Keypoint Recognition using Randomized Trees

Vincent Lepetit and Pascal Fua[1]
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Computer Vision Laboratory, I&C Faculty
CH-1015 Lausanne, Switzerland
{Vincent.Lepetit, Pascal.Fua}@epfl.ch
http://cvlab.epfl.ch

To appear in IEEE PAMI

Abstract

In many 3–D object-detection and pose-estimation problems, run-time performance is of critical importance. However, there usually is time to train the system, which we will show to be very useful. Assuming that several registered images of the target object are available, we developed a keypoint-based approach that is effective in this context by formulating wide-baseline matching of keypoints extracted from the input images to those found in the model images as a classification problem. This shifts much of the computational burden to a training phase, without sacrificing recognition performance. As a result, the resulting algorithm is robust, accurate, and fast-enough for frame-rate performance.

This reduction in run-time computational complexity is our first contribution. Our second contribution is to show that, in this context, a simple and fast keypoint detector suffices to support detection and tracking even under large perspective and scale variations. While earlier methods require a detector that can be expected to produce very repeatable results in general, which usually is very time-consuming, we simply find the most repeatable object keypoints for the specific target object during the training phase.

We have incorporated these ideas into a real-time system that detects planar, non-planar, and deformable objects. It then estimates the pose of the rigid ones and the deformations of the others.

Index Terms

Image Processing and Computer Vision, Object recognition, Tracking, Statistical, Classifier design and evaluation, Edge and feature detection.

[1]This work was supported in part by the Swiss National Science Foundation.

I. Introduction

In many 3–D object-detection and pose estimation problems ranging from Augmented Reality to Visual Servoing, run-time performance is of critical importance. However, there usually is time to train the system before actually using it. Furthermore 3–D models, or multiple images from which such models can be built, tend to be available. As shown in Figures 1 and 2, we describe here a technique designed to operate effectively in this context by shifting much of the computational burden to the training phase so that run-time detection becomes both fast and reliable.

Our approach, like many others, relies on matching interest points extracted from training images and those extracted from input images acquired at run-time under potentially large perspective and scale variations. What is new is to formulate this wide-baseline matching problem as a classification problem. More specifically, we consider the set of all possible appearances of each individual object keypoint as a class, which we refer to as the *view-set*. During training, given at least one image of the target object, we extract interest points and generate numerous synthetic views of their possible appearance under perspective distortion, which are then used to train a classifier. It is used at run-time to recognize the keypoints under perspective and scale variations by deciding to which view-set, if any, their appearance belongs. We advocate the use of randomized trees [2] as the classification technique, because they naturally handle multi-class problems and are robust and fast, while remaining reasonable easy to train.

Not only is this approach more principled than many earlier ones, but it yields a reduction in run-time computational complexity, which is our first contribution. Our method does away with having to compute *ad hoc* descriptors, rectify patches, and search for nearest-neighbors, which are time-consuming steps common in earlier methods [28], [29], [3], [19], [17], [27], [15], [21].

Another contribution is to show that, in this context, a simple and fast keypoint detector suffices for operation even under large perspective and scale variations, which results in a further increase in performance. While earlier methods require detectors that can be depended upon to produce very repeatable results, which can be very time-consuming, we simply find the most repeatable object keypoints for the specific target object during the training phase,

In the remainder of the paper, we first discuss related work and formulate wide-baseline matching as a classification problem. We then discuss the building of our view-sets and our use

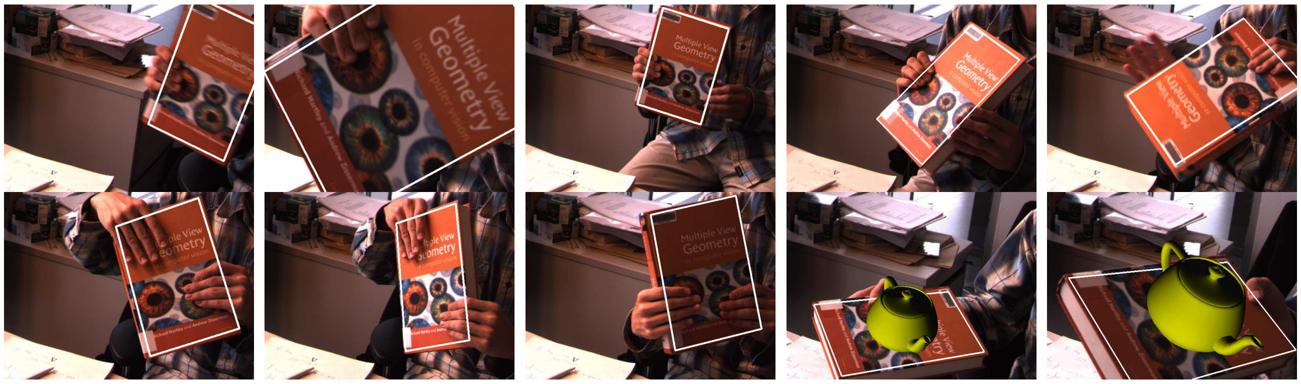


Fig. 1. Detection of a book in a video sequence: As shown by the white outline, the book is detected independently and successfully in all frames at 25Hz in 640×480 images on a standard PC, in spite of partial occlusion, cluttered background, motion blur, large illumination and pose changes. In the last two frames, we add the inevitable virtual teapot to show we also recover 3–D pose. The corresponding video sequence is submitted as supplementary material.

of randomized-trees to perform the classification. Finally, we present our results.

II. Related Work

In the area of automated 3D object detection, we can place existing approaches on a continuum ranging from purely "Global" ones to purely "Local" one. The former typically use machine learning techniques while the latter usually rely on less principled methods. A number of practical techniques, such as the coarse to fine method proposed in [9], combine the strengths of both kinds of methods. We will argue that our approach also does this by applying machine learning techniques for the recognition of local parts, while providing an accurate 3D pose. For clarity's sake, in this section, we nevertheless classify approaches as either global or local according to which aspect dominates.

Global approaches use statistical classification techniques to compare an input image to several training images of an object of interest and decide whether or not it appears in this input image. The methods used range from relatively simple methods such as Principal Component Analysis and Nearest Neighbor search [23] to more sophisticated ones such as AdaBoost and classifiers cascade to achieve real-time detection of human faces at varying scales [9], [30]. These approaches focus on recognizing instances of generic classes without providing an accurate 3D pose estimation. Some of them can handle tens of instances of the object class in one image but are not designed for robustness to occlusions, cluttered backgrounds, or poses different from those in the training set.

By contrast, local approaches use simple 2D features such as corners or edges, which makes them resistant to partial occlusions and cluttered backgrounds: Even if some features are missing, the object can still be detected as long as enough are found and matched. Spurious matches can be removed by enforcing geometric constraints, such as epipolar constraints between different views or full 3D constraints if an object model is available. The use of weaker geometric constraints is discussed in [1], [8] and is shown to achieve robustness to occlusion and clutter. However, to be effective in our context, the feature extraction and characterization should also be insensitive to viewpoint and illumination changes.

As proposed in [13], scale-invariant extraction can be achieved by taking the local extrema of a Laplacian-of-Gaussian pyramid in scale-space as feature points. To increase computational efficiency, the Laplacian can be approximated by a Difference-of-Gaussian [14]. More recent work has focused on achieving affine invariant region detection to handle large perspective distortions as well. [3], [27], [19] used an affine invariant point detector based on the Harris detector, where the affine transformation that equalizes the two eigenvalues of the auto-correlation matrix is evaluated to rectify the patch appearance. [29] achieves invariance by fitting an ellipse to the local texture. [17] proposes a fast algorithm to extract Maximally Stable Extremal Regions. [21] gives an extensive study on these affine invariant regions detectors.

Given the extracted feature points, various local descriptors have been proposed [28], [22]. Among these, the SIFT descriptor [15] has been shown to be one of the most effective [20]. It relies on local orientation histograms and tolerates significant local deformations. In [21], it is applied to rectified affine invariant regions to achieve perspective invariance. In [25], a similar result is obtained by training the system using multiple views of a target object, storing all the SIFT features from these views, and matching against all of them. [11] performs Principal Component Analysis on such local orientation histograms, which appears to improve the reliability of this representation. However, such descriptors can be costly, and whatever the chosen descriptor is, matching is performed by nearest-neighbor search, which tends to be computationally expensive, even when using an efficient data structure [4].

By contrast, in earlier work [12], we argued that formulating wide-baseline matching as a classification problem let us shift much of the computational burden to a training phase. This

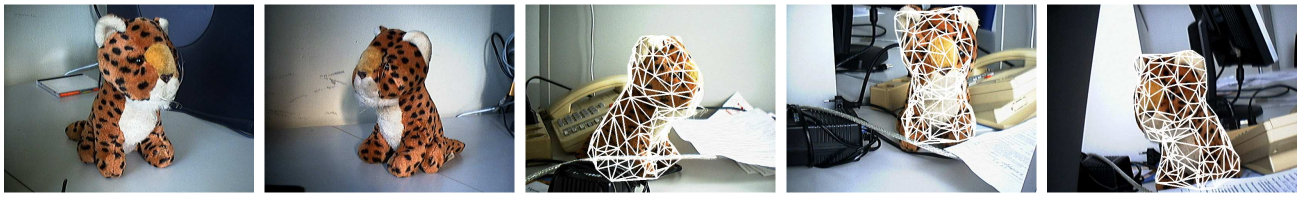


Fig. 2. The method is just as effective for 3–D objects. In this experiment, we detected the stuffed tiger in the three images to the right using a 3–D model reconstructed from several views such as the two first images on the left. Note that it reprojects at the right place in all three images.

reduces the cost of online matching while increasing its robustness. In this early work, we used PCA as the classification technique. Here, we advocate further reducing the run-time computational complexity by using Randomized Trees instead. They have been extensively used for character recognition [2] and more recently for image classification [16]. They require additional training in exchange for increased run-time efficiency because they let us replace projection in the eigenspace and nearest-neighbor search by simple binary tests on image gray-levels.

Classification as a technique for wide baseline matching has also been explored in parallel to our own work [18]. In this approach, the training set is iteratively built from incoming frames. While this is well-adapted for applications that do not allow for a training stage, this approach was not designed to allow recognition from views that are very different from those already seen. Furthermore, it uses kernel PCA, which is even more computationally expensive than PCA-based classification.

III. Wide Baseline Point Matching as a Classification Problem

Our approach relies on matching keypoints found in an input image against those on a target object $\mathbf{O}$. Once potential correspondences have been established, we apply standard techniques to estimate 3–D pose. Therefore, the critical step in achieving results such as those depicted by Figures 1 and 2 is the fast and robust wide-baseline matching that handling large perspective and scale changes implies, which we formulate below in terms of a classification problem.

During training, we construct a set $\mathbf{K} = \{\mathbf{k}_1 \ldots \mathbf{k}_N\}$ of N prominent keypoints lying on the object model. At runtime, given an input patch $\mathbf{p}(\mathbf{k}^{\text{input}})$ centered at a keypoint $\mathbf{k}^{\text{input}}$ extracted from the input image, we want to decide whether or not its appearance matches that one of

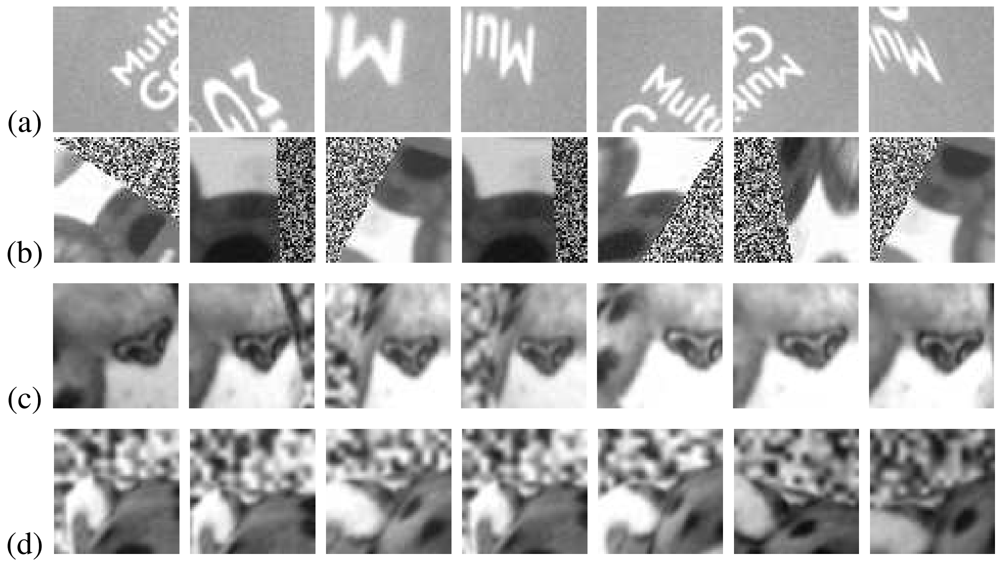


Fig. 3. Sampling the view-set. (a) For a keypoint on the cover of the book of Figure 1, we synthesize new views using random affine transformations and adding white noise. (b) Same thing for another keypoint located on the border of the book. (c, d): In the case of the stuffed tiger, the sampling is obtained using a textured 3–D model seen from different viewpoints.

the N keypoints $\mathbf{k}_i$. In other words, we want to find for $\mathbf{p}(\mathbf{k}^{\text{input}})$ its class label $Y(\mathbf{p}) \in \mathbf{C} = \{-1, 1, 2, \ldots, N\}$, where the -1 label denotes all the points that do not belong to $\mathbf{K}$. Since Y cannot be directly observed, we build a classifier $\hat{Y}$ such as $P(Y(\mathbf{p}) \neq \hat{Y}(\mathbf{p}))$ is small.

In other tasks such as face detection or character recognition, large training sets of labeled data are usually available. However, for automated pose estimation, it would be impractical to require a very large number of sample images. Instead, to achieve robustness with respect to pose and complex illumination changes, we use a small number of images and synthesize many new views of the object using simple rendering techniques. For each keypoint, this gives us a sampling of its *view set*, that is the set of all its possible appearances under different viewing conditions. The first two rows of Figure 3 depict such a sampling for two keypoints detected on the book of Figure 1, the two last rows correspond to two keypoints detected on the stuffed tiger of Figure 2.

These samplings are virtually infinite training sets that serve as input to the tree-building algorithms. The resulting trees form a very fast $\hat{Y}$ run-time classifier as introduced above. Its output is a set of matches that we use to estimate the pose using a standard robust estimation technique.

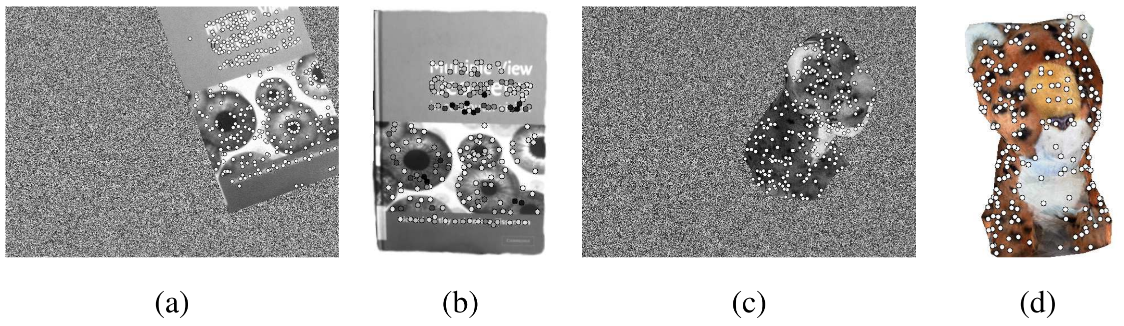


Fig. 4. Building the view-sets. (a) A synthetic view for the book cover, with extracted keypoints, (b) The most stable keypoints selected by our method. (c, d) Same thing for the stuffed tiger.

IV. Building the View Sets

A simple approach to building the view sets would be to extract keypoints from the original images and then process them independently as we did in [12]. However, a more effective approach is to generate new views of the whole object such as those of Figure 4, and extract keypoints from these. In this way, we can easily select keypoints that are stable under noise and perspective distortion, which helps making the matching robust to noise and cluttered background at no additional run-time computational cost.

Recall from Section III, that, for detection purposes, we describe the target object as a set of keypoints $\mathbf{K} = \{\mathbf{k}_i\}$ lying on the target object, and expressed in a reference system related to it. We discuss below the construction of the view-sets given the keypoints and then the process of extracting and selecting them.

A. Sampling the View-Sets

If the object is of uniformly low curvature, patches surrounding the $\mathbf{k}_i$ keypoints can be treated as locally planar and their distortions under perspective projection as homographies. In this case, only one frontal view of the target object is enough to generate new views. We approximate them by affine transformations that we draw randomly to sample the view-set. An affine transformation can be decomposed as $\mathbf{A} = \mathbf{R}_\theta \mathbf{R}_\phi^{-1} \mathbf{S} \mathbf{R}_\phi$, where $\mathbf{R}_\theta$ and $\mathbf{R}_\phi$ are two rotation matrices respectively parameterized by the angles θ and ϕ, and $\mathbf{S} = \text{diag}\,[\lambda_1, \lambda_2]$ is a scaling matrix. For all the planar object examples shown in this paper, θ and ϕ are drawn uniformly from the interval $[-\pi; +\pi]$, and λ_1 and λ_2 from the interval $[0.6; 1.5]$. This range of scales is sufficient to handle variations within a single octave, while larger scale changes are handled by interest point detection at several scales, as discussed below.

If the object is more complex, we can take advantage of a 3–D model that has been built either automatically or by hand using several images of the object. We use it in conjunction with standard texture-mapping techniques to generate new views under perspective transformations. In the case of the stuffed tiger of Figure 2, we used the ImageModeler software package[2] to quickly build a very coarse 3–D model, which has proved to be sufficient. Random profile and frontal views of the tiger were generated by moving the camera around it while maintaining the distance between 30 to 50 cm and the camera tilt angle between -30 and 30 degrees. As a result, we can reliably detect the object from images acquired with cameras moving within those bounds. Furthermore, thanks to the multi-scale matching scheme, we can actually place the camera either further or closer.

We experimented with objects that exhibit relatively few specularities and therefore synthesized our training images without accounting for lighting effects. We can nevertheless handle changes in lighting because the tests we perform to classify patches, which we discuss in more detail in Section V, rely on binary comparisons between image intensities. As a result, they are invariant to the often monotonous intensity changes that lighting differences tend to produce in any given neighborhood and lighting effects only perturb a very small fraction of the matches. Those erroneous matches, being few in number, can easily be discarded by the robust estimator we use for pose estimation. This is why the specular reflections and image saturation effects on the book cover and the plastic part of the stuffed tiger do not degrade the performance of our system, even though it has not been trained using images that exhibit such lighting effects. For more complex cases, it should be possible to capture the material properties and generate very realistic images [7]. However, it would be cumbersome and we have not found it to be necessary.

B. Fast Multiscale Keypoint Extraction

Our approach does not rely on a specific method for extracting keypoints, and we could have used any of the many methods for multiscale detection that have recently been proposed. However since we focus on real-time applications we developed our own method, which is

[2]ImageModeler is a commercial product from Realviz$^{(tm)}$ that allows semi-automated 3–D reconstruction from several views.

sufficiently fast and yet stable for our purposes. As was done in [20], [15], we look for extrema of the Laplacian of Gaussian that are sufficiently "cornerlike." However, instead of computing some score map such as the Laplacian or the Harris matrix at each location and then searching for the keypoints in this map, we proceed as follows.

Following a philosophy similar to the one proposed in [26], for each pixel $\mathbf{m}$, we consider the intensities along a discretized circle centered at $\mathbf{m}$. We then eliminate all pixels whose gray level is close to those of any two diametrically opposed pixels on this circle and that are therefore unlikely to be stable keypoints. Among the remaining ones, we estimate the Laplacian using gray level differences between pixels on the circle and the central one and retain only the locations where this estimate is largest. This very simple approach often requires only a few intensity comparisons to eliminate points in uniform areas and along edges and a few more to select the maxima of the Laplacian. As a result, it is very fast and has proved sufficiently robust for our purposes.

We run this algorithm on the first few octaves of the image and simultaneously use the interest points detected at each octave to train the classifier, as described below. Figure 5 illustrates how this simple procedure and the keypoint classifier work in tandem to recognize the keypoints under large variation of both scale and appearance. This contrasts with earlier techniques [21] that depend on much more sophisticated and expensive methods to accurately extract the affine deformation of a feature.

C. Keypoint Selection

Ideally, all the $\mathbf{k}_i$ keypoints in $\mathbf{K}$ should have a high probability $P(\mathbf{k}_i)$ to be found if they are visible, perspective distortion and image noise notwithstanding.

Let $\mathcal{T}$ be the geometric transformation used to synthesize a new view as described in Section IV-A, and $\widetilde{\mathbf{k}}$ an interest point extracted from this view using the procedure of Section IV-B. $\mathcal{T}$ is either an affine transformation or a projection and by applying $\mathcal{T}^{-1}$ to $\widetilde{\mathbf{k}}$, we can recover a corresponding keypoint $\mathbf{k}$ in the reference system. Thus, given several synthetic views, $P(\mathbf{k})$ can be estimated by simply counting how often it is found. The set $\mathbf{K}$ is then constructed by only retaining keypoints with a high $P(\mathbf{k})$. In our experiments, we retain the 200 keypoints with highest $P(\mathbf{k})$. Figure 4 shows the keypoints selected on the book cover and the stuffed tiger. For each keypoint $\mathbf{k}_i \in \mathbf{K}$, we build the corresponding view set by collecting the $\mathbf{p}$ neighborhood

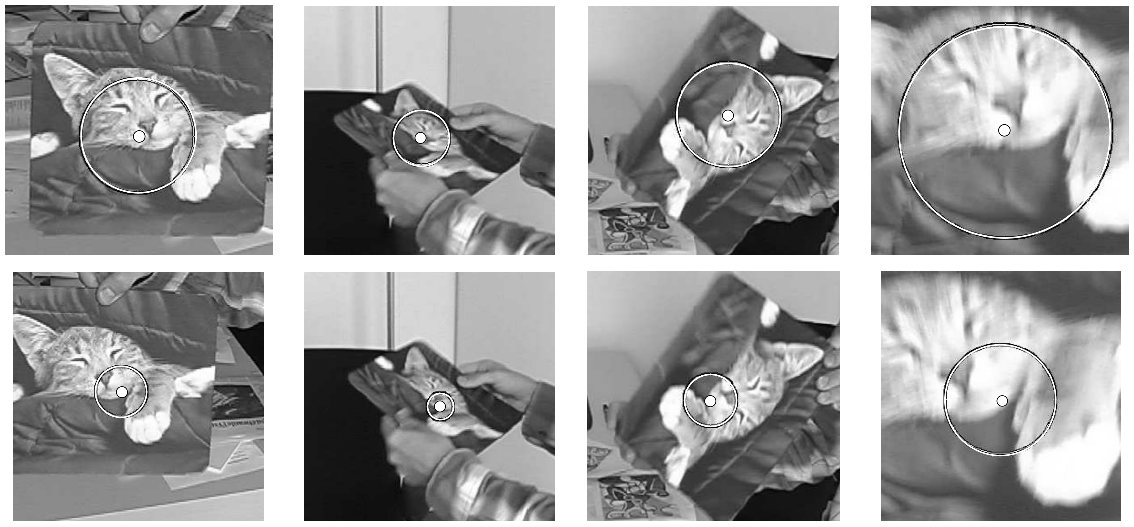


Fig. 5. Robustness to scale and perspective changes. First row: The image on the left shows a keypoint selected on the original image, the three images on the right show the same keypoint retrieved at different scales under perspective distortion, and some saturation and blur effects. Second row: Same for another keypoint.

of the corresponding $\widetilde{\mathbf{k}}$ in the generated images, as shown in Figure 4.

The above procedure accounts for sensitivity of the detector to perspective distortions. However, this may not be sufficient because, when a keypoint is detected in two different images, its precise location may shift a bit due to image noise and viewpoint change. In practice, such a positional shift results in large errors of direct cross-correlation measures. One solution would be to iteratively refine the keypoint localization [19], which could be costly. Instead, we solve this problem by adding white noise to the synthetic views before keypoint extraction. The resulting view sets thus capture the positional shift, and force the classifier to learn invariance to this shift.

Each view is rendered against a different, complex random background. The classifier is therefore trained to recognize the keypoints on cluttered background, including points that are close to the object visual boundary. This contrasts with other patch-based methods that fail to match a keypoint when the corresponding patch overlaps the background.

V. Keypoint Classification and Recognition

Several classification algorithms, such as K-Nearest Neighbor, Support Vector Machines or neural networks could have been chosen to implement the classifier Y introduced in Section III. Among those, we have found randomized trees [2] to be eminently suitable because they naturally

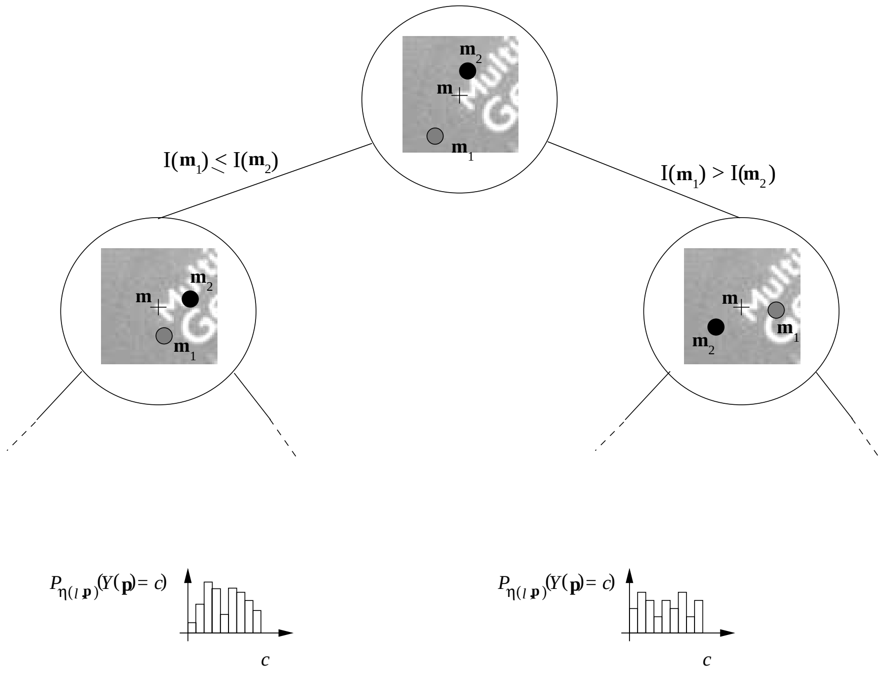


Fig. 6. Generic tree used for keypoint recognition. When using C_2 tests, the nodes contain tests comparing two pixels in the keypoint neighborhood; the leaves contain the $P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$ posterior probabilities.

handle multi-class problems and are robust and fast, while remaining reasonably easy to train. They are simple but powerful tools for classification, introduced and applied to recognition of handwritten digits in [2]. They are closely related to the regression trees in the CART method [5]. Several trees are grown with some form of randomization as in [6] for example, but the queries can be more complex than those of regression trees. In this section we first describe them briefly in the context of our problem for the benefit of the unfamiliar reader. We then study their properties and justify our implementation choices.

A. Randomized Trees

Figure 6 depicts a generic tree. Each internal node contains a simple test that splits the space of data to be classified, in our case the space of image patches. Each leaf contains an estimate based on training data of the posterior distribution over the classes. A new patch is classified by dropping it down the tree and performing an elementary test at each node that sends it to one side or the other. When it reaches a leaf, it is assigned probabilities of belonging to a class depending on the distribution stored in the leaf. Since the numbers of classes, training

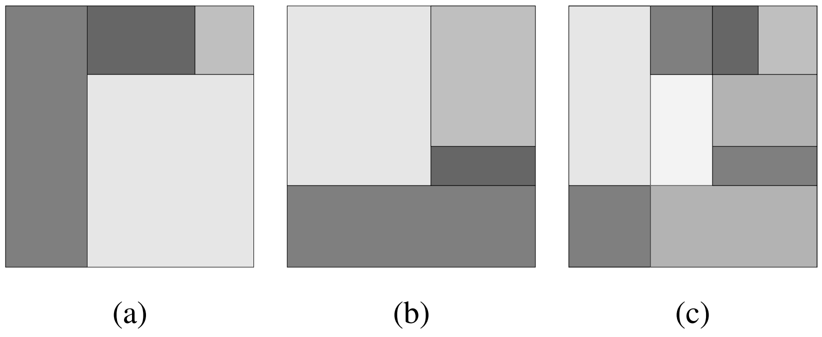


Fig. 7. What makes the Randomized Trees tick. (a) and (b) are two partitions, which result in a finer one (c) when they are superimposed. In a similar manner, each tree performs a different partition of the patch space, and combining their responses results in a finer partition.

examples and possible tests are large in our case, building the optimal tree quickly becomes intractable. Instead, multiple trees are grown so that each tree yields a different partition of the space of image patches. We discuss two different ways to build the trees in Section V-C. Once the trees $T_1, \ldots, T_L$ are built, their responses are combined during classification to achieve a better recognition rate that a single tree could. More formally, the tree leaves store posterior probabilities $P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$, where c is a label in $\mathbf{C}$ and $\eta(l, \mathbf{p})$ is the leaf of tree T_l reached by the patch $\mathbf{p}$. Such probabilities are evaluted during training as the ratio of the number of patches of class c in the training set that reach η and the total number of patches that reach η. Following [2], $\mathbf{p}$ is classified by considering the average of the probabilities $P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c)$:

$$\hat{Y}(\mathbf{p}) = \underset{c}{\text{argmax}}\ p_c(\mathbf{p}) = \underset{c}{\text{argmax}}\ \frac{1}{L} \sum_{l=1\ldots L} P_{\eta(l,\mathbf{p})}(Y(\mathbf{p}) = c) \tag{1}$$

Figure 7 gives an intuitive idea of what makes this approach a good one. Each tree performs a different partition depending on the specific tests it contains. Combining the output of all the trees results in a finer partition that yields a better classification. As the depth and number of trees are increased, this partition becomes finer and finer, and the estimated distribution becomes closer and closer to the actual one, at the cost of increasing both computational expense and memory requirements. This trade-off will be studied in Section V-C.

$p_c(\mathbf{p})$ is the average of the posterior probabilities of class c and constitutes a good measure of the match confidence. We estimate during training a threshold T_c to decide if the match is correct or not with a given confidence s that is

$$P(Y(\mathbf{p}) = c | \hat{Y}(\mathbf{p}) = c, p_c(\mathbf{p}) > T_c) > s\,.$$

In practice, we take s between 60% and 90%. Keypoints for which $p_c(\mathbf{p})$ is lower than T_c are considered as keypoints detected on the background, or misclassified keypoints, and are therefore rejected. This leaves only a small number of erroneous matches that RANSAC can easily handle.

The drawback of Randomized Trees is their greedy use of memory. Their size in memory increases exponentially with the depth, and linearly with the number of trees. For example, a single tree of depth 15 uses about 32 Mo for a 200 classes problem. Therefore, the chosen number of trees and their depth are a trade-off between the computer memory dedicated to store them and the recognition rate. In the following, we study the influence of these parameters on the recognition rate.

B. Node Tests

In our implementation, the tests performed at the nodes are simple binary tests based on the difference of intensities of two pixels $\mathbf{m_1}$ and $\mathbf{m_2}$ taken in the neighborhood of the keypoint. We write these tests as

$$C_2(\mathbf{m_1}, \mathbf{m_2}) = \begin{cases} \text{If } \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_1}) \leq \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_2}) & \text{go to left child} \\ \text{otherwise} & \text{go to right child} \end{cases},$$

where $\mathrm{I}_\sigma(\mathbf{p}, \mathbf{m})$ is the intensity of patch $\mathbf{p}$ at pixel location $\mathbf{m}$, after Gaussian smoothing to reduce influence of noise. Such a test can be seen as a test on the polarity between the two locations $\mathbf{m_1}$ and $\mathbf{m_2}$. In all our experiments, the patches are of size 32×32, so that the total number of possible C_2 tests is 2^{19}. Fortunately, since real-world images exhibit spatial coherence, only a very small subset is required to yield good recognition rates.

As shown below, a few hundreds of these simple tests are usually enough to classify a patch. This involves only a few hundreds intensity comparisons and additions per patch, and is therefore very fast. Furthermore, because they only depend on the order of the pixel intensities between neighbors, they tend to be fairly insensitive to illumination changes other than those caused by a moving shadow. In other words, to achieve the robustness to illumination effects demonstrated in Figure 1, our technique, unlike many others, does *not* require us to normalize the pixel intensities, for example by setting the L_2 norm of the intensities to one.

Note, however, that one advantage of randomized trees is that they impose no restriction on the kind of tests performed at the nodes. A tree can even contain different kinds of tests. An infinite number of tests other than introduced above could have been designed, most of which would be computationally more demanding. To show that the ones we chose represent a good compromise between recognition ability and run-time speed, we will compare in Section V-D their performance with those of more complex tests, including tests based on histograms of local orientations and inspired by the SIFT detector [15].

C. Growing the Trees

To improve the recognition rate, we use multiple trees that should partition the patches space in different manners, as depicted by Figure 7. We experimented with two different methods for building such trees.

The first method is the one used in [2]: The trees are constructed in the classical, top-down manner, where the tests are chosen by a greedy algorithm to best separate the given examples. The expected gain in information is used to evaluate the separation efficiency. The gain caused by partitioning a set S of examples in several subsets S_i according to a given test is measured as

$$\Delta E = -\sum_i \frac{|S_i|}{|S|} E(S_i) \, , \tag{2}$$

where $E(s)$ is the Shannon's entropy $-\sum_{j=1}^{N} p_j \log_2(p_j)$ with p_j the proportion examples in s belonging to class j, and $|.|$ denotes the size of the set. The process of selecting a test is repeated for each non-terminal node, using only the training examples falling in that node. The recursion is stopped when the node receives too few examples, or when it reaches a given depth. In our implementation, the depth varies between 10 and 15, and the minimum of examples is fixed to 10.

The second method is much faster and simpler: Instead of picking questions according to a criterion, we simply pick a random set. This can be seen as an extreme simplification of the first method. For example, in the case of the C_2 test, the two locations $\mathbf{m}_1$ and $\mathbf{m}_2$ for each node are picked at random within the patch, independently of the training samples that fall into the node and of the tests performed further up in the tree.

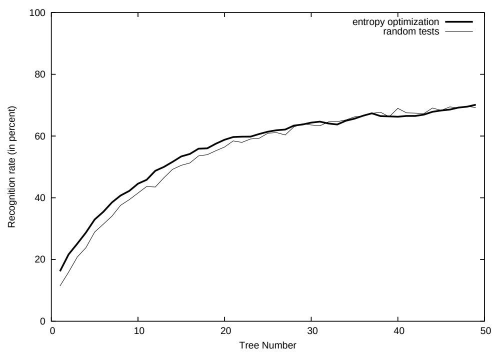
To compare the two tree-building methods we have introduced, we used them both on the set of 200 keypoints depicted by Figure 4. This resulted in two sets of trees whose nodes contained C_2 binary tests and whose depth was limited to the same value.

When using the simpler approach, we grew the trees by randomly picking locations for each node, as discussed above. When using the entropy minimizing approach, we first synthesized 100 new views. We then recursively built the trees by trying n different tests at each node and keeping the best one according to the criterion of Equation (2). For the root node, we chose $n = 10$, a very small number, to reduce the correlation between the resulting trees. For all other nodes, we used $n = 100d$, where d is the depth of the node. Note that this heuristic involves randomizing on both tests and training data. As noted in [2], the randomizing on tests is far more powerful than randomizing on data. We do the latter, not to improve classification performance, but to make our greedy algorithm tractable. Since intra-class variations are large, without it, too many examples would be needed to sample a class in a representative way.

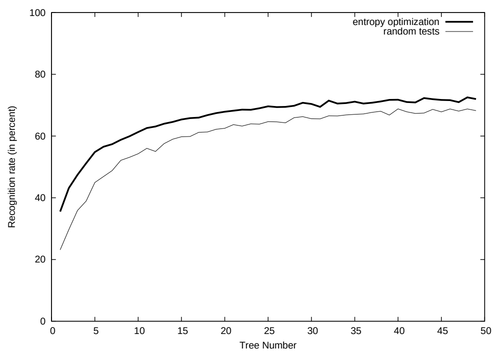
In the case of the completely random approach to building trees, $\mathbf{m}_1$ and $\mathbf{m}_2$ were simply chosen at random. For the two sets, the tree depth is limited to a given maximal depth, and the posterior probabilities are estimated from 1000 new random views per keypoint.

We present here results for trees with a depth limited to 12, which was found to be a good trade-off between the memory requirements and recognition rate. After having grown the trees, the posterior probabilities in the terminal nodes were estimated using 5000 new training images. We then measured the recognition rate R of the two sets of trees by generating new images under random poses, as the ratio of the number of correctly recognized patches and the total number of generated patches. The evolution of R for the two sets of trees with respect to the number of trees is depicted Figure 8(a). Taking the tests at random usually results in a small loss of reliability at least when the number of trees is not large enough but considerably reduces the learning time. The time dedicated to growing the trees drops from tens of minutes to a few seconds on a 2.8 GHz machine.

We also experimented with normalizing the $\mathbf{p}$ patches of Section III both during training and at run-time to achieve higher recognition rates for a given number of trees. As in [15] we attribute a 2–D orientation to the keypoints that is estimated from the histogram of gradient directions in a patch centered at the keypoint. Note that by contrast with [15], we do not require a particularly repeatable method. We just want it to be reliable enough to reduce variation within classes. Once



(a)



(b)

Fig. 8. Comparing the classification rates obtained using trees grown in two different manners, as a function of the number of trees. (a) Without and (b) with patch orientation normalization. The thick lines depict results obtained by selecting tests that maximize the information gain. The thin lines depict results obtained by randomly chosen tests, which result in a small loss of reliability but considerably reduces the training time. Note that in all cases the normalization lets us achieve better results with fewer trees. However when enough trees are used, it does not improve the rates anymore.

the orientation of an extracted keypoint is estimated, its neighborhood is rectified. Figure 8(b) compares the recognition rates with this normalization step for the two different methods of selecting the tests. Taking the tests at random results in a slightly larger but still small loss of reliability. More importantly, the normalization gives us significantly improved rates when using only a small number of trees. However, when using a large number of trees, the recognition rates are similar with and without the normalization.

We draw two practical conclusions from these experiments. First, using random tests is sufficient and keeps the learning time reasonable for practical applications. Second, the orientation normalization step is not required, but lets us reduce the number of trees. Therefore the choice of using such a normalization becomes a trade-off between the amount of time required to normalize and to classify, which is proportional to the number of trees. The amount of memory used by the trees should also be taken into account. In practice, we have found it more effective to normalize and use fewer trees, in part because the normalization we use is much simpler than a full affine rectification. Therefore, in the remainder of the paper, we normalize the patches and use randomly chosen tests.

D. Comparing Different Node Tests on Different Image Types

The above results were obtained using binary tests involving the comparison of gray levels of two neighboring pixels. Here we discuss the value of replacing these tests by more sophisticated ones involving additional pixels. In addition to the family of C_2 binary tests, we also consider tests involving more pixels locations. We define the C_4 family made of tests that compare the differences of intensities of two pairs of pixel locations:

$$C_4(\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \mathbf{m_4}) = \begin{cases} \text{If } \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_1}) - \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_2}) \leq \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_3}) - \mathrm{I}_\sigma(\mathbf{p}, \mathbf{m_4}) & \text{go to left child;} \\ \text{otherwise} & \text{go to right child.} \end{cases}$$

These tests compare both the strength and polarity of the edge between $\mathbf{m_1}$ and $\mathbf{m_2}$, and those of the edge between $\mathbf{m_3}$ and $\mathbf{m_4}$.

We also define the C_h family made of more sophisticated tests based on local orientation histograms computed as follows. Like for the SIFT characterization, each patch is divided into 4×4 subregions, and an array of histograms with 8 orientation bins is computed for each region. The tests in the C_h family compare the values of two bins:

$$C_h(u_1, v_1, o_1, u_2, v_2, o_2) = \begin{cases} \text{If Bin}(u_1, v_1, o_1) \leq \text{Bin}(u_2, v_2, o_2) & \text{go to left child;} \\ \text{otherwise} & \text{go to right child.} \end{cases}$$

The responses of the C_2, C_4 and C_h families of tests were compared using two kinds of feature points. In one case, we used a set we denote as “Title”. It includes 100 keypoints detected on the title of the book cover of Figure 1, which represents strong and structured edges. The second set of keypoints denoted “Eyes” is made of 100 keypoints detected on the picture of the same book cover, which presents mostly textured areas. For both sets, we grew several sets of trees by varying the tests and tree depths. We tested them for depths ranging from 10 to 15, as larger values quickly become intractable due to the increasing memory requirements. Figures 9 and 10 summarize the results as a function of the number of trees and of their depth.

Somewhat surprisingly, the C_4 tests results are slightly worse than the C_2 ones. The C_h tests slightly outperform the C_2 tests on the Title set, which is not very textured, and yield approximately the same recognition rate on the more textured Eyes set.

However, and this is remarkable, the differences are not really significant. To illustrate that, in Figure 11, we superpose the results using the three kinds of tests and the same tree depth equal to 15. This is a very interesting property of the classification approach: Complex tests are

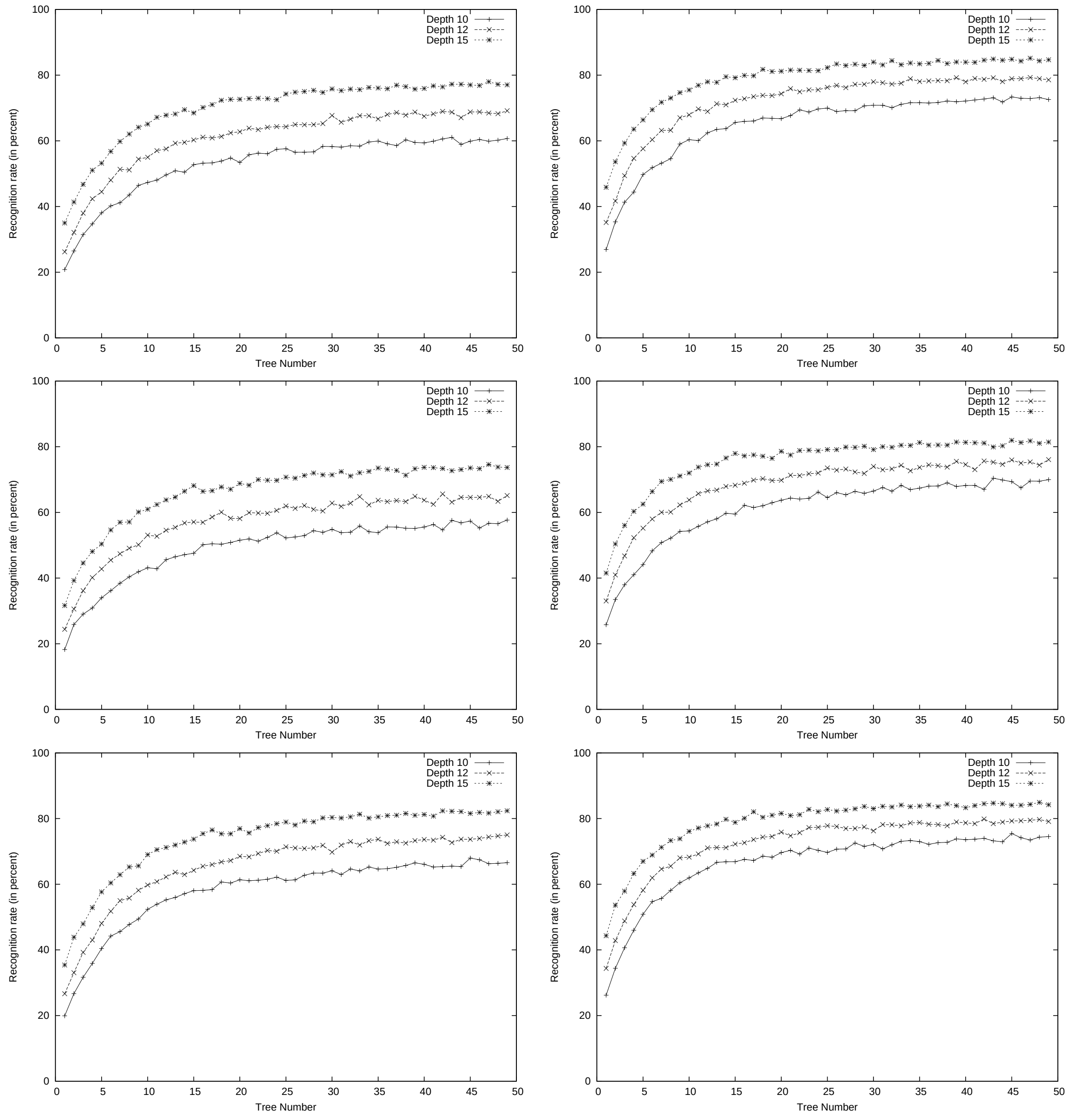
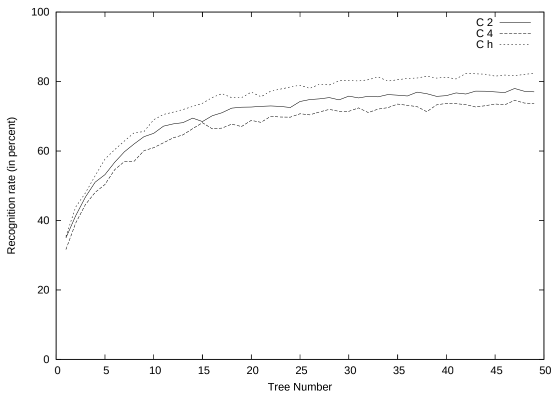


Fig. 9. Comparison of recognition rates as a function of the number of trees, for different tree depths and different tests, applied to the “Title” (left) and the “Eyes” (right) keypoints sets. First row: using the C_2 family tests; Second row: using the C_4 family tests; Third row: using the C_h family tests.

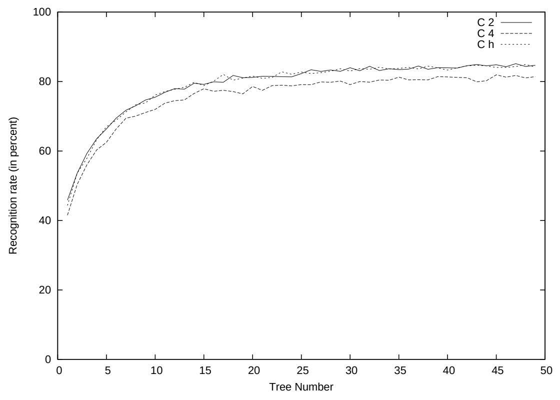
not necessary, simple ones suffice to reach similar performances. This can be explained by the fact that the partition obtained when combining the trees becomes sufficiently fine when enough trees are used, provided that each individual tree partitions the patch space in a different way. Since the C_h tests are computationally much more costly, in practice we use the C_2 tests as the slight loss in performance does not appear to have any ill-effect on our RANSAC-based

		C_2	C_4	C_h
Title set	depth 10	60.7%	57.7%	66.6%
	depth 12	69.2%	65.1%	75.0%
	depth 15	77.0%	73.7%	82.4%
Eyes set	depth 10	72.7%	70.0%	74.5%
	depth 12	78.6%	76.1%	84.2%
	depth 15	84.7%	81.4%	84.2%

Fig. 10. Classification rates using either the C_2, C_4 and C_h test families using 50 trees. Tests from the C_h family slightly outperform the ones from C_2, at an additional computation cost.



(a)



(b)

Fig. 11. Classification rates for C_2, C_4 and C_h tests families as a function of the number of trees. (a) “Title” set, (b) “Eyes” set. The depth of the trees is 15.

approach to pose estimation.

VI. Results

For planar objects such as the books of Figures 1 and 13 and the mouse pad of Figure 12, we first use RANSAC to estimate an affine transformation between the reference image and the input image. It is then refined by robust least-squares estimation. As can be seen in the figures and the video sequences submitted as supplementary material, the pose estimation is robust to illuminations changes, scale changes, occlusions, and some image blur. As shown in Figure 14, we also applied our method for the detection of a sail over a 4000 thousand frame video taken with a home camcorder during a regatta. Despite the bad condition—the sail is not very textured,

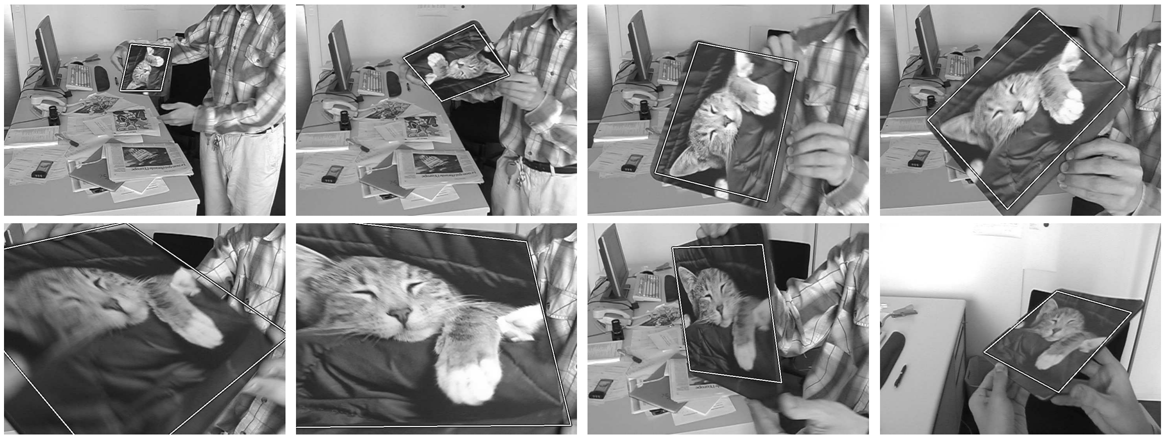


Fig. 12. Detection of a mouse pad at different scales on a cluttered background.

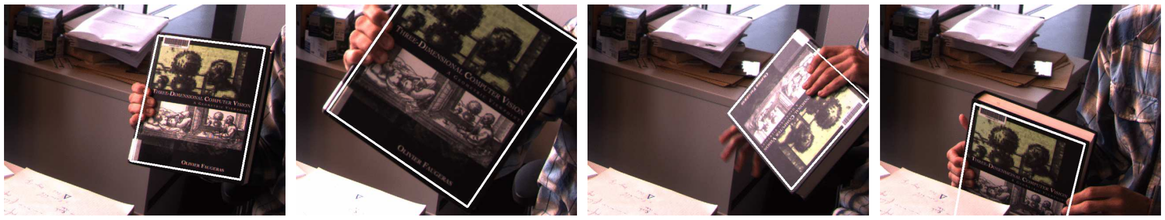


Fig. 13. Another example on which our system has been tested.

it moves in and out of the field of view, the camera motion is very jerky and the illumination changes all the time—the sail is detected in all frames where a sufficient portion of the sail is seen.

Our implementation has been tested on numerous objects under different lighting conditions using a simple webcam, and an executable can be downloaded from our website. It runs at 25 frames per second on a 2.8 GHz PC.

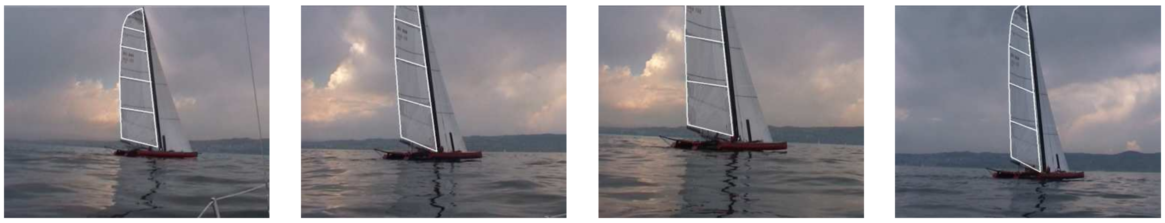


Fig. 14. Detecting a sail. Four frames from a 4000 frame video acquired using a home camcorder during a regatta. Even though the camera jerks, the zoom changes and the lighting conditions are poor, the sail is detected in all frames, such as those shown above, where a sufficient portion of the sail is seen.

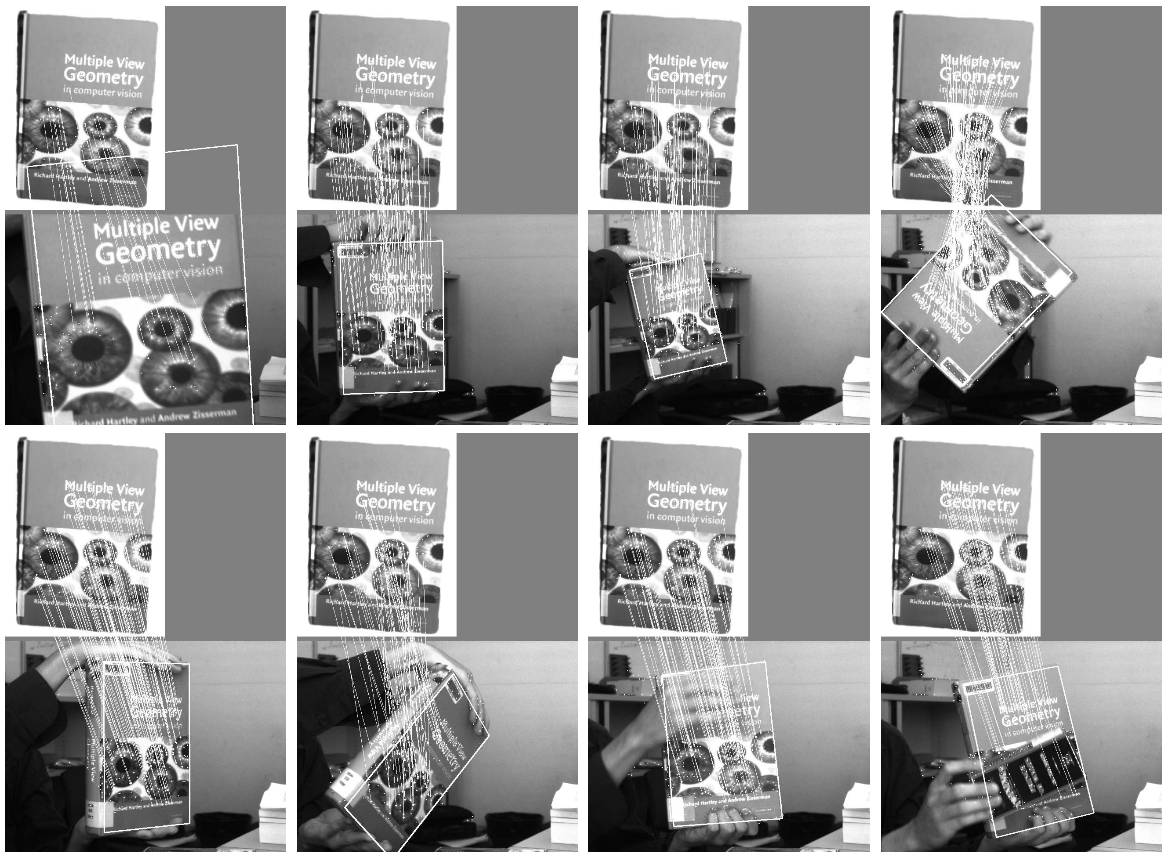


Fig. 15. Detection of the book: The white lines represent the inlier matches established in real-time under several poses. The corresponding video sequence is submitted as supplementary material.

Figures 2 and 18 show detection results for the stuffed tiger for which a quickly built 3–D model. We use a P3P algorithm and RANSAC to robustly estimate its pose. The object object is successfully detected from different sides, different distances and both from below and above.

We compared our results with those obtained using the executable that implements the SIFT method [15] kindly provided by David Lowe. As shown in Figures 16 and 17, when there is little distortion, the SIFT approach yields a few more matches. However, when too much perspective distorts the object image, it produces far fewer, while our approach is not perturbed.

To be fair, we note again that this robustness comes at the cost of a training stage that SIFT does not require. If training is possible, as suggested in [15] and demonstrated in [25], SIFT can be made more robust to distortions by using multiple views of the target object, storing all the SIFT features from these views, and matching against all of them. This does not

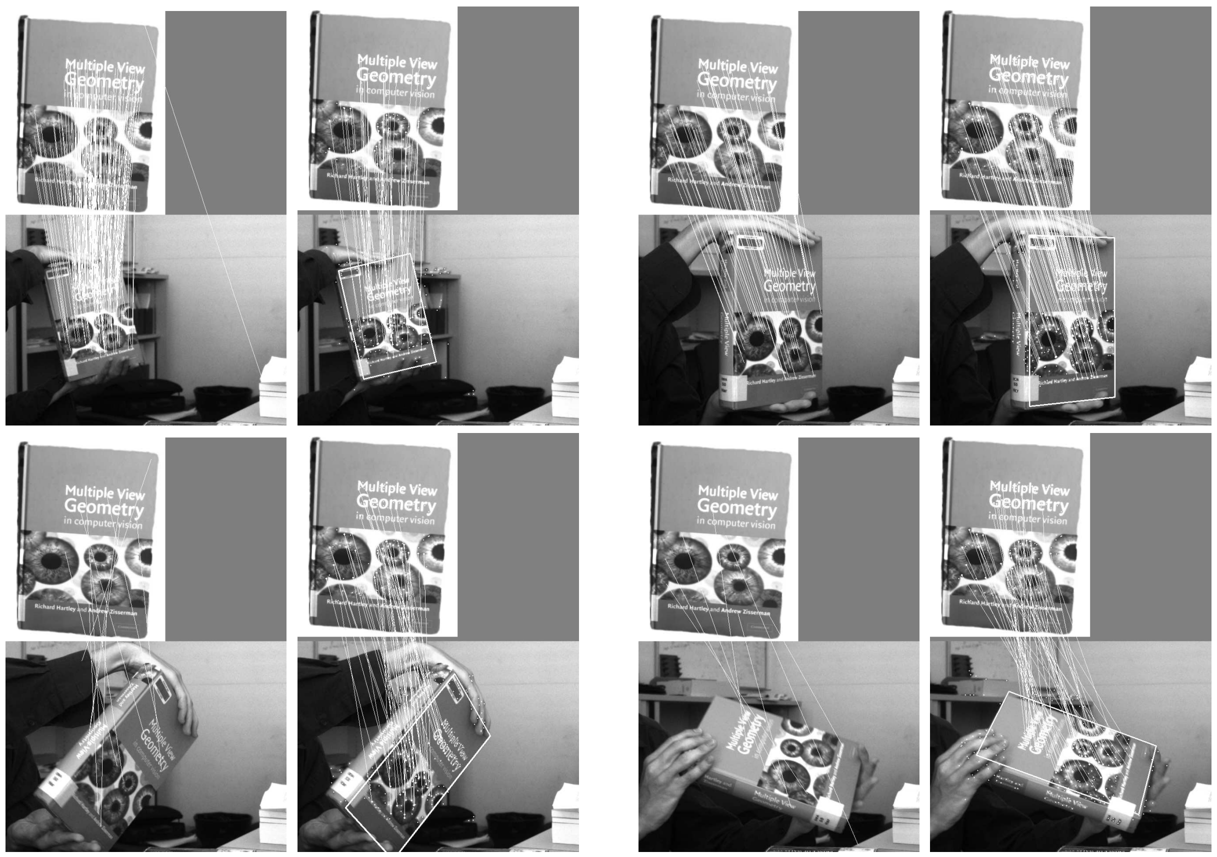


Fig. 16. Comparison with SIFT (left image) and our approach (right image). When there is little distortion, the SIFT approach gives a few more matches, but when too much perspective distorts the object image, it gives only few matches, while our approach is not perturbed.

significantly increase the matching time when approximate nearest-neighbor k-D tree search is used. It remains, however, less computationally effective than ours, which depends only on a limited number of pixel intensity comparisons.

As shown Figures 19 and 20, our method can also be used to detect deformable objects and estimate their deformation at 10 frames per second on a 2.8 GHz PC. As described in [24], objects are modeled by deformable meshes. The keypoints positions are expressed as weighted sums of the mesh vertices in the model image and change as the mesh is deformed. Fitting then amounts to minimizing a criterion that is the sum of two terms. The first is a robust estimate of the square distances of the keypoints in the model image to that of the corresponding ones in the input image. The second is a quadratic deformation energy [10]. This quadratic term

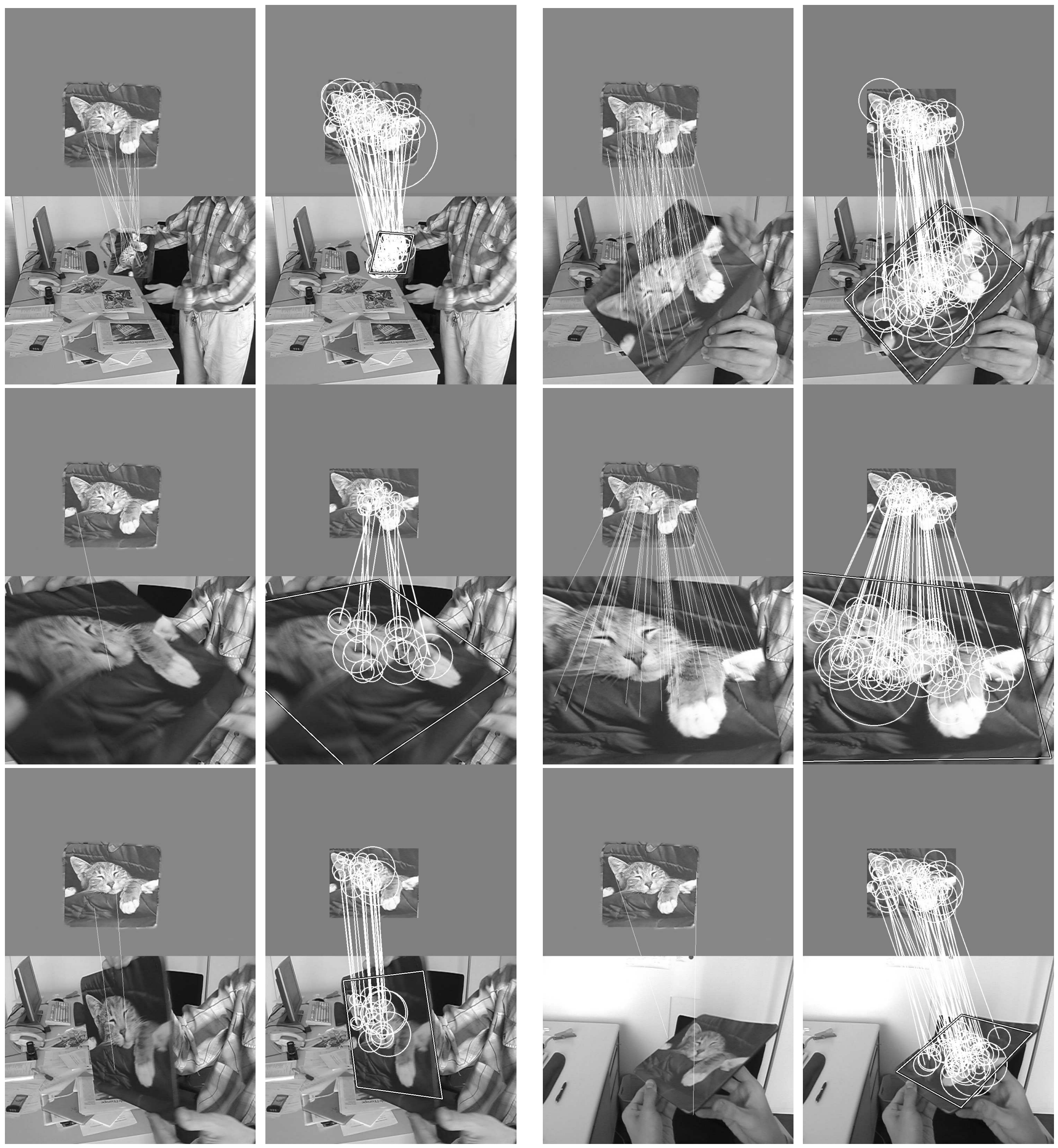


Fig. 17. Comparison with SIFT (left image) and our approach (right image) on the mouse pad sequence.

allows the use of a semi-implicit minimization scheme that converges even when the initial estimate is very far from the solution, which, in our context, is what happens when the object is severely deformed. When combined with an appropriately defined robust estimator for the keypoint distances and optimization schedule, this approach to minimization allows real-time detection in under 100 milliseconds while being robust to large deformations, severe occlusions, and changes in lighting.

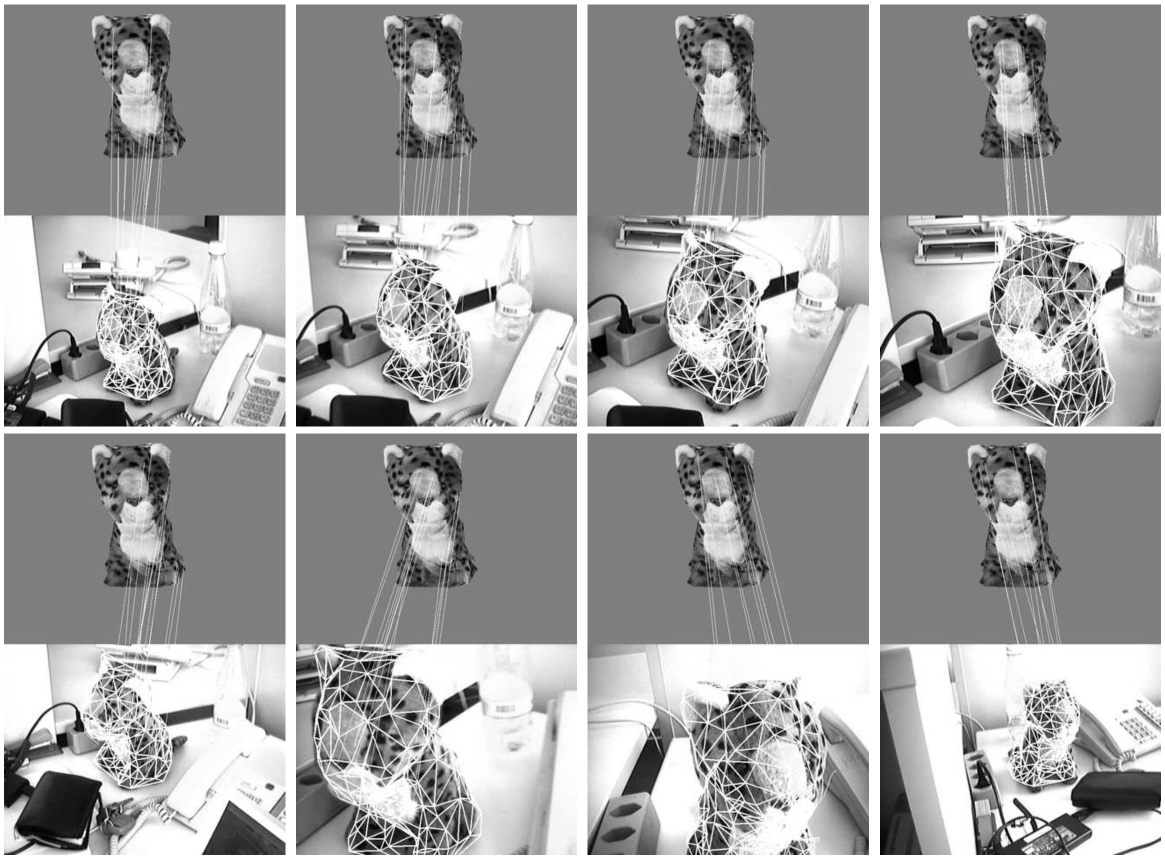


Fig. 18. Our method can take advantage of a 3–D model when available, to detect the target and estimate its pose from different viewpoints.

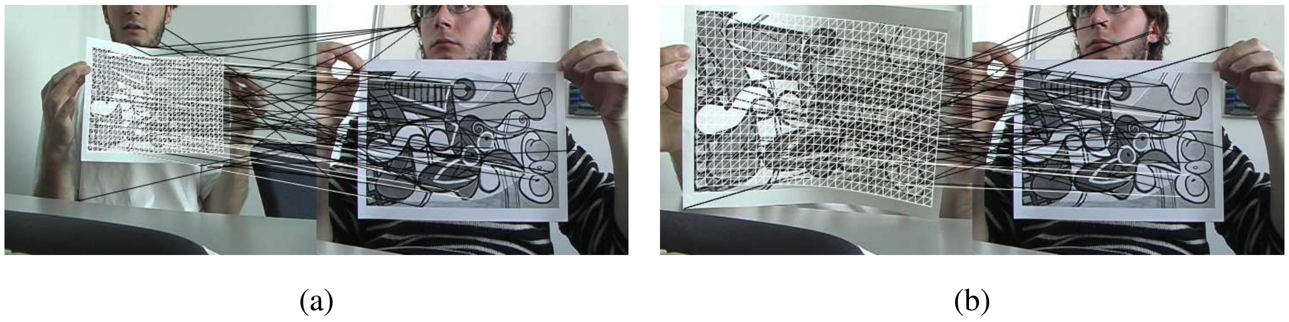


Fig. 19. Detecting a deforming sheet of paper at 10Hz on a 2.8 GHz PC. (a) The left image is the input image in which the sheet of paper is detected and its deformation estimated. The right one is the training image. The lines represent the matches recovered using our method. Black ones correspond to those estimated to be outliers and the white ones to inliers. (b) Same thing for a different input image.

VII. Conclusion and Perspectives

We have developed an approach to 3–D object-detection and pose estimation that relies on statistical learning techniques to perform much of the required computation during a training

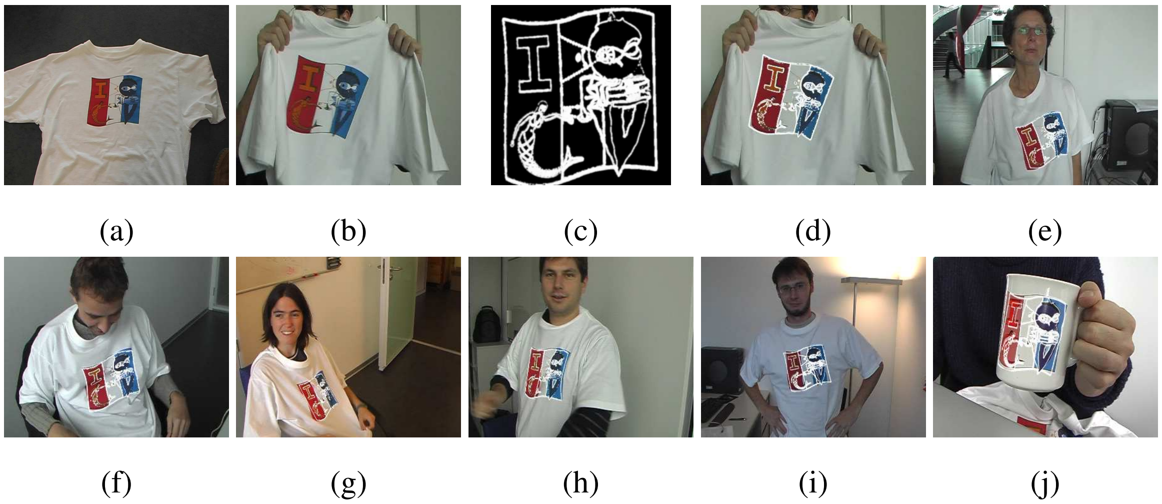


Fig. 20. Detecting a deforming T-Shirt. (a) Model image used for training purposes. (b) To illustrate the mapping our algorithm computes, we find the contours of the model using a simple gradient operator and we use them as a validation texture (c) which is overlaid on the input image using the recovered transformation (d). Additional results are obtained in different conditions (e to j). Note that in all cases, including the one where the T-shirt is replaced by a cup (j), the white outlines project almost exactly at the right place, thus indicating a correct registration and shape estimation.

phase so that, at run-time, we can achieve both speed and reliability. Like many others, our approach relies on matching keypoints extracted from one or more training images of the target object against those extracted from input images. What is new is to have reformulated the wide-baseline matching problem, which this involves, as a classification problem that can be effectively solved using randomized trees that are both fast and easily implemented.

We have shown that very good results can be obtained using a very simple approach to building the trees. More sophisticated building methods yield slight performance increases, but at the cost of much larger requirements both in terms of computation and memory. In practice, given the fact that we use robust techniques to estimate 3–D poses from the matches, we have not found it worthwhile to impose that extra burden as the simpler techniques are sufficient to achieve robust frame-rate detection and pose estimation. However, we could easily replace individual components of our approach, such as the keypoint detector, by improved or more generic ones as they become available without changing its overall philosophy.

Our approach is well adapted to cases where one or more images of the target object are available for off-line training purposes. If the object is either planar or nearly so, the images can be directly used without any preprocessing. If the object is fully 3–D, the images can be used to build the rough 3–D model required to train the system. Our next step is to speed up the training procedure itself so that it can also become an in-line process, thereby removing the major limitation of our approach when compared to state-of-the-art ones that do not require any *a priori* training. One obvious way to achieve this is to first track the target object under favorable conditions using standard techniques, which will allow us to follow keypoints across frames and to incrementally build the view-sets and the corresponding trees. This will constitute a starting point for our future investigations.

ACKNOWLEDGEMENTS

The authors would like to thank François Fleuret for suggesting the use of randomized trees and for his insightful comments, and Tom Drummond for discussions on keypoint detection.

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*, 9(7):1545–1588, 1997.

[3] A. Baumberg. Reliable Feature Matching across Widely Separated Views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[4] J. Beis and D.G. Lowe. Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Puerto Rico, 1997.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[6] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH*, July 1998.

[8] R. Fergus, P. Perona, and A. Zisserman. A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition. In *Conference on Computer Vision and Pattern Recognition*, July 2005.

[9] F. Fleuret and D. Geman. Coarse-To-Fine Visual Selection. *International Journal of Computer Vision*, 41(1):85–107, January 2001.

[10] P. Fua. *RADIUS: Image Understanding for Intelligence Imagery*, chapter Model-Based Optimization: An Approach to Fast, Accurate, and Consistent Site Modeling from Imagery. Morgan Kaufmann, 1997. O. Firschein and T.M. Strat, Eds.

[11] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Conference on Computer Vision and Pattern Recognition*, pages 111–119, 2000.

[12] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. In *Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

[13] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270, 1994.

[14] D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[15] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[16] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Conference on Computer Vision and Pattern Recognition*, 2005.

[17] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, pages 384–393, London, UK, September 2002.

[18] J. Meltzer, M.-H. Yang, R. Gupta, and S. Soatto. Multiple View Feature Descriptors from Image Sequences via Kernel Principal Component Analysis. In *European Conference on Computer Vision*, pages 215–227, may 2004.

[19] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002. Copenhagen.

[20] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. In *Conference on Computer Vision and Pattern Recognition*, pages 257–263, June 2003.

[21] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Accepted to International Journal of Computer Vision*, 2005.

[22] F. Mindru, T. Moons, and L. VanGool. Recognizing color patterns irrespective of viewpoint and illumination. In *Conference on Computer Vision and Pattern Recognition*, pages 368–373, 1999.

[23] S. K. Nayar, S. A. Nene, and H. Murase. Real-Time 100 Object Recognition System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1198, 1996.

[24] J. Pilet, V. Lepetit, and P. Fua. Real-Time Non-Rigid Surface Detection. In *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[25] D. Pritchard and W. Heidrich. Cloth motion capture. In *Eurographics*, volume 22(3), pages 263–271, September 2003.

[26] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *International Conference on Computer Vision*, Beijing, China, October 2005.

[27] F. Schaffalitzky and A. Zisserman. Multi-View Matching for Unordered Image Sets, or "How Do I Organize My holiday Snaps?". In *Proceedings of European Conference on Computer Vision*, pages 414–431, 2002.

[28] C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.

[29] T. Tuytelaars and L. VanGool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *British Machine Vision Conference*, pages 412–422, 2000.

[30] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.