Randomized Trees for Real-Time Keypoint Recognition

Vincent Lepetit Pascal Lagger Pascal Fua
Computer Vision Laboratory
École Polytechnique Fédérale de Lausanne (EPFL) 1015 Lausanne, Switzerland
Email: {Vincent.Lepetit, Pascal.Lagger, Pascal.Fua}@epfl.ch

Abstract

In earlier work, we proposed treating wide baseline matching of feature points as a classification problem, in which each class corresponds to the set of all possible views of such a point. We used a K-mean plus Nearest Neighbor classifier to validate our approach, mostly because it was simple to implement. It has proved effective but still too slow for real-time use.

In this paper, we advocate instead the use of randomized trees as the classification technique. It is both fast enough for real-time performance and more robust. It also gives us a principled way not only to match keypoints but to select during a training phase those that are the most recognizable ones. This results in a real-time system able to detect and position in 3D planar, non-planar, and even deformable objects. It is robust to illuminations changes, scale changes and occlusions.

1. Introduction

Wide baseline keypoint matching has proved to be an effective tool for applications ranging from camera registration to object detection. Since the pioneering work by Schmid and Mohr [1], the algorithms have become ever more robust to scale, viewpoint, illumination changes and partial occlusions [2, 3, 4, 5, 6, 7].

These wide baseline matching methods, however, are typically designed to match two images but do not take advantage of the fact that, for object detection and pose estimation purposes, both a 3D object model and several training images may be available. As shown in Figs 1 and 2, our goal is to incorporate this additional information into a keypoint recognizer that is both robust and fast enough for real-time object detection, whether or not the object is planar.

The key ingredient of our approach is to treat wide baseline matching of feature points as a classification problem, in which each class corresponds to the set of all possible views of such a point. During training, given at least one image of the target object, we synthesize a large number of views of individual keypoints. If the object can be assumed to be locally planar, this is done by simply warping image patches around the points under affine or homographic deformations. Otherwise, given the 3D model, we use standard Computer Graphics texture-mapping techniques. This second approach relaxes the planarity assumptions. At run-time, we can then use a powerful and fast classification technique to decide to which view set, if any, an observed feature belongs. This method is as effective and much faster than the usual way of computing local descriptors and comparing their responses. Once potential correspondences have been established between the interest points of the input image and those lying on the object, we apply a standard RANSAC-based method to estimate the 3D pose.

In previous work [8], we used a K-mean plus Nearest Neighbor classifier to validate our approach, mostly because it was simple to implement. It has proved effective but still too slow for real-time use. Here, we advocate instead the use of randomized trees [9] as the classification technique. It is both faster and more robust, at the possible expense of additional training time. Furthermore, it also gives us a principled way, not only to recognize keypoints, but also to select during the training phase those that yield the best recognition rate. As a result, even though we use a monoscale algorithm for keypoint extraction [10], we can achieve scale-invariance across a range of scales by using training images at different resolutions and retaining only those keypoints that are stable within the range.

In short, the contribution of this paper is not only a faster algorithm but also one that is more robust through the appropriate selection of keypoints to be recognized.

In the remainder of the paper, we first discuss related work and recall how wide baseline matching can be stated as a classification problem. We then present the proposed keypoint selection method, detail our new classification method based on randomized trees, and comment the results.

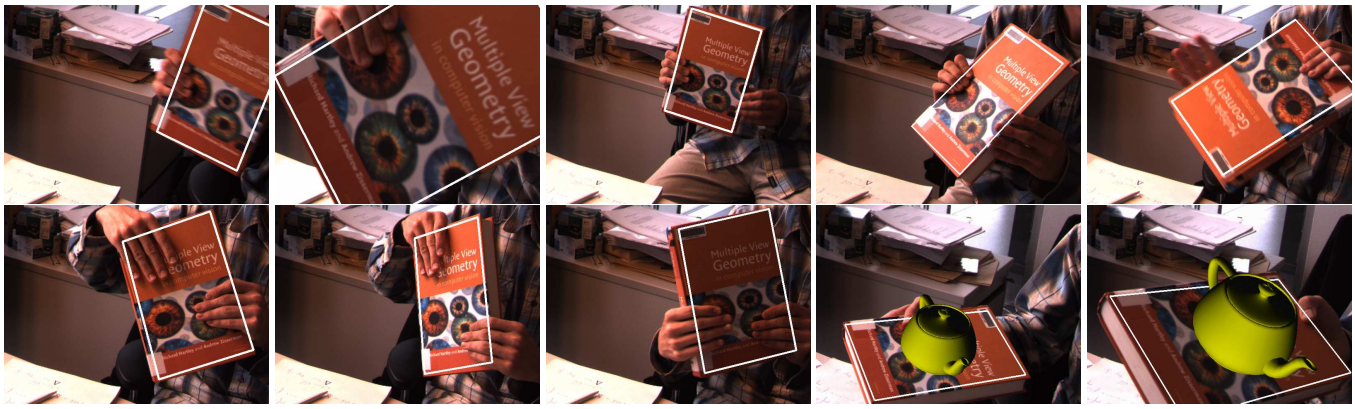

Figure 1. Detection of a book in a video sequence: The book is detected independently and successfully in all subsequent frames at 25Hz in 640×480 images on a standard PC, in spite of partial occlusion, cluttered background, motion blur, large illumination and pose changes. In the last two frames, we add the inevitable virtual teapot to show we also recover 3D pose. A video sequence is available at http://cvlab.epfl.ch/research/augm/detect.html

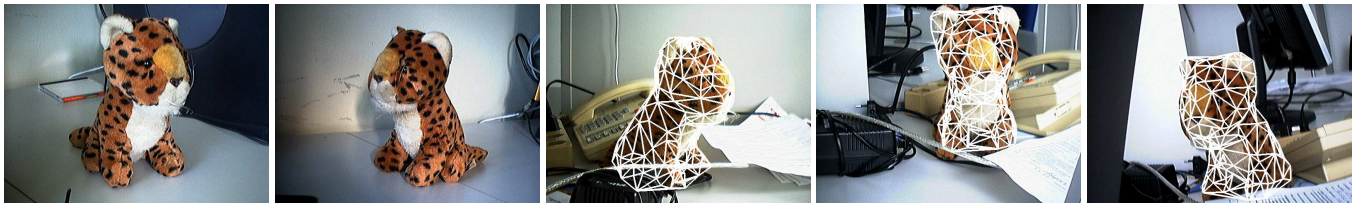

Figure 2. The method is just as effective for 3D objects. In this experiment, we detected the teddy tiger using a 3D model reconstructed from several views such as the two first images on the left.

2. Related Work

In the area of automated 3D object detection, we can distinguish between "Global" and "Local" approaches.

Global ones 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 [11] to more sophisticated ones such as AdaBoost and classifiers cascade to achieve real-time detection of human faces at varying scales [12]. Such approaches, however, are not particularly good at handling occlusions, cluttered backgrounds, or the fact that the pose of the target object may be very different from those in the training set. Furthermore, these global methods cannot provide accurate 3D pose estimation.

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. For local approaches to be effective, feature point extraction and characterization should be insensitive to viewpoint and illumination changes. Scale-invariant feature extraction can be achieved by using Harris detector [13] at several Gaussian derivative scales, or by considering local optima of pyramidal difference-of-Gaussian filters in scale-space [7]. Mikolajczyck et al. [4] have also defined an affine invariant point detector to handle larger viewpoint changes, that has been used for 3D object recognition [14], but it relies on an iterative estimation that would be too slow for our purposes.

Given the extracted feature points, various local descriptors have been proposed: Schmid and Mohr [1] compute rotation invariant descriptors as functions of relatively high order image derivatives to achieve orientation invariance. Baumberg [3] uses a variant of the Fourier-Mellin transformation to achieve rotation invariance. He also gives an algorithm to remove stretch and skew and obtain an affine invariant characterization. Allezard et al. [15] represent the keypoint neighborhood by a hierarchical sampling, and rotation invariance is obtained by starting the circular sampling with respect to the gradient direction. Tuytelaars and al. [2] fit an ellipse to the texture around local intensity extrema to obtain correspondences remarkably robust to viewpoint changes. Lowe [7] introduces a descriptor called SIFT based on several orientation histograms, that is not fully

affine invariant but tolerates significant local deformations. Most of these methods are too slow for real-time processing, except for [5] that introduces Maximally Stable Extremal Regions to achieve near frame rate matching of stable regions. By contrast, our classification-based method runs easily at frame rate, because it shifts much of the computational burden to a training phase and, as a result, reduces the cost of online matching while increasing its robustness.

Classification as a technique for wide baseline matching has also been explored by [16] in parallel to our previous work. In this approach, the training set is iteratively built from incoming frames, and kernel PCA is used for classification. While this is interesting for applications when a training stage is not possible, our own method allows to detect the object under unseen positions since we synthesize new views. The classification method described in this paper also has a lower complexity than their approach.

3. Keypoint Matching as Classification

Let us first recall how matching keypoints found in an input image against keypoints on a target object $\mathbf{O}$ can be naturally formulated as a classification problem [8]. During training, we construct a set $\mathbf{K} = \{\mathbf{k}_1 \dots \mathbf{k}_N\}$ of N prominent keypoints lying on the object. At runtime, given an input patch $\mathbf{p}(\mathbf{k}^{\text{input}})$ centered at a keypoint $\mathbf{k}^{\text{input}}$ extracted in the input image, we want to decide whether or not it can be an view of one of the N keypoints $\mathbf{k}_i$. In other words, we want to assign to $\mathbf{p}$ a class label $Y(\mathbf{p}) \in \mathbf{C} = \{-1, 1, 2, \dots, N\}$, where the -1 label denotes all the points that do not belong to the object. Y cannot be directly observed and we aim at constructing a classifier $\hat{Y}$ such as $P(Y \neq \hat{Y})$ is small.

In other recognition tasks, such as face 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 to train our classifier: This approach gives us a virtually infinite training set to perform the classification.

For each keypoint, we can then constitute a sampling of its *view set*, that is the set of all its possible appearances under different viewing conditions. This sampling allows us to use statistical classification techniques to learn them during an offline stage, and, finally, to perform the actual classification at run-time. This gives us a set of matches that lets us estimate the pose.

Figure 3. The most stable keypoints selected by our method on the book cover and the teddy tiger.

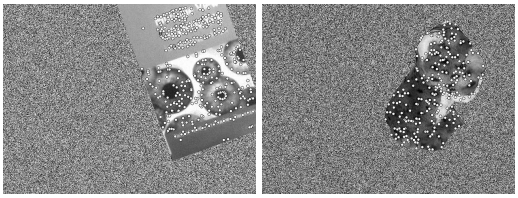

Figure 4. An example of generated views for the book cover and the teddy tiger, and the extracted keypoints for these views.

4. Building the Training Set

In [8], we built the view sets by first extracting the keypoints $\mathbf{k}_i$ in the given original images then generating new views of each keypoint independently. As depicted in Fig.4, it is more effective to generate new views of the whole object, and extract keypoints in these views. This approach allows us to solve in a simple way several fundamental problems at no additional computation cost at run-time: We can easily determine stable keypoints under noise and perspective distortion, which helps making the matching robust to noise and cluttered background.

4.1. Local Planarity Assumptions

If the object can be assumed to be locally planar, a new view can be synthesized by warping a training image of the object using an affine transformation that approximates the actual homography. The affine transformations 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. In this paper, we use a random sampling of the affine transformations space, the angles θ and ϕ varying in the range $[-\pi; +\pi]$, and the scales λ_1 and λ_2 varying in the range $[0.2; 1.8]$. Those ranges are much larger than the ones used in our earlier work, and can be handled because we now determine the most stable points and thanks to our new classification method.

4.2. Relaxing the Planarity Assumptions

One advantage of our approach is that we can exploit the knowledge of a 3D model if available. Such a model is very useful to capture complex appearance changes due to changes in the pose of a non convex 3D object, including occlusions and non-affine warping. Given the 3D model, we use standard Computer Graphics texture-mapping techniques to generate new views under perspective transformations.

In the case of the teddy tiger of Fig.2, we used Image Modeler[1] to reconstruct its 3D model. An automated reconstruction method could also have been used, another alternative would have been to use image-based rendering techniques to generate the new views.

4.3. Keypoint Selection

We are looking for a set of keypoints $\mathbf{K} = \{\mathbf{k}_i\}$ lying on the object to detect, and expressed in a reference system related to this object. We should retain the keypoints with a good probability $P(\mathbf{k})$ to be extracted in the input views at run-time.

4.3.1. Finding Stable Keypoints

Let $\mathcal{T}$ denote the geometric transformation used to generate a new view, and $\widetilde{\mathbf{k}}$ a keypoint extracted in this view. $\mathcal{T}$ is an affine transformation, or a projection if the 3D model is available. By applying $\mathcal{T}^{-1}$ to $\widetilde{\mathbf{k}}$, we can recover its corresponding keypoint $\mathbf{k}$ in the reference system. Thus, $P(\mathbf{k})$ can be estimated for keypoints lying on the objects from several generated views. The set $\mathbf{K}$ is then constructed by retaining keypoints $\mathbf{k}_i$ with a high $P(\mathbf{k}_i)$. In our experiments, we retain the 200 first keypoints according to this measure. Fig. 3 shows the keypoints selected on the book cover and the teddy tiger.

The training set for keypoint $\mathbf{k}_i$ is then built by collecting the neighborhood $\mathbf{p}$ of the corresponding $\widetilde{\mathbf{k}}$ in the generated images, as shown in Figs. 5 and 6.

4.3.2. Robustness to Image Noise

When a keypoint is detected in two different images, its precise location may shift a bit due to image noise or viewpoint changes. In practice, such a positional shift results in large errors of direct cross-correlation measures. One solution is to iteratively refine the point localization [4], which can be costly.

In our method, this problem is directly handled by the fact that we extract the keypoints $\widetilde{\mathbf{k}}$ in the synthesized views: These images should be as close as possible to actual images captured from a camera, and we add white noise to the generated views. To simulate a cluttered background, the new object view is rendered over a complex random background. That way, the system is trained with images similar to those at run-time.

[1] ImageModeler is a commercial product from Realviz$^{(tm)}$ that allows 3D reconstruction from several views with manual intervention.

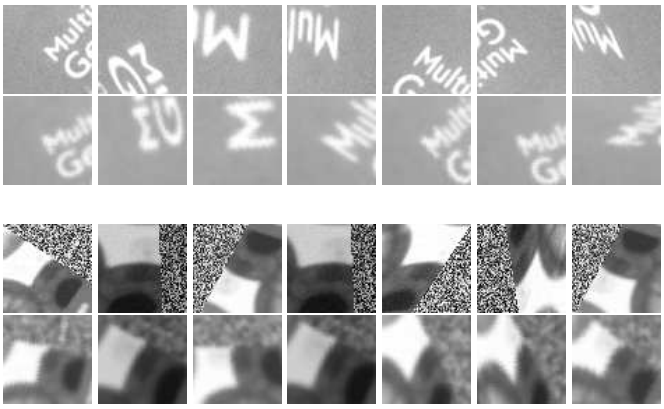

Figure 5. First row: Patches centered at a keypoint extracted in several new views, synthesized using random affine transformations and white noise addition. Second row: Same patches after orientation correction and Gaussian smoothing. These preprocessed patches are used to train the keypoint classifier. Third and fourth rows: Same as before for another keypoint located on the border of the book.

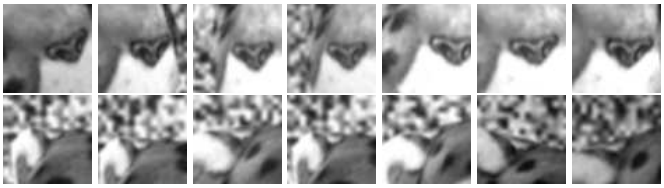

Figure 6. In the case of the teddy tiger, we restricted the range of acceptable poses and the orientation correction was not used.

5. Keypoint Recognition

In [8], we used a K-mean plus Nearest Neighbor classifier to validate our approach, because it is simple to implement and it gives good results. Nevertheless, such classifier is known to be one of the less efficient classification methods. We show in this section that randomized trees are better suited to our keypoint recognition problem, because they allow very fast recognition and they naturally handle multi-class problems.

5.1. Randomized Trees

Randomized trees are simple but powerful tools for classification, introduced and applied to recognition of handwritten digits in [9]. [17] also applied them to recognition

of 3–D objects. We quickly recall here the principles of randomized trees for the unfamiliar reader. As depicted by Fig. 7, each non-terminal node of a tree contains a simple test that splits the image space. In our experiments, we use tests of the type: "Is this pixel brighter than this one ?". Each leaf contains an estimate based on training data of the conditional distribution over the classes given that an image reaches that leaf. A new image is classified by dropping it down the tree, and, in the one tree case, attributing it the class with the maximal conditional probability stored in the leaf it reaches.

We construct the trees in the classical, top-down manner, where the tests are chosen by a greedy algorithm to best separate the given examples, according to the expected gain of information. The process of selecting a test is repeated for each non-terminal descendant 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.

Since the numbers of classes, training examples and possible tests are large in our case, building the optimal tree becomes quickly intractable. Instead we grow multiple, randomized trees: For each tree, we retain a small random subset of training examples and only a limited random sample of tests at each node, to obtain weak dependency between the trees. More details about the trees construction can be found in [10].

5.2. Preprocessing

In order to make the classification task easier, the patches $\mathbf{p}$ of the training set or at run-time are preprocessed to remove some variations within the classes attributable to perspective and noise.

The generated views are first smoothed using a Gaussian filter. We also use the method of [7] to attribute a 2D orientation to the keypoints and achieve some normalization. The orientation is estimated from the histogram of gradient directions in a patch centered at the keypoint. Note that we do not require a particularly stable method, since the same method is used for training and run-time recognition. We just want it to be reliable enough to reduce the variability within the same class. Once the orientation of an extracted keypoint is estimated, its neighborhood is rectified as shown Fig. 5.

Illumination changes are usually handled by normalizing the views intensities in some way, for example by normalizing by the L2 norm of the intensities. We show below that our randomized trees allow to skip this step. The classification indeed relies on tests comparing intensities of pixels. This avoids the use of an arbitrary normalization method and makes the classification very robust to illumination changes.

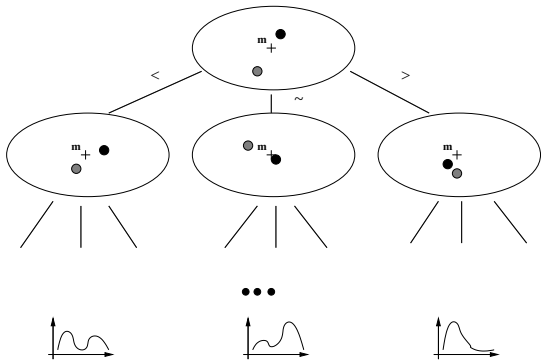

Figure 7. Type of tree used for keypoint recognition. The nodes contain tests comparing two pixels in the keypoint neighborhood; the leaves contain the d_l posterior distributions.

5.3. Node Tests

In practice, we use ternary tests based on the difference of intensities of two pixels taken in the neighborhood of the keypoint:

$$\begin{array}{lll} \text{If } I(\mathbf{p},\mathbf{m_1}) - I(\mathbf{p},\mathbf{m_2}) & < -\tau & \text{go to child 1;} \\ \text{If } |I(\mathbf{p},\mathbf{m_1}) - I(\mathbf{p},\mathbf{m_2})| & \leq +\tau & \text{go to child 2;} \\ \text{If } I(\mathbf{p},\mathbf{m_1}) - I(\mathbf{p},\mathbf{m_2}) & > +\tau & \text{go to child 3.} \end{array}$$

$I(\mathbf{p},\mathbf{m})$ is the intensity of patch $\mathbf{p}$ after the preprocessing step described in Section 5.2, at pixel location $\mathbf{m}$. $\mathbf{m}_1$ and $\mathbf{m}_2$ are two pixel locations chosen to optimize the expected gain of information as described above. τ is a threshold deciding in which range two intensities should be considered as similar. In the results presented in this paper, we take τ to be equal to 10.

This test is very simple and requires only pixel intensities comparisons. Nevertheless, because of the efficiency of randomized trees, it yields reliable classification results. We tried other tests based on weighted sums of intensities *a la* Adaboost, on gradients or on Haar wavelets without significant improvements on the classification rate.

5.4. Run Time Keypoint Recognition

Once the randomized trees $T_1, \ldots, T_L$ are built, the posterior distributions $P(Y = c|T = T_l, \text{ reached leaf } = \eta)$ can be estimated for each terminal node η from the training set. At runtime, the patches $\mathbf{p}$ centered at the keypoints extracted in the input image are preprocessed and dropped down the trees. Following [9], if $d_l(\mathbf{p})$ denotes the posterior distribution in the node of tree T_l reached by a patch $\mathbf{p}$, $\mathbf{p}$ is classified considering the average of the distributions $d_l(\mathbf{p})$:

$$\hat{Y}(\mathbf{p}) = \underset{c}{\text{argmax}}\, d^c(\mathbf{p}) = \underset{c}{\text{argmax}}\, \frac{1}{L} \sum_{l=1\ldots L} d_l(\mathbf{p})$$

$d^c(\mathbf{p})$ is the average of the posterior probabilities of class c and constitutes a good measure of the match confidence.

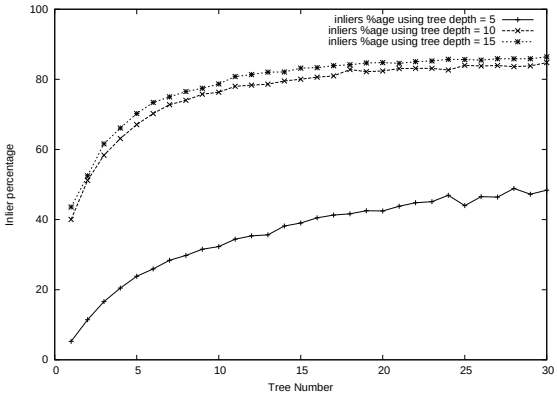

Figure 8. Percentage of correctly classified views with respect to the number of trees, using trees of depth 5, 10, and 15.

We can estimate during training a threshold D_c to decide if the match is correct or not with a given confidence s:

$$P(Y(\mathbf{p}) = c|\hat{Y}(\mathbf{p}) = c, d^c(\mathbf{p}) > D_c) > s$$

In practice we take $s = 90\%$. Keypoints giving $d^c(\mathbf{p})$ lower than D_c are considered as keypoints detected on the background, or misclassified keypoints, and are therefore rejected. It leaves a small number of outlier matches, and the pose of the object is found by RANSAC after few iterations.

5.5. Performance

The correct classification rate $P(\hat{Y}(\mathbf{p}) = c|Y(\mathbf{p}) = c)$ of our classifier can then be measured using new random views. The graph of Fig. 8 represents the percentage of keypoints correctly classified, with respect to the number of trees, for several maximal depths for the trees. The graph shows no real differences between taking trees of depth 10 or 15, so we can use trees with limited depth. It also shows that 20 trees are enough to reach a recognition rate of 80%. Growing 20 trees of depth 10 takes about 15 minutes on a standard PC.

Since the classifier works by combining the responses of sub-optimal trees, we tried to re-use trees grown for a first object for another object, as shown Fig. 9: We updated the posterior probabilities in the terminal nodes, but kept the same tests in the non-terminal nodes. We experienced a slight drop of performance, but not enough to prevent the system from recognizing the new object. In this case, the time required for training drops to less than one minute.

6. Results

6.1. Planar Objects

We first tried our algorithm on planar objects. Fig. 11 shows matches between the training image and input images estab-

Figure 9. Re-usability of the set of trees: A new object is presented to the system, and the posterior probabilities are updated. The new object can then be detected.

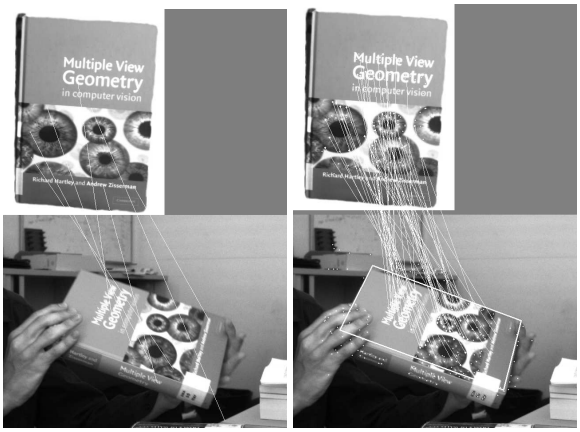

Figure 10. Comparison with SIFT. When too much perspective distorts the object image, the SIFT approach gives only few matches (left), while our approach is not perturbed (right).

lished in real-time. The estimated pose is then accurate and stable enough for Augmented Reality as shown Fig. 1.

We compared our results with those obtained using the executable that implements the SIFT method [7] kindly provided by David Lowe. As shown in Fig. 10, when too much perspective distorts the object view, this method gives only few matches, while ours is not perturbed. Ours is also much faster. For a fair comparison, remember that we take advantage of a training stage possible in object detection applications, while the SIFT method can also be used to match two given images. Another difference is that we cannot handle

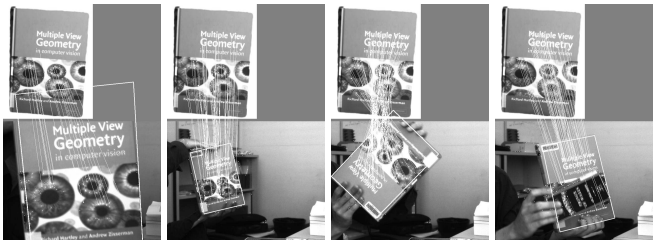

Figure 11. Detection of the book: Inlier matches established in real-time under several poses.

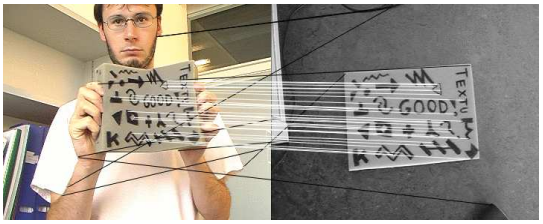

Figure 12. Deformable objects: the object is detected and its deformation estimated, using the method described in [18].

as much scale changes as SIFT because we do not use (yet) a multi-scale keypoint detection.

6.2. Detecting a 3D Object

Fig. 2 shows results of the detection of a teddy tiger. As mentioned above, its 3D textured model was reconstructed from several views with the help of ImageModeler. It can be detected from different sides, and front and up views.

6.3. Detecting Deformable Objects

Our method is also used in [18] to detect deformable objects and estimate their deformation in real-time. The matches are used not only to detect but also to compute a precise mapping from a model image to the input image as shown Fig. 12.

7. Conclusion and Perspectives

We proposed an approach to keypoint matching for object pose estimation based on classification. We showed that using randomized trees yields a powerful matching method well adapted to object detection.

Our current approach to keypoint recognition relies on comparing pixel values in small neighborhoods around these keypoints. It works very well for textured objects, but loses its effectiveness in the absence of texture. To increase the range of applicability of our approach, we will investigate the use of other additional image features, such as spread gradient [19]. We believe our randomized tree approach to keypoint matching to be ideal to find out those that are most informative in any given situation and, thus, to allow us to mix different image features in a natural way.

Acknowledgements

The authors would like to thank François Fleuret for suggesting to use randomized trees and for his insightful comments.

References

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

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

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

[4] K. Mikolajczyk and C. Schmid, "An Affine Invariant Interest Point Detector," in *European Conference on Computer Vision*. 2002, pp. 128–142, Springer, Copenhagen.

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

[6] 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*, 2002, pp. 414–431.

[7] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 20, no. 2, pp. 91–110, 2004.

[8] 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.

[9] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[10] V. Lepetit and P. Fua, "Towards Recognizing Feature Points using Classification Trees ," Technical Report IC/2004/74, EPFL, 2004.

[11] S. K. Nayar, S. A. Nene, and H. Murase, "Real-Time 100 Object Recognition System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1186–1198, 1996.

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

[13] C.G. Harris and M.J. Stephens, "A Combined Corner and Edge Detector," in *Fourth Alvey Vision Conference, Manchester*, 1988.

[14] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3D Object Modeling and Recognition using Affine-Invariant Patches and Multi-View Spatial Constraints," in *Conference on Computer Vision and Pattern Recognition*, June 2003.

[15] N. Allezard, M. Dhome, and F. Jurie, "Recognition of 3D Textured Objects by Mixing View-Based and Model-Based Representations," in *International Conference on Pattern Recognition*, Barcelona, Spain, Sep 2000, pp. 960–963.

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

[17] B. Jedynak and F. Fleuret, "Reconnaissance d'objets 3D à l'aide d'arbres de classification," in *Proceedings of Image'Com*, 1996.

[18] 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.

[19] Y. Amit, D. Geman, and X. Fan, "A coarse-to-fine strategy for multiclass shape detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.