Towards Recognizing Feature Points using Classification Trees

(EPFL Technical Report IC/2004/74)

Vincent Lepetit Pascal Fua
Computer Vision Laboratory
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
Email: {Vincent.Lepetit, Pascal.Fua}@epfl.ch

September 15, 2004

Abstract

In earlier work [Lepetit et al.*, 2004], we proposed to treat wide baseline matching of feature points as a classification problem and proposed an implementation based on K-means and nearest neighbor classification. We showed that this method is both reliable and faster than competing methods, but still too slow for real-time implementations. Here we show that using decision trees instead speeds up the computation greatly, while increasing the robustness. This allows point matching under large viewpoint and illumination changes that is suitable for accurate object pose estimation at 25 Hz on a standard Pentium IV PC.*

Most of the previous methods rely either on using ad hoc *local descriptors or on estimating local affine deformations. By contrast, we treat wide baseline matching of keypoints as a classification problem, in which each class corresponds to the set of all possible views of such a point. Given one or more images of a target object, we train the system by synthesizing a large number of views of individual keypoints and by using statistical classification tools to produce a compact description of this view set. At run-time, we rely on this description to decide to which class, if any, an observed feature belongs. This formulation allows us to use decision trees to reduce matching error rates, and to move some of the computational burden from matching to training, which can be performed beforehand.*

We will show that our method is both reliable and fast enough to detect and estimate in real-time the 3D pose of an object in the presence of occlusions, illumination changes, and cluttered backgrounds.

1 Introduction

While there are many effective approaches to tracking, they all require an initial pose estimate, which remains difficult to provide automatically, fast and reliably. Among methods that can be used for this purpose, those based on feature point matching have become popular since the pioneering work of Schmid and Mohr [Schmid and Mohr, 1997] because this approach appears to be more robust to scale, viewpoint, illumination changes and partial occlusions than edge or eigen-image based ones. Recently, impressive wide-baseline matching results have been obtained [Tuytelaars and VanGool, 2000, Baumberg, 2000, Mikolajczyk and Schmid, 2002, Schaffalitzky and Zisserman, 2002, Lowe, 2004], which make this approach even more attractive.

These wide baseline matching methods, however, are typically designed to match two images but not to take advantage of the fact that, for pose estimation purposes, both a 3D object model and several training images may be available. In this paper, we propose a method that allows us to use this additional information to build a keypoints recognizer with at a much reduced computational cost at run-time, without loss of matching performance. It also allows to relax the locally planar assumption.

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 is more complex to implement but relaxes the planarity assumptions. At run-time, we can then use powerful and fast classification techniques to decide to which view set, if any, an observed feature belongs, which 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 3D pose.

In previous work [Lepetit *et al.*, 2004], we used a K-mean plus Nearest Neighbour classifier to validate our approach, mostly because it is simple to implement. Here, we advocate the use of decision trees as the classification technique, that work faster and more robustly, at the possible expense of additional training time. In Figure 1, we show how it can be used to detect and estimate the pose of an object in real-time.

In the remainder of the paper, we first discuss related work. We then recall how wide baseline matching can be stated as a classification problem in Section 3 and detail our new classification method based on decision trees in Section 4. We also describe our own method for keypoint detection in Section 5. Quantitative and qualitative results are presented Section 6.

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 [Nayar *et al.*, 1996] to more sophisticated ones such as AdaBoost and classifiers cascade to achieve real-time detection of human faces at varying scales [Viola and Jones, 2001]. 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 the ones 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 [Jurie, 1999], 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 [Allezard *et al.*, 2000].

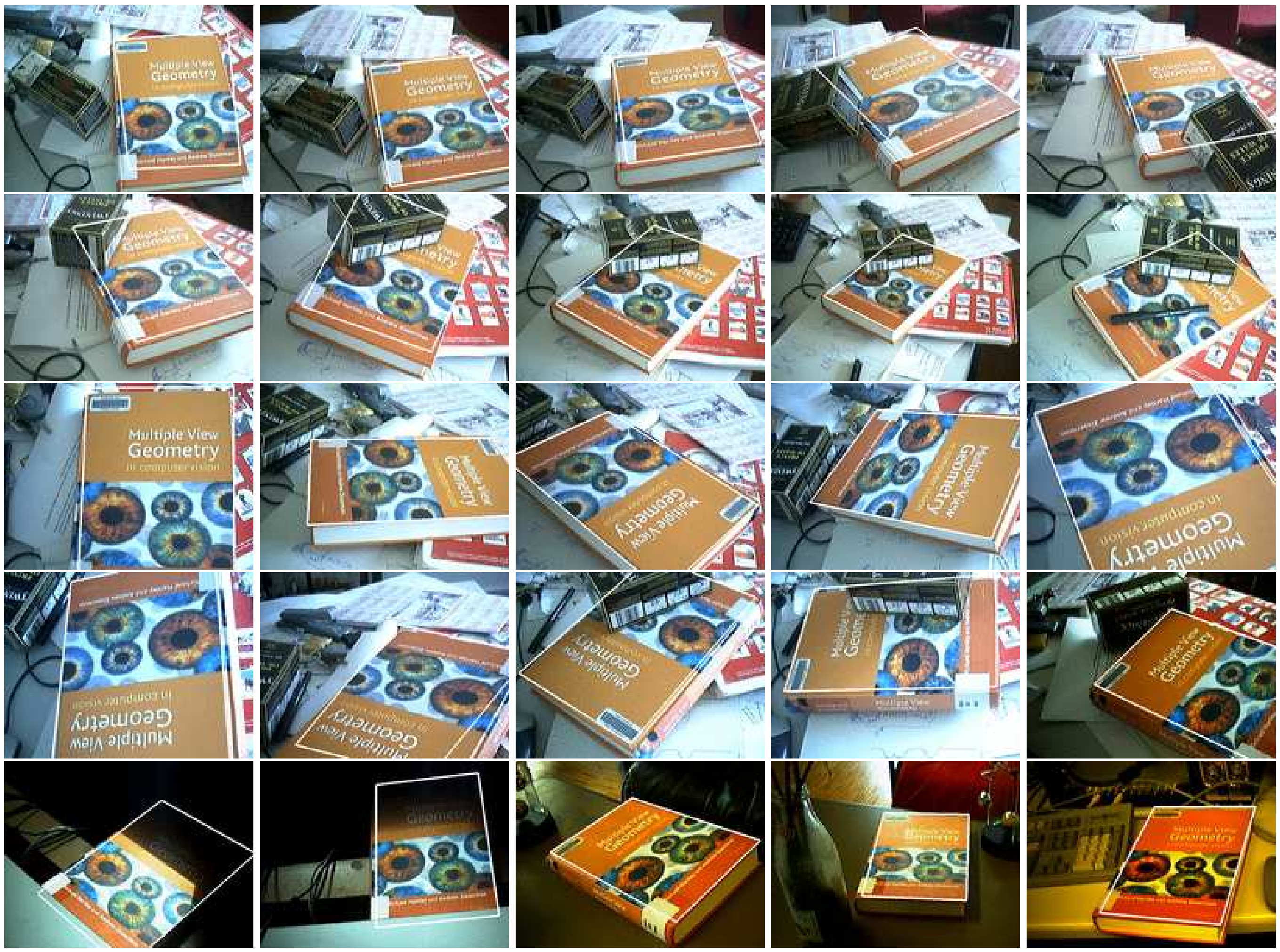

Figure 1: The book is detected and its 3D pose estimated at 25Hz, on a standard PC, in spite of partial occlusion, cluttered background, and large illumination and pose changes.

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 the Harris detector [Harris and Stephens, 1988] at several Gaussian derivative scales, or by considering local optima of pyramidal difference-of-Gaussian filters in scale-space [Lowe, 1999]. Mikolajczyck et al. [Mikolajczyk and Schmid, 2002] have also defined an affine invariant point detector to handle larger viewpoint changes, that have been used for 3D object recognition [Rothganger *et al.*, 2003], 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 [Schmid and Mohr, 1997] compute rotation invariant descriptors as functions of relatively high order image derivatives to achieve orientation invariance. Baumberg [Baumberg, 2000] 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. [Allezard *et al.*, 2000] represent the key point neighborhood by a hierarchical sampling, and rotation invariance is obtained by starting the circular sampling with respect to the gradient direction. Tuytelaars and al. [Tuytelaars and VanGool, 2000] fit an ellipse to the texture around local intensity extrema and use the Generalized Color Moments [Mindru *et al.*, 1999] to obtain correspondences remarkably robust to viewpoint changes. Lowe [Lowe, 2004] introduces a descriptor called SIFT based on several orientation histograms, that is not fully affine invariant but tolerates significant local deformations.

In short, local approaches have been shown to work well on highly textured objects, to handle partial occlusions, and to tolerate errors in the correspondences. However, even if they can be used for object detection and pose estimation, they rely on relatively expensive point matching between a sample and an input image. By contrast, our approach is geared towards shifting much of the computational burden to a training phase during which we build descriptors from the set of sample images and, as a result, reducing the cost of online matching while increasing its robustness.

3 Feature Point Matching as a Classification Problem

3.1 Approach

Matching interest points found in an input image against feature points on a target object $\mathbf{O}$ can be naturally formulated as a classification problem as follows. During training, we construct a set $\mathbf{F_O}$ of N prominent feature points lying on $\mathbf{O}$. Given an input patch $\mathbf{p} \in \mathbf{P}$, the space of all image patches of a given size, we want to decide whether or not it can be an image of one of the N interest points. In other words, we want to assign to $\mathbf{p}$ a 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 the object — the background. Y cannot be directly observed, and we aim to construct a classifier $\hat{Y} : \mathbf{P} \rightarrow \mathbf{C}$ such as $P(Y \neq \hat{Y})$ is small.

In other recognitions 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 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 feature points in $\mathbf{F_O}$ using simple rendering techniques to train our classifier.

For each feature point, this constitutes a sampling of its *view set*, that is the set of all its possible appearances under different viewing conditions. We can then 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 to estimate the pose.

3.2 Creating View Sets

Constructing the viewset of points is relatively simple, and we focus here on some of the implementation details that ensure invariance to illumination changes and also robustness to point localization error that can occur while extracting the feature points.

3.2.1 Construction Under Local Planarity Assumptions

For a given point in the training image, if the surface can be assumed to be locally planar, a new view of its neighborhood can be synthesized by warping using an affine transformation, that approximates the actual homography:

$$(\mathbf{n} - \mathbf{n}_0) = \mathbf{A}\,(\mathbf{m} - \mathbf{m}_0) + \mathbf{t} \qquad (1)$$

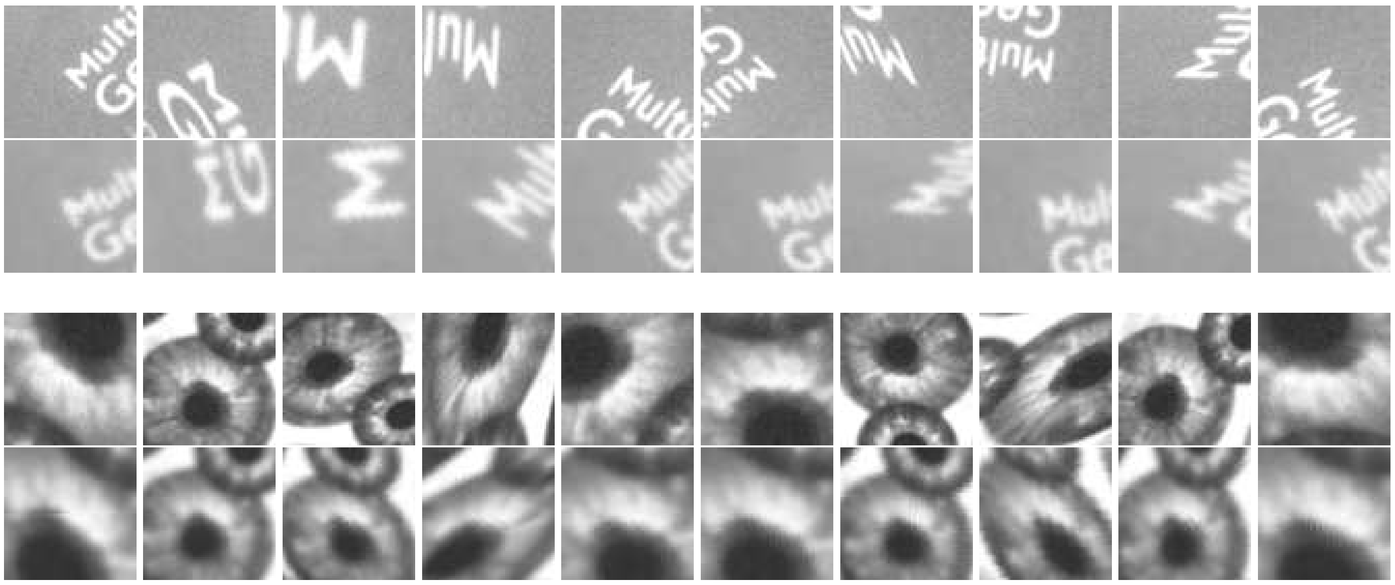
Figure 2: First row: New views of a keypoint detected in the training image of a book cover, synthesized using random affine transformations and white noise addition. Second row: Same views after orientation correction and Gaussian smoothing. These preprocessed views are used to train the keypoint classifier. Third and fouth rows: Same for another keypoint.

where $\mathbf{m}_0$ are the coordinates of the keypoint detected in the training image, $\mathbf{n}_0$ are the coordinates of the patch center, and $\mathbf{n}$ the new coordinates of the warped point $\mathbf{m}$. The matrix $\mathbf{A}$ 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; $\mathbf{t} = [t_u, t_v]^t$ is a 2D translation vector [Hartley and Zisserman, 2000].

The view set is created by generating the views corresponding to a random sampling of the space of the $(\theta, \phi, \lambda_1, \lambda_2, t_u, t_v)$ parameters. Figure 2 shows such views. As discussed below, we use non null values of $\mathbf{t}$ to handle possible localization error of the keypoints.

3.2.2 Robustness To Image Noise

The synthesized views should be as close as possible to actual images captured from a camera, and noise from camera acquisition should also be simulated. First the noise in the original image is reduced using a median filter, then white noise is added to the generated views.

3.2.3 Robustness To Localization Error

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 [Mikolajczyk and Schmid, 2002]. The keypoint descriptor in [Lowe, 1999] handles this problem by carefully assuring that a gradient vector contributes to the same local histogram even in case of small positional shifts.

In our case, we simply allow the translation vector $\mathbf{t}$ of the affine transformation of Equation 1 to vary in the range of few pixels when generating the view sets. These small shift corresponds to the noise that arises in corner detection.

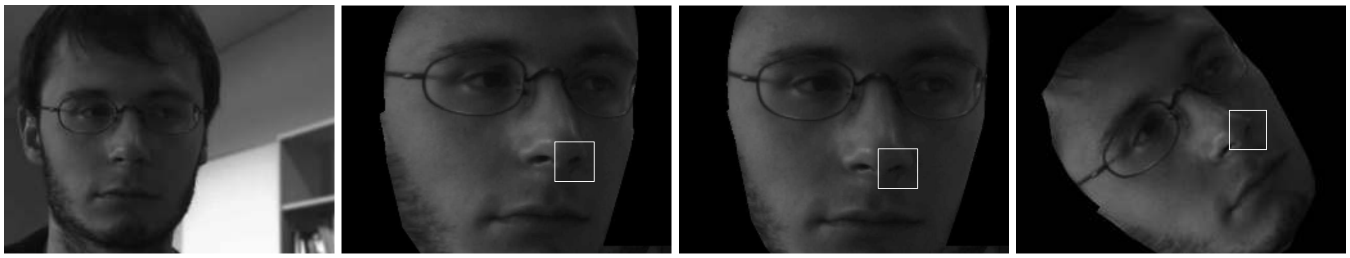

Figure 3: Three synthetic views of a human face, generated from the original image on the left. The patches extracted from these images to build a viewset of a keypoint on the nose are represented.

3.2.4 Invariance To Illumination Changes

Even if handling illumination changes can be done by normalizing the views intensities like in our previous work, we will show that using classification trees allows to skip this step. The classification indeed relies on tests of type "Is this pixel brighter than this one ?". This avoids the use of an arbitrary normalization method, and makes the classification very robust to illumination changes.

3.2.5 Semi Invariance To 2D Rotation

The classification task can be simplified if one can preprocess the data in a way that removes some variations within the classes. It has been shown that normalization can be achieved by attributing a 2D orientation to the keypoints: [Allezard *et al.*, 2000, Mikolajczyk and Schmid, 2002] take the image gradient at the keypoint position, [Lowe, 2004] attributes the orientation from the histogram of gradient directions in a patch centered on the keypoint. We describe our own method in Section 5. Note that we do not require a particularly stable method, since the same method is used for the training and at run-time recognition. We just want it to be reliable enough to remove some variation within the same point views.

3.2.6 Relaxing the Planarity Assumptions

One advantage of our approach is that we can exploit the knowledge of a 3D model if available. It does not have to be especially accurate as the generic face model we use to illustrate our technique. Such a model is very useful to capture complex appearance changes due to changes in pose of a non convex 3D object, including occlusions and non-affine warping. For example, as shown in Figure 3, we generate several views of a keypoint on the left side of the nose by texture mapping the face in several positions.

This approach also lets us merge the information from several training images in a natural way: The generated viewsets can simply be combined when they correspond to the same 3D point as depicted in Figure 4.

4 Using Classification Trees

In previous work [Lepetit *et al.*, 2004], we used a K-mean plus Nearest Neighbour classifier to validate our approach, because it is simple to implement and it gave good results. Nevertheless, it is known to be one of the less efficient classification methods. Even though they are more complex, classification trees are better suited to our problem, because they allow very fast recognition and they naturally handle multi-class problems. We will

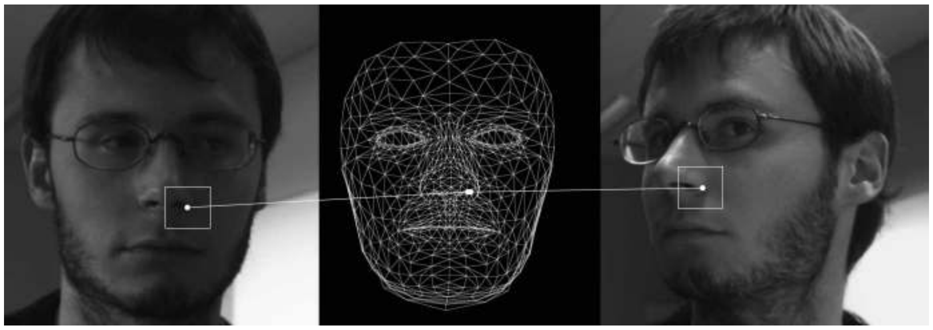
Figure 4: Using two different training images to build the viewset of the same keypoint.

show that their recognition rate is higher than our previous classifier, and that they provide a probability distribution for the match that is reliable and useful in the robust pose estimation stage. As could be expected, that leads to the performance improvements discussed in Section 6.

4.1 Short Description

Classification trees are powerful and popular tools, and have been used to solve many pattern recognition problems [Amit and Geman, 1997, Fleuret and Geman, 2001, Viola and Jones, 2001] — the cascade in [Viola and Jones, 2001] being a degenerated form of tree. As depicted by Figure 5, each non-terminal node of the 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 of the conditional distribution over the classes given that an image reaches that leaf. These estimates are simply relative frequencies based on training data. 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. A good criterion to evaluate the separation efficiency is the expected gain in information. 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 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 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.

4.2 Node Tests

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

$$\begin{array}{llrcll} \text{Test}(\mathbf{dm_1}, \mathbf{dm_2}) \equiv & \text{If} & \tilde{I}(\mathbf{m}+\mathbf{dm_1}) - \tilde{I}(\mathbf{m}+\mathbf{dm_2}) & < & -\tau & \text{go to node child 1;} \\ & \text{if} & |\tilde{I}(\mathbf{m}+\mathbf{dm_1}) - \tilde{I}(\mathbf{m}+\mathbf{dm_2})| & \leq & +\tau & \text{go to node child 2;} \\ & \text{if} & \tilde{I}(\mathbf{m}+\mathbf{dm_1}) - \tilde{I}(\mathbf{m}+\mathbf{dm_2}) & > & +\tau & \text{go to node child 3.} \end{array}$$

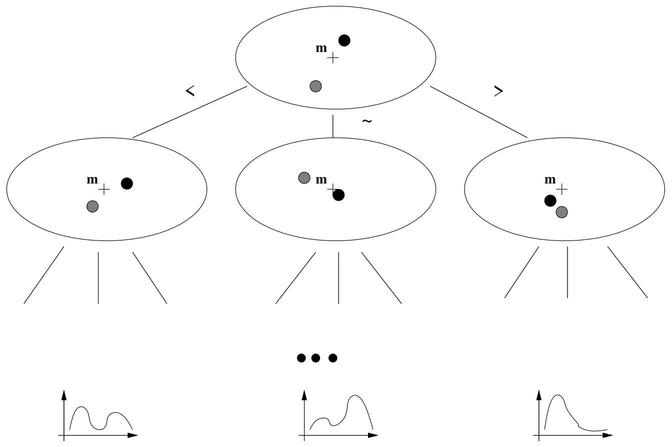

Figure 5: The type of tree used for keypoint recognition. The nodes contain tests comparing two pixels in the keypoint neighborhood, the leaves contain the posterior distributions.

where $\tilde{I}$ is the image smoothed with a Gaussian filter to reduce sensibility to noise, $\mathbf{m}$ is the location of the keypoint, $\mathbf{dm}_1$ and $\mathbf{dm}_2$ two displacements in the neighborood of the keypoint, and τ a threshold deciding in which range two intensities should be considered as similar. $\mathbf{dm}_1$ and $\mathbf{dm}_2$ are chosen to optimize Criterion 2 given above. We use a fix value for τ (in the following experiments, we took $\tau = 10$).

Note that other kinds of tests could be used, and that different kinds of tests could coexist in the same tree in a natural way, since the criterion used to choose the tests does not depend of their nature.

4.3 Randomized Trees

When the number of classes, of training examples and of possible tests are large as in our case, building the optimal tree becomes quickly untractable. Instead we grow multiple, randomized trees as in [Amit and Geman, 1997]: for each tree, we retain a small random subset of training examples, and we entertain only a small random sample of tests at each node.

4.4 Run Time Keypoint Recognition

Once the randomized trees $T_1, \ldots, T_K$ are build, the posterior distributions $P(Y = c | T = T_k, \text{ reached terminal node } = \eta)$ can be estimated for each terminal node η from the training set. If we denote $d_k(\mathbf{m})$ the posterior distribution in the node of tree T_k reached by the keypoint $\mathbf{m}$, the keypoint $\mathbf{m}$ is classified considering the average of the distributions $d_k(\mathbf{m})$:

$$\hat{Y}(\mathbf{m}) = \operatorname*{argmax}_c \, d(\mathbf{m}) = \operatorname*{argmax}_c \frac{1}{K} \sum_{k=1 \ldots K} d_k(\mathbf{m})$$

Keypoints with a too low posterior probability $d(\mathbf{m})$ are rejected as keypoints detected on the background, or misclassified keypoints.

5 Keypoint Detection

Numerous good methods to detect interesting points in an image have already been proposed, and we could use one of them in our approach. Nevertheless, we developed our own method that we believed to be interesting to present here, for several reasons:

- it is close to the method proposed for keypoint classification;
- it is fast;
- it reveals to be very stable in practice;
- it can attribute an orientation to the keypoint.

The stability for both keypoint detection and orientation attribution can be visualized Figure 7. Some other approaches certainly have greater stability, but ours has a very low complexity, and is a good alternative when computation time is an issue since it runs easily in real-time on a standard PC.

Since keypoint detection can be seen as a two-class point classification problem ("Is this pixel a keypoint or not ?"), we can define a very similar method than the one presented above for keypoint recognition. For speed reasons, we developed an "engineered classifier", but it is fundamentally of the same nature than the trees described in Section 4.

As described in Figure 6.a, the basic idea of our method is to consider the intensities along a circle centered on each candidate keypoint. If two diametrically opposed pixels on this circle have approximately the same intensity that the candidate keypoint at the center, we decide that this point is *not* a keypoint. Indeed, for points in an uniform area or along an edge, one can always find such a pair of diametrically opposed pixels. Therefore we scan the circle doing tests of type:

$$\begin{array}{lrcll} \text{If} & |\tilde{I}(\mathbf{m}) - \tilde{I}(\mathbf{m} + \mathbf{dR}_\alpha)| & \leq & +\tau & \text{and} \\ \text{if} & |\tilde{I}(\mathbf{m}) - \tilde{I}(\mathbf{m} - \mathbf{dR}_\alpha)| & \leq & +\tau & \text{then } \mathbf{m} \text{ is not a keypoint,} \end{array}$$

where $\mathbf{dR}_\alpha = (R\cos\alpha; R\sin\alpha)$, R being a chosen radius and α varying in the range $[0; \pi]$. Because we deal with discretized images, in practice we have to compare not only the diametrically opposed pixels but also their neighbors, to avoid responses near edges, as shown in Figure 6.b. One easily sees that we can construct a tree similar to the one in Section 4 that would make these tests along the circle. Usually, non keypoints are rejected very quickly, without scanning the whole circle.

This classifier usually produce several positive responses around keypoint locations. Like in other point detectors, we compute a score computed for each response, and retain the local optimum locations as keypoint. The Laplacian of Gaussian is a commonly used score that has been proven to produce stable keypoints. It can be approximated up to a scale factor by:

$$\text{LoG}(\mathbf{m}) \approx \sum_{\alpha \in [0;\pi[} \tilde{I}(\mathbf{m} - \mathbf{dR}_\alpha) - \tilde{I}(\mathbf{m}) + \tilde{I}(\mathbf{m} + \mathbf{dR}_\alpha)$$

Of course, this expression can be computed while scanning the circle, and only for a very limited number of pixel locations, that makes it efficient in our context. We experimentally verified the quality of this approximation: for example, for a radius $R = 7$ and an smoothed image $\tilde{I}$ obtained with a Gaussian filter mask of size 7×7 and $\sigma_G = 1$, we obtained an approximation of the Laplacian of Gaussian with $\sigma_{Lap} = 3.5$, with an error lower than 10%.

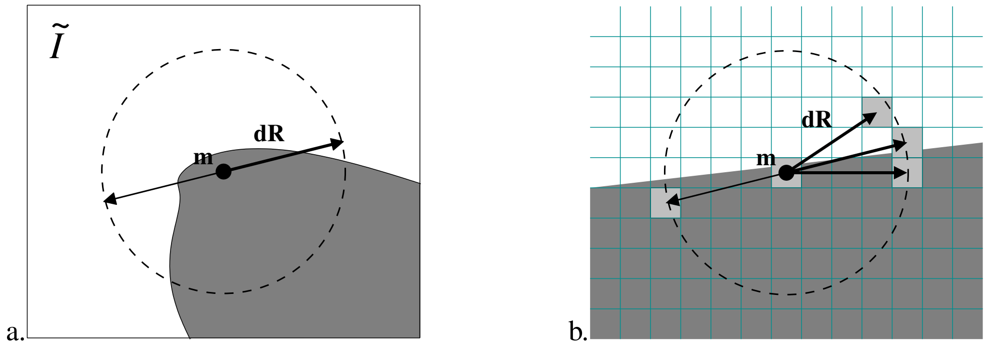

Figure 6: a. Keypoint detection principle: if $\tilde{I}(\mathbf{m})$, $\tilde{I}(\mathbf{m} + \mathbf{dR})$, and $\tilde{I}(\mathbf{m} - \mathbf{dR})$ are similar, $\mathbf{m}$ is not a keypoint. b. Because we deal with discretized images, we have to compare not only the diametrically opposed pixels but also their neighbors, to avoid responses near edges.

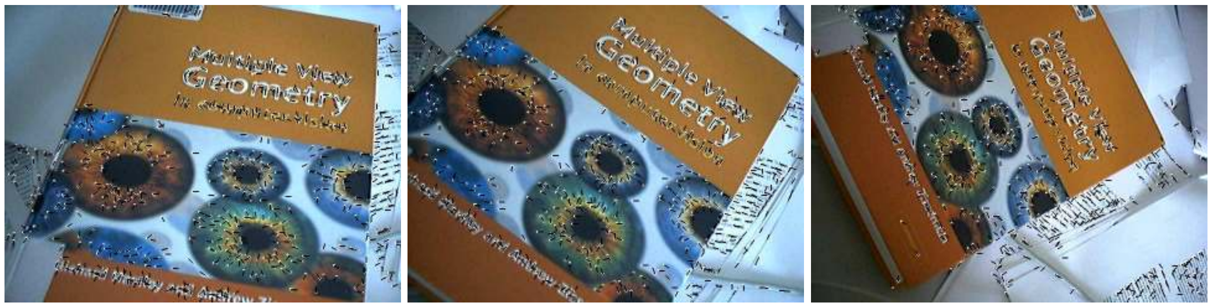

Figure 7: Some results of keypoint detection and orientation attribution with our algorithm.

We use the same framework to attribute an orientation to the keypoints. We take the orientation $\alpha_{\mathbf{m}}$ such as:

$$\alpha_{\mathbf{m}} = \underset{\alpha \in [0;2\pi]}{\text{argmax}} \; |\tilde{I}(\mathbf{m}) - \tilde{I}(\mathbf{m} + \mathbf{dR}_\alpha)|.$$

This orientation is stable enough to normalise the neighborhood of the keypoint with respect to the 2D rotation. One more time, it is very efficient to determine while scanning the circle.

6 Experiments

To test the proposed method, we performed the following experiments: we detected keypoints in the image of the book cover shown Figure 8 using our method described in Section 5, and retained the 200 strongest keypoints. Several trees were grown as follow:

- 100 new labelled random views are generated per keypoint;
- for each node, we try n tests and keep the best one according to Criterion (2);
- for the root node, we use $n = 10$, a very small number to reduce correlation between trees;
- for the other nodes, we use $n = 100d$, where d is the depth of the node;
- the tree depth is limited to a given maximal depth.

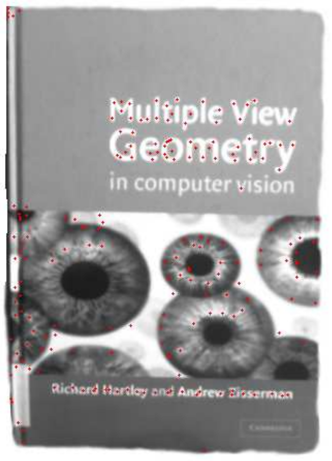

Figure 8: The original image of the book cover and the retained 200 strongest keypoints.

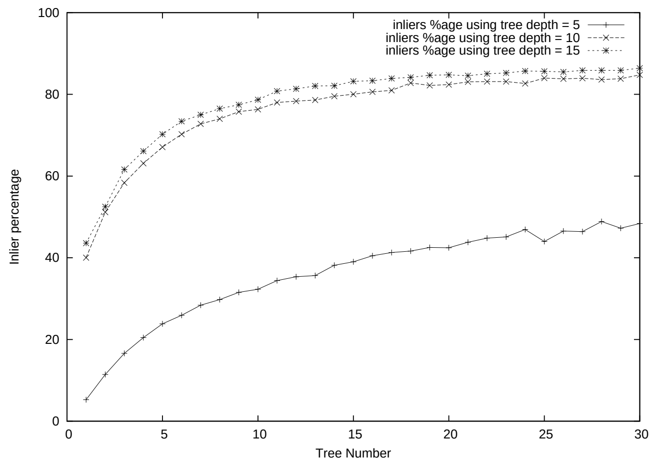

Figure 9: Percentage of correctly classified views with respect to the number of trees, using trees of depth 5, 10, and 15.

- the posterior probabilities are estimated from 1000 new random views per keypoint.

All the views are 64×64 patches, generated from random affine deformations with $(\lambda_1, \lambda_2) \in [0.5; 1.5]^2$, $\theta \in [0; 2\pi]$, $\phi \in [0; \pi]$, $(t_u, t_v) = [-2; +2]^2$, and preprocessed as described in Section 3. The classifier made of these trees can then be tested using (once more) 1000 new random views. The graph of Figure 9 represents the percentage of views correctly classified, with respect to the number of trees, for several maximal depth 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 take about 15 minutes.

The advantage of local approaches is that the object can be detected provided than enough points on the object are recognized, so this classification rate is largely sufficient to detect the object. At run-time, the classifier is used to recognize the detected keypoints. We then estimate the global affine transformation between the training image and the input image, robustly using RANSAC. Since the outlier rate is low, RANSAC usually finds a satisfying transformation after a few trials. This affine transformation is then refined with an homography. The whole process (keypoint detection, recognition and pose estimation) runs at 25 Hz on a standard PC. Figure 1 shows some examples of detections. An executable demo can downloaded at `http://cvlab.epfl.ch/research/augm/detect.html`, and tested by the interested reader on his/her own exemplar of the book.

7 Conclusion and Perspectives

We proposed an approach to keypoint matching for object pose estimation based on classification. We showed that using decision trees for the classification yields to a powerful matching method well adapted to object recognition. Moreover we also presented a method for keypoint detection that relies on a very similar methodology.

The very next development is the application to real 3D objects recognition. Our approach is very general, and lets us relax the locally planar assumption. In fact, it has the potential to recognize complex shaped textured objects, under large viewpoint and illumination changes even with specular materials, assuming we can generate images of the object under such changes. This is a realistic assumption since there are many Computer Graphics methods designed for this purpose, which opens new avenues of research.

We also plan to investigate other kinds of features considered in the classification tree nodes. In this paper, we restricted the tests to comparison between intensities of two pixels, but it would be interesting to extend the tests to test gradient values, or use Haar wavelets like in [Viola and Jones, 2001]. We already get good recognition rates for a well-textured object, and also considering such features would extend our approach to a larger class of objects. Thus, we believe that our approach is an important step toward much better object recognition and detection methods, and opens good possibilities for future research.

Acknowledgments

The keypoint detection was developed in collaboration with Tom Drummond from University of Cambridge. The authors would also like to thank Francois Fleuret from EPFL for pointing out classification trees for the classification method.

References

[Allezard *et al.*, 2000] 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*, pages 960–963, Barcelona, Spain, Sep 2000.

[Amit and Geman, 1997] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

[Baumberg, 2000] Adam Baumberg. Reliable feature matching across widely separated views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[Fleuret and Geman, 2001] Franois Fleuret and Donald Geman. Coarse-to-fi ne visual selection. *International Journal of Computer Vision*, 41(1):85–107, January 2001.

[Harris and Stephens, 1988] C.G. Harris and M.J. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference, Manchester*, 1988.

[Hartley and Zisserman, 2000] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[Jurie, 1999] F. Jurie. Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model. *Computer Vision and Image Understanding*, 73(3):357–373, 1999.

[Lepetit *et al.*, 2004] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classifi cation Problem for Fast and Robust Object Pose Estimation. In *CVPR*, Washington, DC, June 2004.

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

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

[Mikolajczyk and Schmid, 2002] K. Mikolajczyk and C. Schmid. An affi ne invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002. Copenhagen.

[Mindru *et al.*, 1999] 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.

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

[Rothganger *et al.*, 2003] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using affi ne-invariant patches and multi-view spatial constraints. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, jun 2003.

[Schaffalitzky and Zisserman, 2002] 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.

[Schmid and Mohr, 1997] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.

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

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