Real-Time Feature Matching using Adaptive and Spatially Distributed Classification Trees

Aurélien Boffy *
CERTIS Laboratory
École des Ponts et Chaussées
Paris, France

Yanghai Tsin
Real-Time Vision and Modeling Dpt
Siemens Corporate Research
Princeton, USA

Yakup Genc

Abstract

This paper presents a method for real-time wide-baseline feature matching. The approach is based on the work of Lepetit and colleagues [9], where randomized decision trees are trained to establish correspondences between detected features in a training image, and those in input frames. Though extremely promising, their actual results can vary depending on the viewpoint and illumination conditions. We combine two approaches to alleviate its limitations. The first aims to update the trees at run-time, adapting them to the actual viewing conditions. The second consists in spatially distributing the trees, so that each of them models a certain viewing volume more precisely. The result is a more stable matching method that significantly extends detectable range and is much more robust to illumination changes, such as cast shadows or reflections.

1 Introduction

We address the real-time feature matching problem in this paper. We are concerned with matching features observed from two views separated by a wide baseline. This problem remains one of the most fundamental issues in computer vision research. A solution to it has many applications, such as object detection, recognition, tracking and 3D reconstruction. The problem is difficult because unlike short baseline feature tracking methods [12], a crucial source of information, spatial-temporal adjacency of feature points, is absent. Furthermore, a feature appearance can be distorted by large geometric transformations or significant illumination changes.

The community has observed some exciting breakthroughs in the past several years. For example, SIFT feature detector [11] and affine invariant feature detectors [2, 13] have seen maturity in real applications. Recently, significant progress was made by Lepetit et. al. [9]. They treat the issue of feature matching as a classification problem by using randomized classification trees [1]. Their system shows real-time (25 frames per second) feature matching very robustly.

This paper aims to alleviate some of the limitations of their method in order to make it even more widely applicable. As noted in their paper, an object can be reliably detected only within a certain range of viewpoints and illumination conditions, which are

*The work was done while Aurélien Boffy was working for Siemens.

determined by the training images that are rendered from a model. Once trained, the performance can no longer be improved. In addition, during the image rendering process, any artifacts introduced, such as unrealistic occlusion and shadow/shading, may have an impact and cannot be corrected. Another limiting factor of their method is the trade-off between feature matching accuracy and the volume covered by the trees, when each trained tree attempts to cover all viewpoints.

In observation of these problems, we propose several techniques to significantly improve the performance of the classification trees at run-time. Our insights are summarized as follows:

- Live feedback from an input video, such as true matches (inliers) or false matches (outliers), can be used to improve the trees. This effectively removes artifacts introduced in the training stage and can significantly increase the detectable range. Note that detection of inliers/outliers can be achieved independently by geometric methods, and will be discussed later on.

- Trees can be spatially distributed, meaning that each of them only covers a predefined viewing volume. For tracking, temporal coherence of poses can help adapt a specific subset of trees, while locally weighted sum of matching probabilities can outperform matching accuracy of the original method.

We would like to point out that our method is suitable for both object detection and tracking. When there is no information regarding the current camera position, all trained trees are used to match features. Once the object is detected, the camera pose estimate is used to select the subset of trees, which will be used for the next frame. As a result, the computational cost can be reduced during tracking. Smooth handovers are achieved by allowing overlap among viewing volumes of different trees.

The remainder of this paper is structured as follows. In section 2, we review previous work on the topic and we describe more precisely the work of Lepetit et al. Section 3 focuses on the improvement of the classification trees during the real-time stage, while Section 4 explains the spatially distributed trees. Section 5 presents results of object detection and tracking in comparison with the work of Lepetit et al., and Section 6 presents conclusions.

2 Related Work

2.1 Previous Work on Feature Matching

Short baseline feature tracking [12] was among the first successful methods in computer vision. Attempts have been made to match features separated by a wide baseline. Among the most successful ones are the SIFT feature detector [11] and affine invariant feature detectors [2, 13]. Authors for both the SIFT feature detector and some affine invariant feature detectors found their inspirations from earlier work in scale space analysis [10] and studies of biological visual systems, especially the center-surround structures of mammalian ganglion cells [8]. Although powerful, these feature detectors are usually computationally costly on sequential computers, which makes them difficult to be applied in resource-demanding applications.

2.2 The Original Method of Lepetit et. al.

Lepetit et al. [9] proposed a wide-baseline feature matching method that has real-time performance. They formulate matching as a classification problem. Each feature of the object selected during the training phase is considered as a class corresponding to the set of all its possible appearances. The goal of feature matching is achieved by attributing each newly detected feature in a test image to one of these classes. As the features are classified by binary or ternary decision trees and decisions are made by very simple tests at each node, the feature matching process is extremely fast.

Given either a single image of an object or a fully textured 3D model, a set of training images are synthesized by warping the reference model to a set of viewpoints defined by random samples of transformation parameters. Since the training images are synthesized, the true correspondences are known. Consequently, the detectability of each feature in the synthesized images can be studied. The most stable features of the object, i.e., the ones that can be detected repeatably despite noise, are selected.

A random set of labeled features are used to build the decision trees. At each internal node, a set of tests involving intensity comparison between two pixels are randomly drawn. The test that results in maximum information gain is chosen as the splitting criterion for the node. The process is repeated until the number of training examples left is small, or when a given depth is reached. At each leaf node, the sample frequency of each feature class is stored. This is an estimate of the conditional distribution over the classes given that a feature reaches that leaf. To reduce variances of the conditional distribution estimates, multiple randomized trees are trained independently. During the testing phase, a test feature is dropped down each tree independently. The average distribution amongst those stored in all reached leaf nodes is used to classify the input feature, utilizing maximum a posteriori estimation.

3 Adaptive Classification Trees

3.1 The Adaptive Method

The principle is to extend the training phase by analyzing matching performance on live input videos. We propose to update the trees with the frames actually captured by the camera. There are two types of live updates.

First, for each frame, after the feature matching step, independent geometric methods, such as the 3-point pose estimation algorithm [6] (with known 3D coordinates of the feature points selected during the training phase) or the fundamental matrix estimation algorithm [7] (with unknown 3D), are coupled with RANSAC [3] to determine if each declared correspondence is a good match (inlier) or a mistake (outlier). If the projection (projected point or epipolar line) of a training feature is beyond a given distance from the declared "correspondence", an outlier is detected. Otherwise, an inlier is found. Inlier matches are used to reinforce the distributions on the reached leaf nodes by increasing the probabilities of the feature classes corresponding to them (see Figure 1 for an illustration).

Second, in the case of known 3D coordinates of the selected training features, we can improve the trees further by recovering the lost matches. Once we know the pose of the camera, we can project onto the frame the training features which were not correctly matched with any detected feature. If the distance between the projection of a training

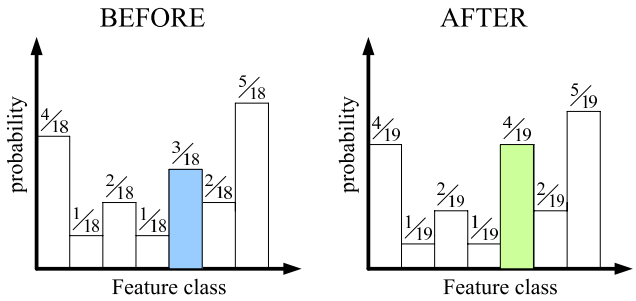


Figure 1: *Update of the probability distribution function of a leaf.* The conditional probability is kept by sample frequencies, i.e., K the number of features that reach the leaf, and K_i the number of samples belonging to class i, resulting in a probability estimate of K_i/K. During the on-line training, we just continue in the same way to increase the K and K_i values.

feature p and the nearest detected feature p' is smaller than a given threshold, it means that the classification of this feature p' has failed. We then identify the leaf nodes that p' reached, and increase the probabilities corresponding to the feature class of p, so that it becomes more likely that the classification of this feature will succeed in the future. A similar approach can be applied to the fundamental matrix based method.

3.2 How Does the Proposed Method Work?

In the following, we explain how the proposed method circumvents some difficulties of the original work.

Geometric modeling imperfections can be compensated for by updating the trees with the images actually captured by the camera, instead of only using synthesized views. For local 3D structures, e.g., 3D corners, any 2D transformation is insufficient to capture their appearance changes. If a set of examples with incorrect appearances are synthesized and used to train the trees, the performance will be poor at the testing phase. However, even if such a feature p is matched with a wrong feature, as long as RANSAC is robust enough to estimate the camera pose despite these outlier matches (often the case by definition of RANSAC), the true match p' can be identified and the corresponding probability table can be updated. If enough observations of this type are made, p' will eventually be matched with p despite the initial synthetic errors.

This brings up another advantage of the proposed method. Even for complex 3D structures, our method allows us to use only a single image plus a sparse set of 3D points corresponding to the selected features, as inputs. 2D transformations (homography or affine) are used to synthesize new views. Imperfections in these planar transformations can be amended by live adaptive training. We successfully conducted our experiments following this approach. As a result, we avoid the requirement for dense full models of 3D objects. Furthermore, the requirement for 3D coordinates of the selected features can also be relaxed if 1) the planar transformations result in a sufficient number of correct matches from which the 3D coordinates of most features can be estimated using structure from motion methods [7] or 2) epipolar geometry alone is sufficiently effective for discriminating inliers/outliers.

Similarly, errors introduced by other modeling imperfections, such as appearance modeling inaccuracy (limited resolution of the input model, shadow, shading or reflection), can be compensated for in the same way. Given enough adaptation, the trees will eventually model more precisely the actual appearances of an object under different geometric and photometric conditions. As a result, we are able to extend the working ranges of viewpoints and illumination conditions tremendously.

3.3 Discussion

A necessary condition under which the adaptive method improves performance is that pose estimation is successful for the frame under consideration. This can be achieved by starting the live training only when the object is detected with confidence, e.g., with enough inlier matches.

Live update can also fail on rare occasions. For example, if an un-modeled feature q blocks the true feature p and the camera stays in the same place for a while, feature q will tend to be considered as the "correspondence" of p. However, if the camera is continuously moving, such an erroneous update usually does not happen repeatedly for the same class in the same leaf node. Furthermore, if the true feature is exposed later, the tree can adapt back to the real correspondence. As a result, a second requirement for the method to work is that such accidental views happen rarely. This is always the case in our study. In fact, research has been done regarding the generic view assumption and the rareness of accidental views [4].

Another concern with the proposed method is overfitting. If we update the trees while the camera stays in the same region for a long time, the trees could be excessively adapted to this position and the performance could drop if the camera quickly moves to another place. However, if we update only a local set of trees instead of all the trees, as we discuss in the next section, this problem can be avoided.

Notice that we only update the leaf nodes of the trees while leaving the internal nodes and the structure of the trees intact. It has been observed by Lepetit et al. [9] that they could even use, with good performance, the same tree structure for a different object, by only updating the leaf nodes. This implies that the leaf nodes are the most discriminative part in separating one feature from another. Our experiments confirm their observation.

4 Spatially Distributed Trees

4.1 Overview

It has been observed that the appearance of objects stays approximately the same in a certain viewing volume, and changes suddenly at some singular boundaries. This has been the subject of aspect graph research [5]. Similarly, the appearance of a selected feature on an object stays similar in a local viewing volume, but it may be drastically different from some distant locations. For each tree, the original work in [9] attempts to capture global feature appearances. Inevitably, there is a trade-off between feature matching performance and the volume covered by the trees.

We propose to specialize the trees, so that each of them models a specific viewing volume more accurately. These regions have overlaps, so that we still use multiple trees. As the within-class variation in each tree is lower, we can use fewer trees that are smaller

to achieve the same feature matching accuracy. Thus, the computational time can be reduced. This is especially true in the "tracking" phase, where the subset of trees can be identified by the position of the camera in the previous frame. For the "detection" task, where the camera position is not known, we propose to use all the trees and the computational time depends on the total number of trees.

Used in combination with adaptive trees, this approach allows us to only update a local set of trees. Thus, we avoid overfitting the trees in a global scale. They are biased toward classifying a local region more precisely, but that is desirable in this setting.

Another advantage of this method concerns the stable feature selection process. In the initial method of Lepetit et al., the selected features are those that are globally stable, which may be rare and volatile. Spatially distributed trees enable us to use different features according to the position of the camera. As a matter of fact, some features can be very stable in a limited region but not detectable elsewhere. They could be adopted for the local trees, but would certainly be ignored in the initial algorithm proposed by Lepetit et al.

4.2 Implementation Details

A regular grid of positions (uniform in angular and logarithmic radius directions) is defined surrounding the object of interest (see Figure 2). A position, combined with a range (radius), defines a spherical viewing volume. Neighboring spheres have a large amount of overlap. Each position is assigned to a classification tree, which is designed to detect and track the object if the camera position is located inside its spherical coverage region.

Training images of each tree are rendered by randomly positioning the camera inside the tree's coverage region. Note that to be able to synthesize an image from a specific point of view, we assume that a textured 3D model of the object is available. Thus, while a single image of the object is required to use adaptive trees (plus a sparse set of 3D points if needed), a textured 3D model is necessary to take full advantage of spatially located trees.

At run-time, we select the trees that cover the camera position in the previous frame. They are used for feature matching and are updated, if necessary [*]. To ensure smooth transitions, we use weighted average of conditional probability tables to classify each input feature, where the weight is a monotonically decreasing function of the distance from the camera position to the center of each sphere.

If the object is not detected in the previous frame, all the trees are used for feature matching. Empirically, we observe that the object detection performance does not suffer compared to the original method [9], but it is a bit slower due to involvement of more features and more trees. Another approach would be to keep some trees defined without specific location like in the method of Lepetit et. al. and use them in such cases. Note that we select a local set of stable features for each spatially distributed tree. If the number of features is on average N, and if we use T trees, the total number of features is between N and $T \cdot N$ (empirically it is about $3N$ due to shared stable features).

[*] We assume that camera movement between frames is small compared to the coverage region of each tree. Consequently, the two camera centers corresponding to the two consecutive frames are covered by the same set of trees.

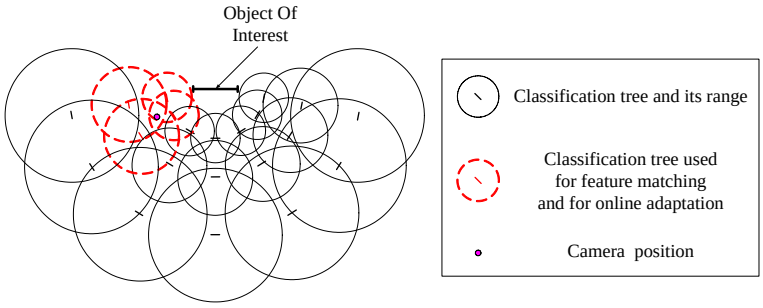


Figure 2: *Spatial distribution of the classification trees and their ranges.* The figure is for illustration purpose only. The actual distribution involves 3D spheres and larger overlaps.

5 Experimental Results

Improvements of our method over the original method are demonstrated in this section. The videos are acquired live using a low-end web cam (Unibrain Fire-I). Initial trees are built from a model of the object, i.e., a single image of an object plus a sparse set of 3D points corresponding to the selected features for the adaptive tree technique, or a textured 3D model for the spatially distributed trees. Note again that the requirements for the sparse set of 3D points can be relaxed if a robust structure from motion method [14] is integrated.

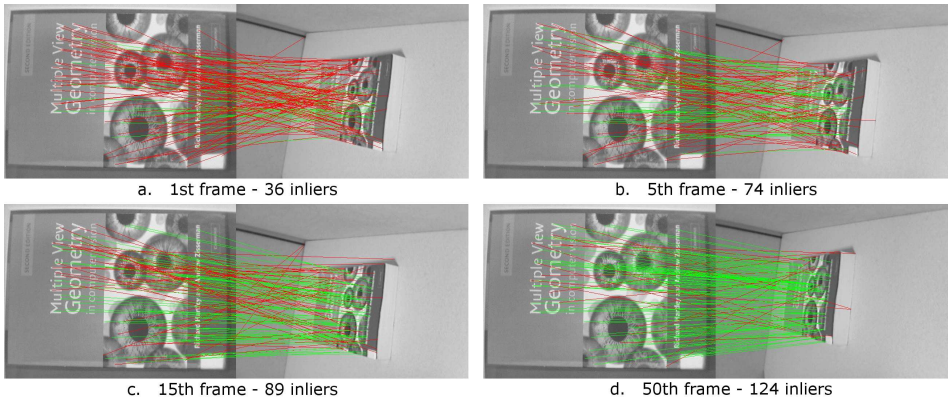
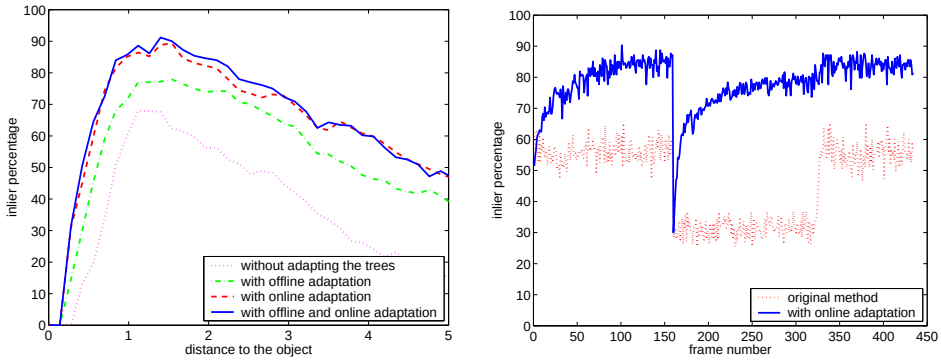


Figure 3: *Evolution of the adaptive trees.* The inlier matches are drawn using green lines, while the outliers are in red. Note that the number of inlier matches is always increasing.

We first demonstrate the power of the adaptive method without using the spatial distribution technique. After initial construction of the trees, we can adapt them both offline using video sequences acquired under different conditions, and online using live video.

Figure 3 shows the effectiveness of the adaptive method on a video sequence, where a book cover can hardly be detected by the original method (frame 1) from an oblique viewing angle, but is eventually detected with a significant increase in the number of matched features. Note that during the test, the camera is moving instead of being stationary.



(a) (b)

Figure 4: (a) *Influence of adaptive trees*. We used a single video sequence for online adaptive training. (b) *Adaptation to sudden illumination changes.* Fluctuation of the percentage of inliers, when a sudden illumination change happens at frames 155 and 330.

The following experiments are conducted with video sequences of a toy car against a cluttered background. Figure 4(a) shows the percentage of inlier matches as a function of the distance between the camera and the object, assuming this distance is 1 for the training image. The performances are significantly improved when actual images are used to adapt the trees. In this experiment, we used a sequence of 350 frames for offline adaptation, and a sequence of 440 frames for performance testing and online adaptation. A forest of 20 trees is used and 200 features are selected. Notice that performance improvement is significant, even though we only use a short sequence for offline adaptive training and turn off online adaptation. This clearly shows the strength of the adaptive method.

Figure 4(b) shows performance fluctuation with strong illumination changes. For the experiment, we switch on and off a strong light to cause saturation and reflection in the input frames. Both the original and the adaptive methods are noticeably affected when the light is turned on. However, by updating the trees with feedbacks from the actual images, they adapt themselves quickly. Furthermore, when the light is suddenly turned off, the matching performance does not suffer, meaning that the trees are not overfitted for a particular light. Without adaptation, the matching performance remains very poor while the light is on. We observed similar performance for other types of illumination changes, such as cast shadow or shading. Figure 5 gives a specific example in this test. We can find many correspondences, even when the lighting conditions are drastically different from the one used for the training image. The object cannot be detected at all in this image without an adaptation procedure starting from a detectable frame.

Next, we show the performance improvement due to the spatially distributed trees. For comparison purposes, we use the method of Lepetit et al. [9] to construct trees with different scale ranges, i.e., distance $[0.3, 2.5]$ and $[2.0, 5.0]$. For each range, a forest of 20 trees utilizing 200 stable features is constructed. It is not surprising to see that the inlier percentage is high within the nominal range, while it drops very quickly outside (Figure 6). When the trees are stretched over the whole range $[0.3, 5.0]$, the performance suffers globally.

Spatially distributed trees clearly alleviate this trade-off and extend the detectable

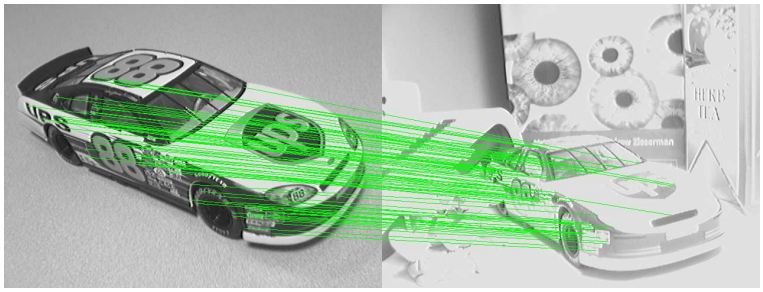


Figure 5: *Feature matching under extreme illumination conditions.* About 40% of the 200 selected features are well matched despite strong saturation and reflection.

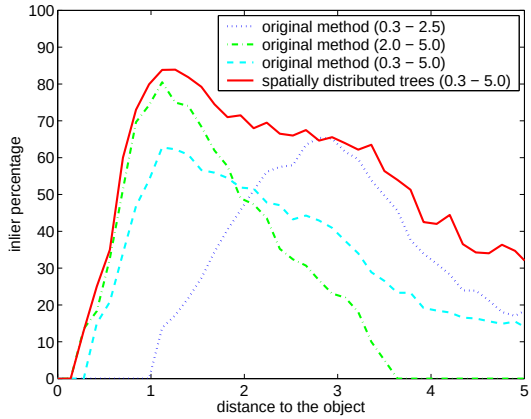


Figure 6: *Performance comparison: spatially distributed trees.* Spatially distributed trees increase both matching accuracy and coverage.

range without any drop of performance. 40 spatially distributed trees and a total of 250 features are used. However, only between 15 and 20 trees (out of 40) are used to estimate the pose for each frame. We chose not to adapt the trees for these experiments.

The inlier percentage is computed by only taking into account the set of features contained in the tree closest to the camera position. Note that in robust methods such as RANSAC, as long as the number of correspondences is sufficient, the percentage of inliers is far more important than the total number of inliers. In our case, we are able to find a subset of features (usually 80 features) containing a large percentage of inliers, which is better for pose estimation than using all 200 features in the original method but with a small percentage of correct correspondences. In this sense, we consider it a fair comparison. Also notice that we use a local cluster of trees instead of the closest tree, for less variance in the probability tables computation, and smooth transitions among coverage regions.

6 Conclusion and Perspectives

We proposed two improvements to the original randomized tree feature matching method.

By analyzing the feedback from the input video, we acquire additional knowledge about the appearance of the object under the actual viewing conditions. This is used to update the classification trees at run-time. Thus, we are able to handle more complex 3D objects, shadows, specular materials and extreme viewing angles.

Using spatially distributed trees designed to detect and track the object in predefined viewing volumes makes the method more robust to viewpoint changes, and makes feature tracking faster.

References

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

[2] V. Ferrari, T. Tuytelaars, and L. Van Gool. Wide baseline multiple view correspondence. In *CVPR*, pages 718–725, 2003.

[3] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[4] W.T. Freeman. Exploiting the generic viewpoint assumption. *IJCV*, 20(3), 1996.

[5] Z. Gigus, J. Canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE TPAMI*, 13(6):542 – 551, 1991.

[6] R. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *CVPR*, 1991.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[8] E.C. Hildreth. The detection of intensity changes by computer and biological vision systems. *Computer Vision, Graphics and Image Processing*, 27:1–27, 1983.

[9] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, 2005.

[10] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness struct. In *ECCV*, 1994.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *IJCAI-81*, pages 674–679, Vancouver, 1981.

[13] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[14] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *ICCV'03*.