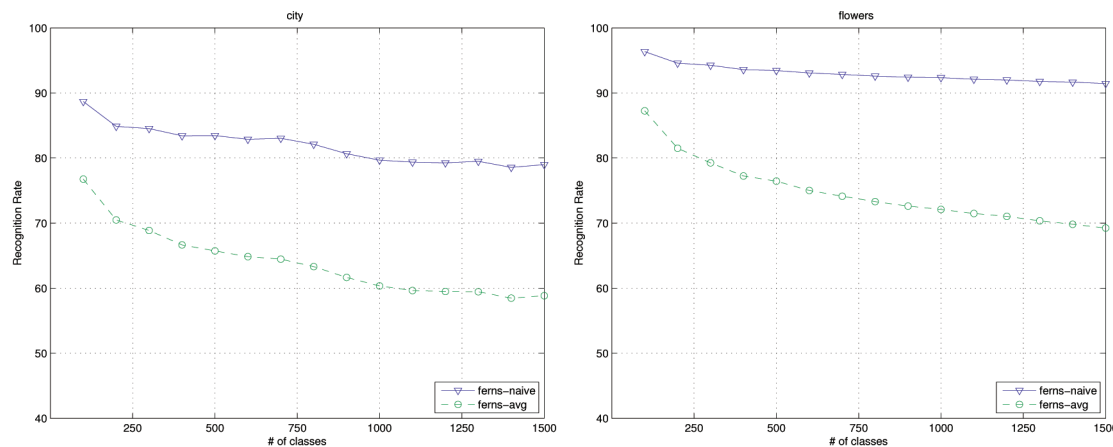

Fig. 9. Recognition rate as a function of the number of classes. While the naive combination produces a very slow decrease in performance, posterior averaging exhibits a much sharper drop. The tests are performed on the high-resolution versions of the *City* and *Flowers* data, respectively.
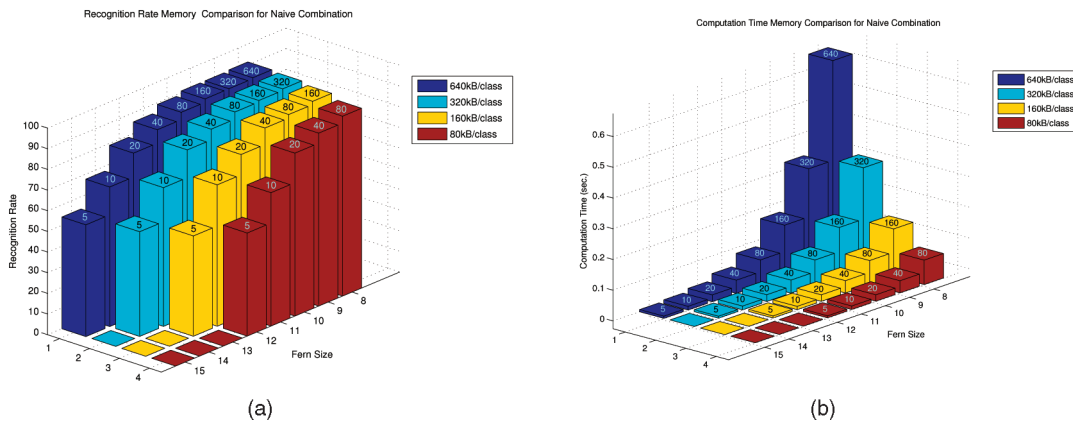
Fig. 10. (a) Recognition rate and (b) computation time as a function of the amount of memory available and the size of the ferns being used. The number of ferns used is indicated on the top of each bar and the y-axis shows the fern size. The color of the bar represents the required memory amount, which is computed for distributions stored with single-precision floating numbers. Note that while using many small ferns achieves higher recognition rates, it also entails a higher computational cost.

5.1 Ferns versus SIFT to Detect Planar Objects

We used the 1,074-frame video depicted in Fig. 1 to compare ferns against SIFT for planar object detection. It shows a mouse pad undergoing motions involving a large range of rotations, scalings, and perspective deformations against a cluttered background. We used as a reference an image in which the mouse pad is seen frontally. We match the keypoints extracted from each input image against those found in the reference image using either ferns or SIFT and then eliminate outliers by computing a homography using RANSAC.

The SIFT keypoints and the corresponding descriptors are computed using the publicly available code kindly provided by Lowe [21]. The keypoint detection is based on the difference of Gaussians over several scales, and for each keypoint, dominant orientations are precomputed. By contrast, the ferns rely on a simpler keypoint detector that computes the maxima of Laplacian on three scales, which provides neither dominant orientation information nor a finely estimated scale. We retain the 400 strongest keypoints in the reference, and 1,000 keypoints in the input images for the two methods.

We train 20 ferns of size 14 to establish matches with the keypoints on the video frame by selecting the most probable class. In parallel, we match the SIFT descriptors for the keypoints on the reference image against the keypoints on the input image by selecting the one which has the nearest SIFT descriptor. Given the matches between the reference and input image, we use a robust estimation followed by nonlinear refinement to estimate a homography. We then take all matches with reprojection error less than 10 pixels to be inliers. Fig. 1 shows that the ferns can match as many points as SIFT, and sometimes, even more.

It is difficult to perform a completely fair speed comparison between our ferns and SIFT for several reasons. SIFT reuses intermediate data from the keypoint extraction to compute canonic scale and orientations and the descriptors, while ferns can rely on a low-cost keypoint extraction. On the other hand, the distributed SIFT C code is not optimized, and the Best-Bin-First KD-tree of [3] is not used to speed up the nearest-neighbor search.

However, it is relatively easy to see that our approach requires much less computation. Performing the individual tests of Section 3 requires very little time and most of the time
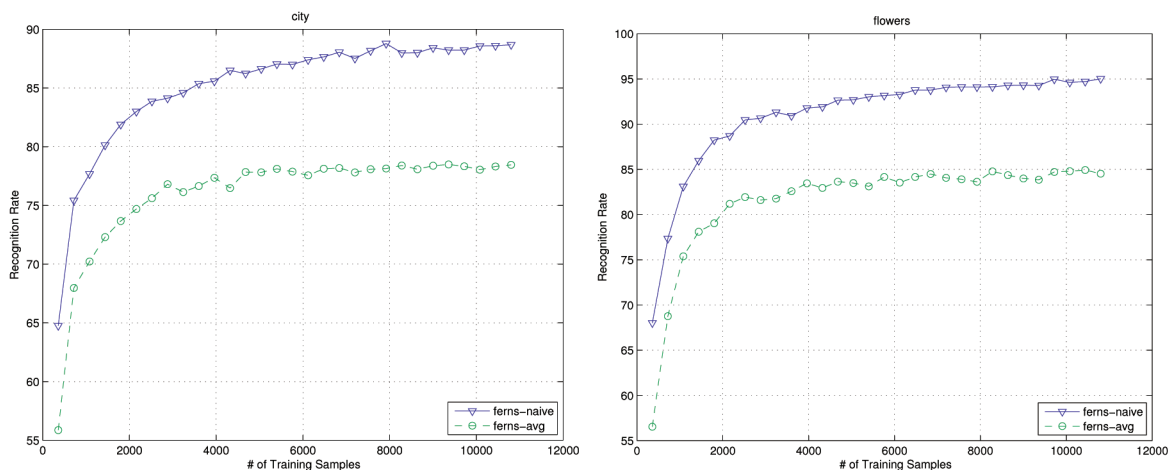

Fig. 11. Recognition rate as a function of the number of training samples for each one of the three images of Fig. 6. As more training samples are used, the recognition rate increases and it does so faster for the naive Bayesian combination of ferns.
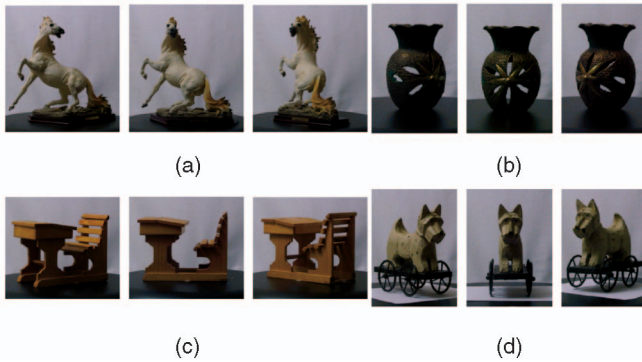
Fig. 12. When detecting a 3D object, viewpoint change is more challenging due to self-occlusions and nontrivial lighting effects. The images are taken from a database presented in [26] and cover a total range of 70 degrees of camera rotation. They are cropped around the object, while we used the original images in the experiments. (a) *Horse* data set. (b) *Vase* data set. (c) *Desk* data set. (d) *Dog* data set.
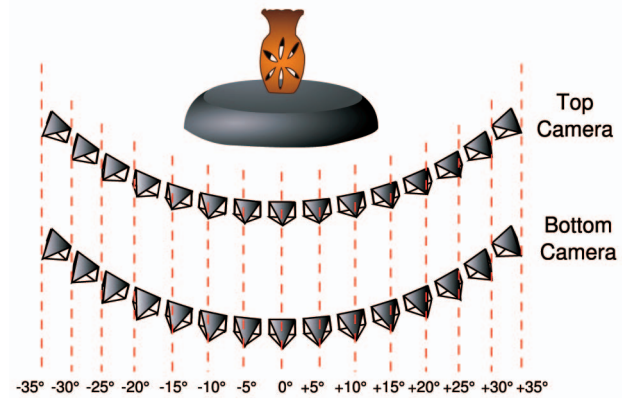

Fig. 13. Generating ground truth data for 3D object detection. Each test object contains two sequences of images taken by the *Top* and *Bottom* cameras while the object rotates on the turntable. The camera geometry has been calibrated using a checkerboard calibration pattern. We use 15 consecutive camera views for evaluation purposes, because it is easy to obtain high-quality calibration for this range of rotation.

is spent computing the sums of the posterior probabilities. The classification of a keypoint requires $H \times M$ additions, with H the number of classes and M the number of ferns. In contrast, SIFT uses $128\,H$ additions and as many multiplications when the Best-Bin-First KD-tree is not used. This represents an obvious advantage of our method at runtime since M can be much less than 128, and is taken to be 20 in practice, while selecting a large number of features for each fern and using tens of thousands of training samples.

The major gain actually comes from the fact that ferns do not require descriptors. This is significant because computing the SIFT descriptors, which is the most difficult part to optimize, takes about 1 ms on a MacBook Pro laptop without including the time required to convolve the image. In contrast, ferns take 13.5×10^{-3} milliseconds to classify one keypoint into 200 classes on the same machine. Moreover, ferns still run nicely with a primitive keypoint extractor, such as the one we used in our experiments. When 300 keypoints are extracted and matched against 200 classes, our implementation on the MacBook Pro laptop requires 20 ms per frame for both keypoint extraction and recognition in 640×480 images, and four-fifths of this time is devoted to keypoint extraction. This corresponds to a theoretical 50 Hz frame rate if one does ignore the time required for frame acquisition. Training takes less than 5 minutes.

Of course, the ability to classify keypoints fast and work with a simple keypoint detector comes at the cost of requiring a training stage, which is usually offline. In contrast, SIFT does not require training and, for some applications, this is clearly an advantage. However, for other applications, ferns offer greater flexibility by allowing us to precisely state the kind of invariance we require through the choice of the training samples. Ferns also let us incrementally update the classifiers as more training samples become available as demonstrated by the SLAM application of Section 5.4. This flexibility is key to the ability to carry out offline computations and significantly simplify and speed up the runtime operation.

5.2 Ferns versus SIFT to Detect 3D Objects

So far, we have considered that the keypoints lie on a planar object and evaluated the robustness of ferns with respect to

perspective effects. This simplifies training as a single view is sufficient and the known 2D geometry can be used to compute ground truth correspondences. However, most objects have truly 3D appearance, which implies that self-occlusions and complex illuminations effects have to be taken into account to correctly evaluate the performance of any keypoint matching algorithm.

Recently, an extensive comparison of different keypoint detection and matching algorithms on a large database of 3D objects has been published [26]. It was performed on images taken by a stereo camera pair of objects rotating on a turntable. Fig. 12 shows such images spanning a 70 degree camera rotation range. We used this image database to evaluate the performance of ferns for a variety of 3D objects. We compare our results against the SIFT detector/descriptor pair which has been found to perform very well on this database. The keypoints and the descriptors are computed using the same software as before [21].

As in [26], we obtained the ground truth by using purely geometric methods, which is possible because the cameras and the turn table are calibrated. The initial correspondences are obtained by using the trifocal geometry between the top/bottom cameras in the center view and every other camera, as illustrated by Fig. 13. We then reconstruct the 3D points for each such correspondence in the bottom/center camera coordinate frame and use these to form the initial tracks that span the -35 degree$/+35$ degree rotation range around a central view. Since the database images are separated by 5 degrees, the tracks span 15 images each for the top and bottom cameras. We eliminate very short tracks and remaining tracks are extended by projecting the 3D point to each image and searching for a keypoint in the vicinity of the projection. Finally, to increase robustness against spurious tracks formed by outliers, we eliminate tracks covering less than 30 percent of the views and the remaining tracks form the ground truth for the evaluation, which is almost free of outliers. Sample ground truth data are depicted in Fig. 14, which shows the complex variations in patch appearance induced by the 3D structure of the objects.

The training is done using views separated by 10 degrees, skipping every other frame in the ground truth data. We
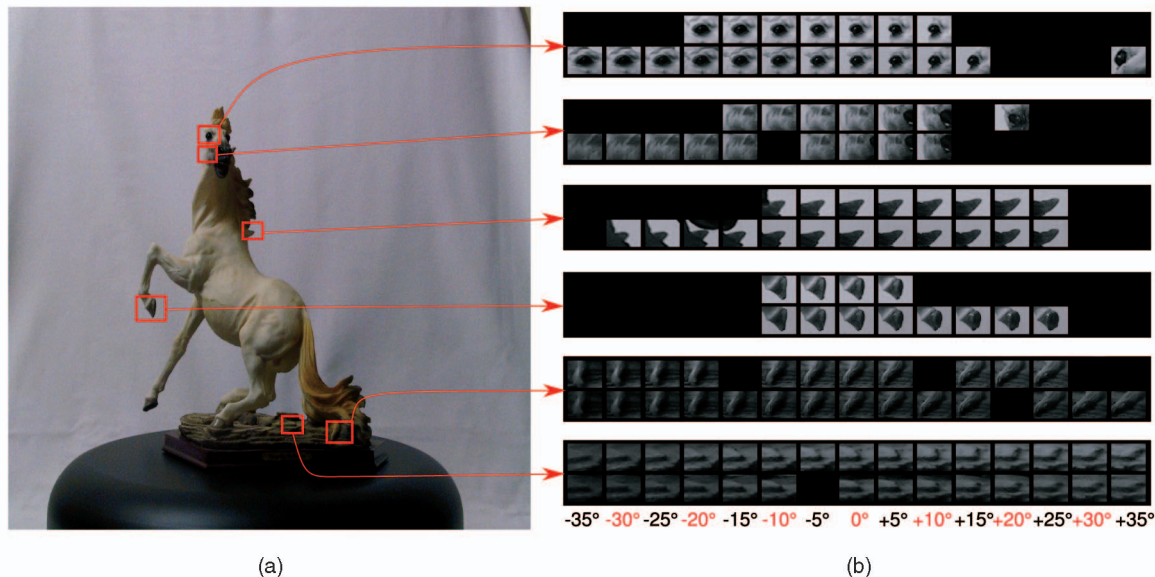
Fig. 14. Samples from the ground truth for the *Horse* data set. The image in (a) shows the center view. The six keypoint tracks in (b) show the content variation for each patch as the camera center rotates around the turntable center. Each track contains two lines that correspond to the *Top* and *Bottom* cameras, respectively. Black areas denote frames for which the keypoint detector did not respond. The views produced by a rotation that is a multiple of 10 degrees are used for training and are denoted by red labels. The others are used for testing.

then use the views that we skipped for testing purposes. This geometry-based sampling is shown in Fig. 14. This sampling creates uneven number of training and test samples for different keypoints as there are gaps in the tracks. The sampling could have been done differently to balance the number of test and training samples for each keypoint. However, our approach to sampling closely mimics what happens in practice when training data come from sparse views and the classifier must account for unequal numbers of training samples.

We train the ferns in virtually the same way we do in the planar case. Each track is assigned a class number and the training images are deformed by applying random affine deformations. We then use all of them to estimate the probability distributions, as discussed in Section 3; 1,000 random affine deformations per training image are used to train 50 ferns of size 11. Ferns classify the test patches by selecting the track with the maximum probability. For SIFT, each test example is classified by selecting the track number of the keypoint in the training set with the nearest SIFT descriptor.

In our tests, we learn the appearance and the geometry of a 3D object from several views and then detect it in new ones by matching keypoints. Hence, the learned geometry can be used to eliminate outliers while estimating the camera pose using the P3P algorithm [16] together with a robust matching strategy such as RANSAC [14]. Unlike [26], we, therefore, do not use the ratio test on descriptor distances or a similar heuristics to reject matches, as this might reject correct correspondences. The additional computational burden can easily be overcome by using the classification score for RANSAC sampling, as presented by [9].

We compare the recognition rates of both methods on objects with different kinds of texture. Fig. 15 shows the recognition rate on each test image together with the average over all frames. The ferns perform as well as

nearest-neighbor matching with SIFT for a whole range of objects with very different textures.

Note that, when using ferns, there is almost no runtime speed penalty for using multiple frames, since as more training frames are added we can increase the size of our ferns. As discussed in Section 4, this requires more memory but does not slow down the algorithm in any meaningful way. In contrast, using more frames for nearest-neighbor SIFT matching linearly slows down the matching, although a clever and approximate implementation might mitigate the problem.

In theory, it should be possible to improve the performance of SIFT-based approach by replacing nearest-neighbor matching with a more sophisticated technique such as K-Nearest Neighbors with voting. However, this would further slow down the algorithm. Our purpose is to show that the fern-based classifier can naturally integrate data from multiple images without the need for a more complex training phase or any handicap in the runtime performance, reaching the performance of standard SIFT matching.

5.3 Panorama and 3D Scene Annotation

With the recent proliferation of mobile devices with significant processing power, there has been a surge of interest in building real-world applications that can automatically annotate the photographs and provide useful information about places of interest. These applications test keypoint matching algorithms to their limits because they must operate under constantly changing lighting conditions and potentially changing scene texture, both of which reduce the number of reliable keypoints. We have tested ferns on two such applications, annotation of panorama scenes and parts of a historical building with 3D structure. Both applications run smoothly at frame rate using a standard laptop and an off the shelf Web camera. By applying standard optimizations for embedded hardware,
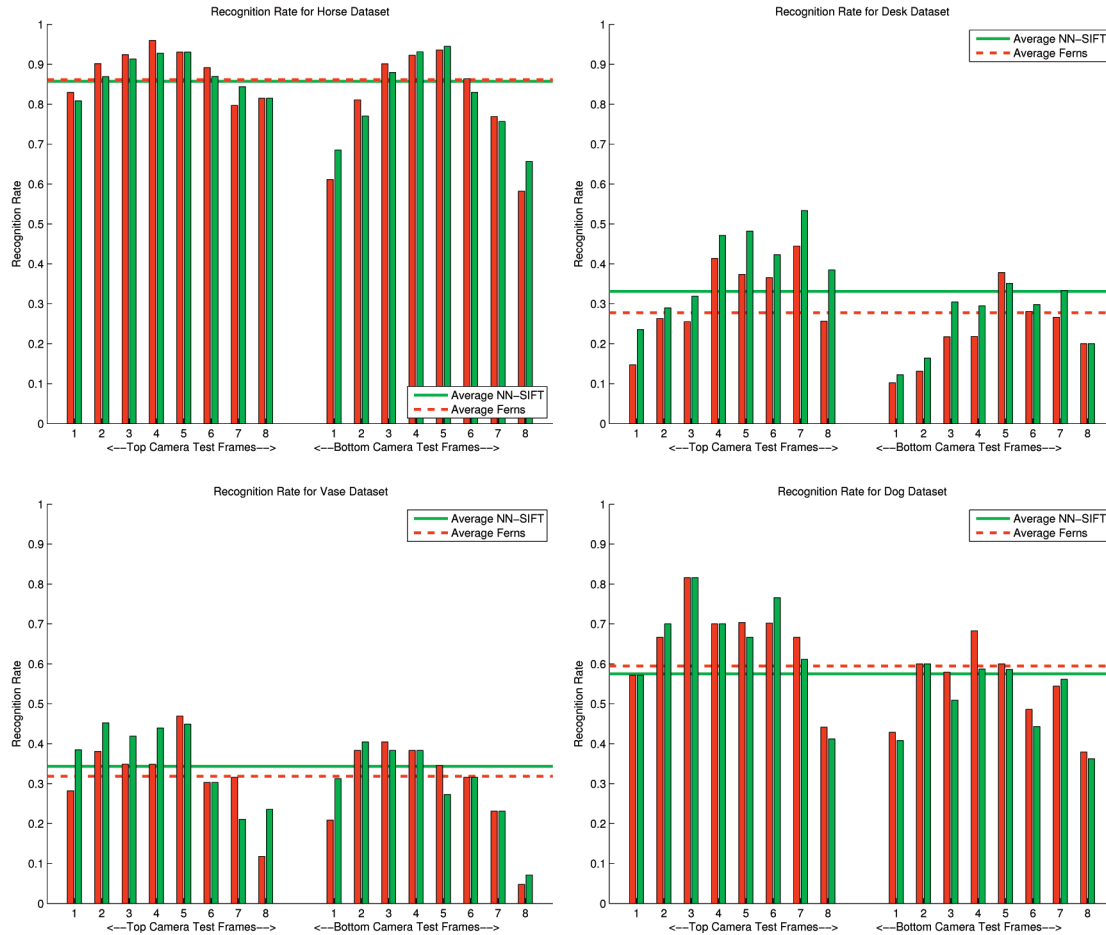
Fig. 15. Recognition rates for 3D objects. Each pair of bars corresponds to a test frame. The red bar on the left represents the rate for ferns and the light green bar on the right the rate for Nearest-Neighbor SIFT matching. The weighted averages over all frames also appear as a dashed line for ferns and solid line for NN-SIFT. The weights we use are the number of keypoints per frame.

we have ported this implementation onto a mobile device that runs at a few frames per second. More recently, ferns have been successfully integrated into commercially available mobile phones to run at frame rates by taking into account specific limitations of the hardware and integrating detection with frame-to-frame tracking [34].

For the panorama application, we trained Ferns using an annotated panorama image stitched from multiple images. At runtime, given an input image and after having established correspondences between the panorama and the test image, we compute a 2D homography and use it to eliminate outliers and to transfer the annotation from the training image to the input image, as shown in Fig. 16. We successfully run a number of tests under different weather conditions and different times of day.

Annotating a 3D object requires training using multiple images from different viewpoints, which is easy to do in our framework, as discussed in the previous section. We also built a 3D model for the object using standard structure from motion algorithms to register the training images followed by dense reconstruction [33], [32]. The resulting fine mesh is too detailed to be used, so it is approximated by a coarse one containing much less detail. Despite its rough structure, this 3D model allows annotation of important parts of the object and the correct reprojection of this

information onto images taken from arbitrary viewpoints, as depicted in Fig. 16.

5.4 SLAM Using Ferns

In this section, we demonstrate that ferns can increase the robustness of a Visual SLAM system. Their role is twofold. We first use them to bootstrap the system by localizing the camera with respect to a known planar pattern in the scene. Second, we incrementally train a second set of Ferns to recognize new landmarks reconstructed by the system. They make the system robust against severe disturbances such as complete occlusion or strong shaking of the camera, as evidenced by the smoothness of the recovered camera trajectory in Fig. 18a.

For simplicity, we use a FastSLAM [24], [25] approach with a single particle to model the distribution over the camera trajectory. This is therefore a simplified version of FastSLAM, but the ferns are sufficiently powerful to make it robust.

As discussed above, we use two different sets of ferns. We will refer to the first set as "Offline Ferns" that we trained offline to recognize keypoints on the planar pattern. As shown in Fig. 17a, we make sure it is visible in the first frame of the sequence to bootstrap the system and replace the four fiducial markers used by many other systems. This
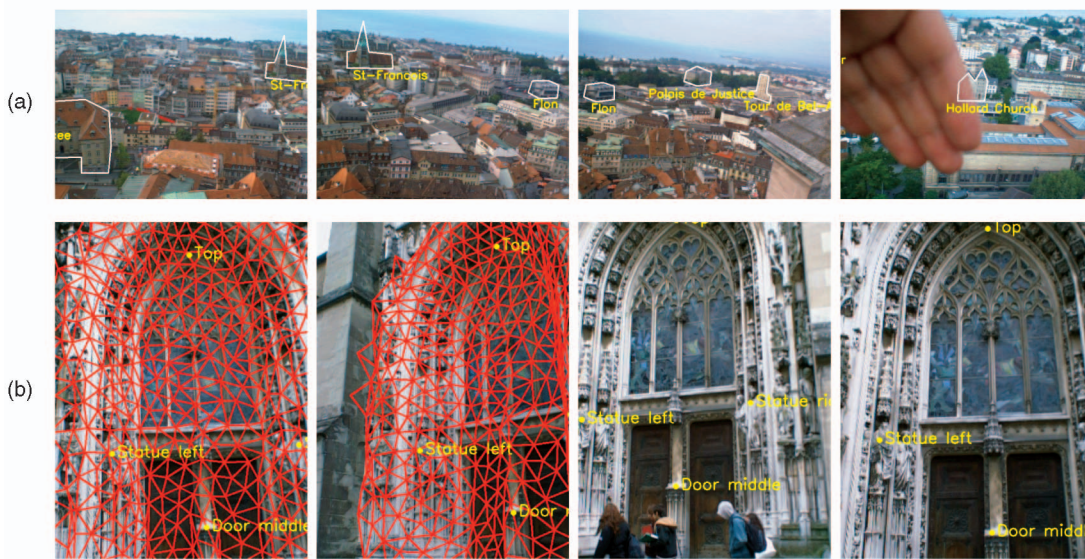
Fig. 16. Automated image annotation. (a) We match an input image against a panorama. Despite occlusions, changes in weather conditions, and lighting, the ferns return enough matches for reliable annotation of major landmarks in the city. (b) We match an image to an annotated 3D model, overlaid on the two leftmost images. This lets us display annotations at the right place.

increases the flexibility of our system since we can use any pattern we want provided that it is textured enough. The second set of ferns, the "Online Ferns," are incrementally trained to recognize the 3D landmarks the SLAM system discovers and reconstructs.

The incremental training takes place over several frames, typically between 20 and 50, where the corresponding
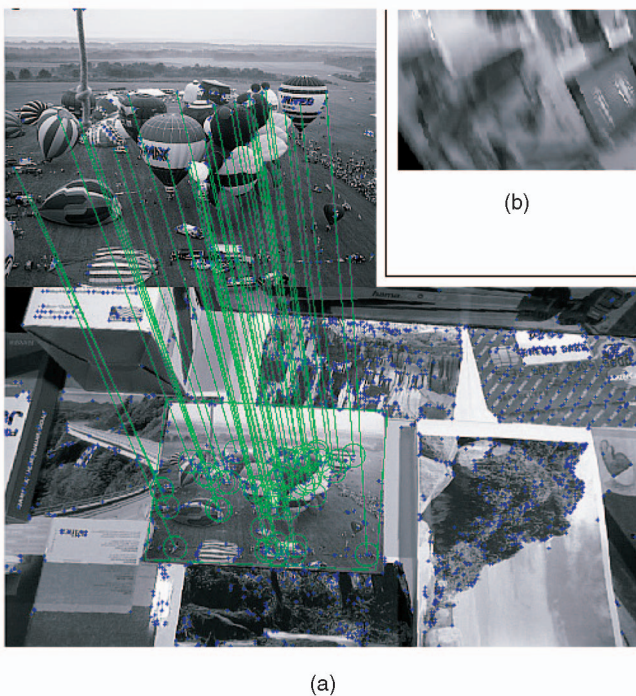

Fig. 17. Ferns applied to SLAM. (a) Initialization: The pattern at the top is detected in the image yielding initial landmarks and pose. (b) A later image from the sequence, which shows both strong camera shaking and motion blur. When the image is that bad, no correspondences can be established and the tracking is lost. Nevertheless, the system automatically recovers when the image quality improves due to the ferns.

landmark was matched. For each, we add a small number of random views of the observed patch to the ferns. In the case where a landmark is observed for the first time, this number is substantially higher, around 100. The comparatively small number of views in the beginning works because motion between frames is relatively small, and by the time the viewing angle has changed significantly, the training will be complete. Of course, not all of the patches need to be retrained, but only the new ones. This is achieved by undoing the normalization step, adding the new observations, and then renormalizing the posteriors. This incremental training is computationally costly and future research will aim at reducing the cost of training [7].

Our algorithm goes through the following steps:

1. Initialize the camera pose and some landmarks by detecting a known pattern using the Offline Ferns.
2. Detect keypoints and match them using the Offline and Online Ferns against the known landmarks.
3. Estimate the camera pose from these correspondences using a P3P algorithm and RANSAC [14]. The estimated pose is refined via a nonlinear optimization.
4. Refine the location estimates of the inlier landmarks using an extended Kalman filter.
5. Create new landmarks. Choose a number of detected keypoints that *do not* belong to any landmark in the map and initialize the new landmarks with a large uncertainty along the line of sight and a much smaller uncertainty in the camera's lateral directions.
6. Retrain ferns with good matches from step 2 and the new landmarks.
7. Loop to step 2.

With this system, we demonstrate that both smooth tracking and recovery from complete failure can be naturally integrated by employing ferns for the matching task.

The reconstructed trajectory in Fig. 18 shows only tiny jags in the order of a few millimeters and appears as smooth
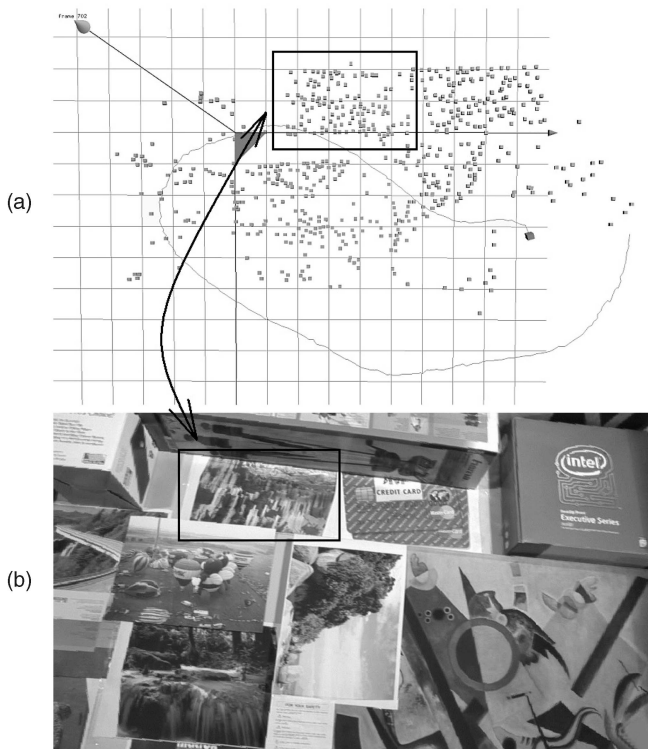
Fig. 18. Ferns applied to SLAM. (a) View from the top on the xy-plane of the 3D space in which the trajectory and landmarks are reconstructed. The landmarks are shown as cubes centered around the current mean value of the corresponding Kalman filter. Note how smooth the trajectory is, given the difficulties the algorithm had to face. (b) Qualitative visual comparison: One group of 3D points and the corresponding group of 2D points are marked. Comparing this group to the surroundings confirms a basic structural correctness of the reconstruction.

as a trajectory that was estimated in a filtered approach to SLAM, such as MonoSLAM [11], [10]. This is especially noteworthy as the camera's state is reestimated from scratch *in every frame* and there is no such thing as a motion model.[2] At the same time, this is a strong indication for an overall correct operation, since an incorrect map induces an unstable state estimation and vice versa. In total, the system mapped 724 landmarks and ran stable over all 2,026 frames of the sequence.

Recently, Williams et al. [35] presented a system that is also capable of recovering from complete failure. They achieved robustness with a hybrid combination between template matching and a modified version of randomized trees. However, their map typically contains one order of magnitude fewer landmarks and there has been no indication that the modified trees will still be capable of handling a larger number of interest points.

We also validated our system quantitatively. First, we checked the relative accuracy for reconstructed pairs of 3D points and found an error between 3.5 and 8.8 percent on their euclidean distances. Second, the absolute accuracy was assessed by choosing two world planes parallel to the ground plane on top of two boxes in the scene on which some points were reconstructed. Their z-coordinates deviated on average 7-10 mm. Given that the camera is at

roughly 0.6-1.4 m from the points under consideration, this represents a very good accuracy.

6 CONCLUSION

We have presented a powerful method for image patch recognition that performs well even in the presence of severe perspective distortion. The "semi-naive" structure of ferns yields a scalable, simple, and fast implementation to what is one of the most critical step in many computer vision tasks. Furthermore, the ferns naturally allow trade-offs between computational complexity and discriminative power. As computers become more powerful, we can add more ferns to improve the performance. Conversely, we can adapt them to low computational power such those on hand-held systems by reducing the number of ferns. This has actually been done in a recent work [34] to achieve real-time performance on a mobile phone.

A key component of our approach is the naive Bayesian combination of classifiers that clearly outperforms the averaging of probabilities we used in an earlier work [20]. To the best of our knowledge, a clear theoretical argument motivating the superiority of naive Bayesian (NB) techniques does not exist.

There is, however, strong empirical evidence that they are effective in our specific case. This can be attributed to two different causes. First, unlike mixture models, the product models can represent much sharper distributions [17]. Indeed, when averaging is used to combine distributions, the resulting mixture has higher variance than the individual components. In other words, if a single fern strongly rejects a keypoint class, it can counter the combined effect of all the other ferns that gives a weak positive response. This increases the necessity of larger amounts of training data and the help of a prior regularization term, as discussed in Section 3. Second, the classification task, which just picks a single class, will not be adversely affected by the approximation errors in the joint distribution as long as the maximum probability is assigned to the correct class [15], [12]. We have shown that such a naive combination strategy is a worthwhile alternative when the specific problem is not overly sensitive to the implied independence assumptions.

REFERENCES

[1] Y. Amit, *2D Object Detection and Recognition: Models, Algorithms, and Networks.* The MIT Press, 2002.
[2] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation,* vol. 9, no. 7, pp. 1545-1588, 1997.
[3] J. Beis and D.G. Lowe, "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1000-1006, 1997.
[4] C.M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.
[5] A. Böcker, S. Derksen, E. Schmidt, A. Teckentrup, and G. Schneider, "A Hierarchical Clustering Approach for Large Compound Libraries," *J. Chemical Information and Modeling,* vol. 45, pp. 807-815, 2005.

---

2. Other systems commonly use a motion model to predict the feature location in the next frame and accordingly restrict the search area for template matching.

[6] A. Bosch, A. Zisserman, and X. Munoz, "Image Classification Using Random Forests and Ferns," *Proc. Int'l Conf. Computer Vision,* 2007.

[7] M. Calonder, V. Lepetit, and P. Fua, "Keypoint Signatures for Fast Learning and Recognition," *Proc. European Conf. Computer Vision,* Oct. 2008.

[8] C. Chow and C. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Trans. Information Theory,* vol. 14, no. 3, pp. 462-467, May 1968.

[9] O. Chum and J. Matas, "Matching with PROSAC—Progressive Sample Consensus" *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 220-226, June 2005.

[10] A.J. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-Time Single Camera Slam," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 6, pp. 1052-1067, June 2007.

[11] A.J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," *Proc. Int'l Conf. Computer Vision,* vol. 2, pp. 1403-1410, 2003.

[12] P. Domingos and G. Provan, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," *Machine Learning,* vol. 29, pp. 103-130, 1997.

[13] L. Fei-Fei, 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.

[14] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[15] J.H. Friedman and U. Fayyad, "On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality," *Data Mining and Knowledge Discovery,* vol. 1, pp. 55-77, 1997.

[16] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete Solution Classification for the Perspective-Three-Point Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 8, pp. 930-943, Aug. 2003.

[17] G.E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation,* vol. 14, pp. 1771-1800, 2002.

[18] D. Hoiem, R. Sukthankar, H. Schneiderman, and L. Huston, "Object-Based Image Retrieval Using the Statistical Structure of Images," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 490-497, 2004.

[19] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, "On Combining Classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, pp. 226-239, Mar. 1998.

[20] V. Lepetit and P. Fua, "Keypoint Recognition Using Randomized Trees," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1465-1479, Sept. 2006.

[21] D. Lowe, Demo Software: Sift Keypoint Detector, http://www.cs.ubc.ca/~lowe/keypoints/, 2008.

[22] D.G. Lowe, "Distinctive Image Features from Scale Invariant Keypoints," *Int'l J. Computer Vision,* vol. 20, no. 2, pp. 91-110, 2004.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A Comparison of Affine Region Detectors," *Int'l J. Computer Vision,* vol. 65, nos. 1/2, pp. 43-72, 2005.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proc. Am. Assoc. Artificial Intelligence Nat'l Conf. Artificial Intelligence,* 2002.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," *Proc. 16th Int'l Joint Conf. Artificial Intelligence,* 2003.

[26] P. Moreels and P. Perona, "Evaluation of Features Detectors and Descriptors Based on 3D Objects," *Int'l J. Computer Vision,* vol. 73, 2006. no. 3, pp. 263-284, July 2007.

[27] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[28] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 971-987, July 2002.

[29] M. Ozuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 2007.

[30] C. Schmid and R. Mohr, "Local Grayvalue Invariants for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 530-534, May 1997.

[31] J. Sivic and A. Zisserman, "Video Google: Efficient Visual Search of Videos," *Toward Category-Level Object Recognition,* pp. 127-144, Springer, 2006.

[32] C. Strecha, R. Fransens, and L. Van Gool, "Wide Baseline Stereo from Multiple Views: A Probabilistic Account," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 552-559, 2004.

[33] C. Strecha, R. Fransens, and L. Van Gool, "Combined Depth and Outlier Estimation in Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* 2006.

[34] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," *Proc. Int'l Symp. Mixed and Augmented Reality,* Sept. 2008.

[35] B. Williams, G. Klein, and I. Reid, "Real-Time Slam Relocalisation," *Proc. Int'l Conf. Computer Vision,* 2007.

[36] F. Zheng and G.I. Webb, "A Comparative Study of Semi-Naive Bayes Methods in Classification Learning," *Proc. Fourth Australasian Data Mining Conf.,* pp. 141-156, 2005.

Mustafa Özuysal received the BSc degree in electrical and electronics engineering in 2002 and the MSc degree in computer vision from Middle East Technical University (METU), Ankara, Turkey. He is currently working toward the PhD degree in the Computer Vision Laboratory (CVLab) at the Swiss Federal Institute of Technology at Lausanne (EPFL). His research focuses on fast keypoint matching for object tracking and detection.
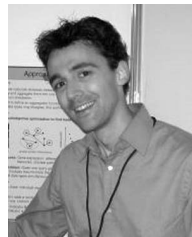
Michael Calonder received the graduate degree in 2006 and the MS degree in mechanical engineering with focus on numerical fluid mechanics and computational science from ETH Zurich, Switzerland, where he is currently working toward the PhD degree in the Computer Vision Group. His thesis focuses on high-speed feature point description and its applications, especially SLAM.

Vincent Lepetit received the engineering and master's degrees in computer science from the ESIAL in 1996 and the PhD degree in computer vision from the University of Nancy, France, in 2001, after working in the ISA INRIA team. He then joined the Virtual Reality Lab at the Swiss Federal Institute of Technology (EPFL) as a postdoctoral fellow and became a founding member of the Computer Vision Laboratory. He has received several awards in computer vision, including the Best Paper Award at CVPR '05. His research interests include vision-based augmented reality, 3D camera tracking, and object recognition.

Pascal Fua received the engineering degree from the Ecole Polytechnique, Paris, in 1984 and the PhD degree in computer science from the University of Orsay in 1989. In 1996, he joined the Swiss Federal Institute of Technology (EPFL), where he is now a professor in the School of Computer and Communication Science. Before that, he was with SRI International and at INRIA Sophia-Antipolis as a computer scientist. His research interests include human body modeling from images, optimization-based techniques for image analysis and synthesis, and the use of information theory in the area of model-based vision. He has coauthored more than 100 publications in refereed journals and conferences. He is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and has been a program committee member of several major vision conferences. He is a senior member of the IEEE.