Real-Time Learning of Accurate Patch Rectification

Stefan Hinterstoisser[1], Oliver Kutter[1], Nassir Navab[1], Pascal Fua[2], Vincent Lepetit[2]

[1]Department of Computer Science, Technical University of Munich (TUM)
Boltzmannstrasse 3, 85748 Garching, Germany
[2]École Polytechnique Fédérale de Lausanne (EPFL), Computer Vision Laboratory
CH-1015 Lausanne, Switzerland
{hinterst,kutter,navab}@in.tum.de, {vincent.lepetit}@epfl.ch

Abstract

Recent work [5, 6] showed that learning-based patch rectification methods are both faster and more reliable than affine region methods. Unfortunately, their performance improvements are founded in a computationally expensive offline learning stage, which is not possible for applications such as SLAM. In this paper we propose an approach whose training stage is fast enough to be performed at run-time without the loss of accuracy or robustness. To this end, we developed a very fast method to compute the mean appearances of the feature points over sets of small variations that span the range of possible camera viewpoints. Then, by simply matching incoming feature points against these mean appearances, we get a coarse estimate of the viewpoint that is refined afterwards. Because there is no need to compute descriptors for the input image, the method is very fast at run-time. We demonstrate our approach on tracking-by-detection for SLAM, real-time object detection and pose estimation applications.

1. Introduction

The recent years have seen the development of affine region detectors [9]. Coupled with a region descriptor such as SIFT [8], they proved to be very useful for many types of applications. More recently, an approach based on learning instead of *ad hoc* detectors was developed by Hinterstoisser *et al.* [6, 5]. This approach appears to be more reliable and much faster, but relies on an extensive training stage. That makes it unqualified for applications such as relocalisation in Simultaneous Localization and Mapping (SLAM), which requires on-the-fly integration of new feature points as they become visible.

In this paper, we propose an approach that removes this limitation. It also relies on training and has the same performances at run-time than [5], but training it for new points is much faster and can be done in real-time. As Fig. 1 shows, it is very useful for SLAM applications in poorly textured environments. Given a frontal view such as the one of Fig. 1(a) of a locally planar part of the scene and the camera internal parameters, we can estimate the six degrees of the camera in the next frames (Fig. 1(b-d)) in real-time, despite the near complete absence of feature points. Since the patches are detected and their poses estimated in every frame independently, the method is very robust to fast motion and occlusion. If frontal views are not available or can not be identified, we retrieve a displacement relative to some reference frame, which can be useful for 3D detection and recognition of low-textured objects, such as the ones in Fig. 1(e,f). Another possible application illustrated by Fig. 1(g,h) is the registration of deformable surfaces.

The approach of [6, 5] is made of two steps. The first step matches an incoming feature point against a database of feature points. It retrieves the "identity" of the point, i.e. the feature point in the database it corresponds to, as well as a coarse estimation of its *pose*, defined as the homography between a reference patch and the patch centered on the point. The second step refines the pose using the inverse compositional algorithm [1] and linear regressors [7]. For the first step, [6] uses the Ferns classifier [11] while [5] relies on linear classifiers. Both methods require a significant amount of time for training.

We keep here this two steps approach, but our main contribution makes the training of the first step much faster. We use an approach closely related to geometric blur [4] to recognize and estimate the pose of the feature points. Each point is characterized by a set of mean patches, where each mean is computed by warping the patch centered on the point over a small range of poses. We used this approach because it allows us to retrieve quickly and reliably the incoming point identities and poses, but also because we developed a method to very quickly compute the mean patches. While the geometric blur approach uses spatially

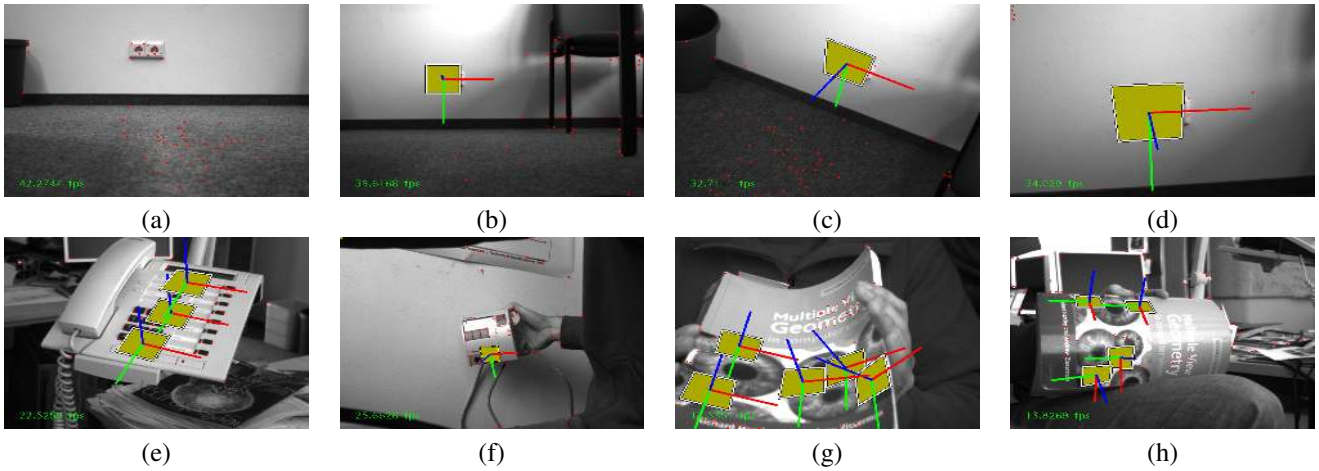

Figure 1. Real-Time Learning of Accurate Patch Rectification. Our method learns new feature points and how to estimate their perspective orientation on-the-fly. As a result, it is a very useful tool for applications such as SLAM relocalisation in poorly textured environments, and detection of of poorly textured objects or deformable objects. In the images (a)-(h) we see rectified patches with coordinate axes attached to them which reflect the retrieved poses.

variant Gaussian kernels and is too slow for our purpose, we use a method based on the linearity of the warping function and the principal components decomposition of the original patch. Computing a mean patch then only requires the computation of a linear combination of precomputed vectors. Another difference to geometric blur [4] is that we directly match the original patches of the incoming points against the mean patches stored in the database. We found it to give good results, and has the advantage of not requiring the computation of the mean patches for the incoming points. For this reason, we call the set of means for a given point a *one-way descriptor*.

Our second step is similar to the one in [6] and [5] but we show that the precomputation of the sample points for each training sample makes real-time computation of the linear regressors possible.

In the remainder of the paper we start by discussing related work before we explain the design of the one-way descriptor. We show how an appearance independent offline training stage helps the real-time composition of an appearance dependent descriptor and compare it to existing rectification methods on synthetic data. Finally, we show real world applications of the method including tracking-by-detection for SLAM applications, real-time object detection and pose estimation applications.

2. Related Work

Affine region detectors are very attractive for many applications since they allow getting rid of most of the image warpings due to perspective transformations. Many different approaches have been proposed and [9] showed that the Hessian-Affine detector of Mikolajczyk and Schmid and the MSER detector of Matas et al. are the most reliable ones. In the case of the Hessian-Affine detector, the retrieved affine transformation is based on the image second-moment matrix. It normalizes the region up to a rotation, which can then be estimated based on the dominant gradient orientation of the corrected patch. This implies using an *ad hoc* method, such as considering the peaks of the histogram of gradient orientations over the patch as in SIFT [8]. However, applying this heuristics on a warped patch tends to make the retrieved pose relatively unstable. In the case of the MSER detector, many different approaches exploiting the region shape are also possible [10], and a common approach is to compute the transformation from the region covariance matrix and solve for the remaining degree of freedom using local maximums of curvature and bitangents. After this normalization, SIFT descriptors are computed in order to match the regions.

The method proposed in [6] performs the other way around, in two steps. During the first step, the patch identity and a coarse pose are retrieved using a method close to [11]. During the second step, a linear predictor is applied to estimate a fine perspective transformation. In contrast, [5] proposes to simultaneously retrieve the pose and the identity by first assuming a feature point identity, then retrieving and refining a coarse pose estimate using linear predictors and finally confirming the identity and the computed pose if the similarity between the rectified patch and the predicted one is close enough. These approaches are robust and retrieve a very accurate pose, however in both cases the first step requires time consuming training. In this paper we follow a similar general approach as [5], however the first step we propose requires almost no time for train-

ing, making possible the integration of new feature points in real-time.

3. Proposed Approach

Given an image patch at run-time, we want to match it against a database of possible patches defined around feature points and accurately estimate its pose represented by a homography. Contrary to [5, 6], we do not want to learn the patches in a long computational expensive offline training phase, but online and in real-time without loosing the ability to accurately estimate the pose of the patches under large viewpoint changes.

We show in the following how to get very quickly a coarse estimate of the pose of assumed keypoints at run-time without relying on a heavy training stage. The refinement of the keypoint pose is performed very similarly to [5, 6] with linear predictors but we use a simple trick to speed up the predictors training. We also describe this second step for the sake of completeness.

3.1. A One-Way Descriptor

As depicted by Fig. 2, our starting idea to estimate the identity and pose of a feature point is to first build a set of mean patches from different reference views of the feature point. Then, by matching the incoming feature points against these mean patches, we get a coarse estimate of the pose with respect to the assumed identity. Computing a single mean patch over the full range of poses would result in a blurred irrelevant patch, but because we compute the means over only a small range of poses, they are meaningful and allow for reliable recognition. Another advantage that comes for free is that the means are robust to image noise and blur. Using the mean patches instead of using the simply blured versions of the original warped patches substantially improves the matching as we show in Fig. 2(c).

As we will show below, this approach has the same performance in terms of speed and accuracy as the linear classifiers of [5]. Its main advantage is the much shorter training stage it requires. However, this reduction is made possible only thanks to a technique based on the Principal Component Analysis of the patches as we describe in this section.

Compared to more standard approaches [9], we do not have to extract an estimate of the feature point pose such as an affine region, nor compute a descriptor for the incoming points, and that makes the approach faster and easier to implement. The set of means that characterizes a feature point in the database can be seen as a descriptor, which we call a "one-way descriptor" since it does not have to be computed for the new points. Our approach increases the number of vectors that have to be stored in the database, but fortunately, efficient methods exist for nearest-neighbor search in large databases of high-dimensional vectors [2].

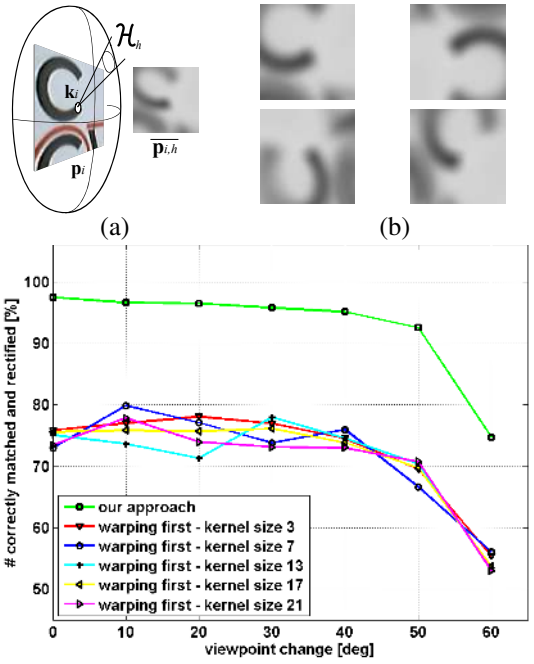

Figure 2. Our one-way descriptor. (a) Our descriptor for a feature point $\mathbf{k}_i$ is made of a set of mean patches $\{\overline{\mathbf{p}_{i,h}}\}$, each computed for a small range of poses $\mathcal{H}_h$ from a reference patch $\mathbf{p}_i$ centered on the feature point $\mathbf{k}_i$. (b) Some other mean patches for the same feature point. The examples shown here are full resolution patches for visibility, in practice we use downscaled patches. (c) Computing a set of mean patches clearly outperforms simple blurring of a set of warped patches.

More formally, given a feature point $\mathbf{k}_i$ in a reference image, we compute a set of mean patches $\{\overline{\mathbf{p}_{i,h}}\}$ where each mean $\overline{\mathbf{p}_{i,h}}$ can be expressed as

$$\overline{\mathbf{p}_{i,h}} = \frac{\int_{H \in \mathcal{H}_h} \mathbf{w}(\mathbf{p}_i, H) dH}{\int_{H \in \mathcal{H}_h} dH} \quad (1)$$

- where H represents a pose, in our case a homography,
- $\mathbf{p}_i$ is the patch centered on the feature point $\mathbf{k}_i$ in the reference image,
- $\mathbf{w}(\mathbf{p}_i, H)$ is the same patch but seen under pose H, and
- $\mathcal{H}_h$ is a range of poses, as represented in Figure 2. The $\mathcal{H}_h$'s are defined so that they cover small variations around a fixed pose H_h but together they span the set of possible poses $\bigcup_h \mathcal{H}_h$.

In practice, the integrals in Eq. (1) are replaced by finite sums and the expression of $\overline{\mathbf{p}_{i,h}}$ becomes

$$\overline{\mathbf{p}_{i,h}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}_i, H_{h,j}) \quad (2)$$

where the $H_{h,j}$ are N poses sampled from $\mathcal{H}_h$.

Once the means $\overline{\mathbf{p}_{i,h}}$ are computed, it is easy to match incoming feature points against the database, and get a coarse pose. Given a patch $\mathbf{p}$ centered on such an incoming point with assumed identity $\widehat{id}$, its coarse pose indexed by $\hat{h}$ is obtained by looking for:

$$\widehat{h} = \underset{i=\widehat{id},h}{\operatorname{argmin}} \|n(\mathbf{p}) - n(\overline{\mathbf{p}_{i,h}})\|^2 , \tag{3}$$

with $n(\cdot)$ a normalizing function that subtracts the mean coordinate and divides by the standard deviation of the vectors. This function makes the matching robust to light changes.

However, computing the $\overline{\mathbf{p}_{i,h}}$ using Eq.(2) is very inefficient because it would require the generation of too many samples $\mathbf{w}(\mathbf{p}_i, H_{h,j})$. In practice, to reach decent results, we had to use at least 300 samples. This takes 1.1 seconds to generate [1], which was not acceptable for interactive applications. We show below that these means can actually be computed very quickly, *independently of the number of samples used*. We first decompose the reference patch $\mathbf{p}_i$ into its principal components:

$$\mathbf{p}_i \propto \overline{\mathbf{v}} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l \tag{4}$$

where $\overline{\mathbf{v}}$ and the $\mathbf{v}_i$ are respectively the mean and principal components —computed offline— of a large set of image patches centered on feature points and L is the dimension of the principal components. The expression of $\overline{\mathbf{p}_{i,h}}$ from Eq.(1) becomes

$$\overline{\mathbf{p}_{i,h}} \propto \frac{1}{N} \sum_j \mathbf{w}(\overline{\mathbf{v}} + \sum_{l=1}^{L} \alpha_l \mathbf{v}_l, H_{j,h}) . \tag{5}$$

A crucial remark that will make our approach efficient is that the warping function $\mathbf{w}(\cdot, H)$ is a linear function, as discussed in the following section.

3.2. Warping is a Linear Function

As illustrated by Fig. 3, we show in this section that the warping function $\mathbf{w}(., H)$ is a linear function. This property will be very useful to simplify Eq.(5) and speed up the computation of the $\overline{\mathbf{p}_{i,h}}$s.

If a simple intensity interpolation strategy is used when warping, for example picking the intensity of the pixel closest to the warped location, then the warping can be expressed by a simple permutation of the pixel intensities between the original patch and the patch after warping.

[1] All times given in this paper were reached on a on a standard notebook (Intel(R) Centrino Core(TM)2 Duo with 2.6GHz and 3GB RAM and an NVIDIA quadro FX3600M with 512MB).

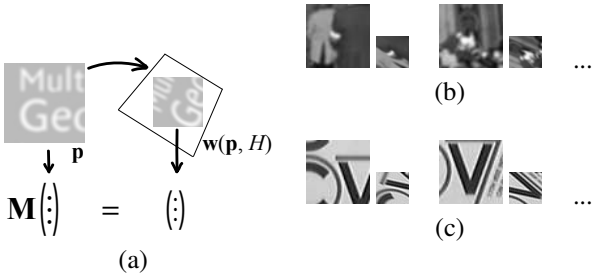

Figure 3. Warping is a linear function. (a) There is a linear transformation $\mathbf{M}$ between the vector made of the intensities of a patch $\mathbf{p}$, and the vector made of the intensities of the same patch after warping $\mathbf{w}(\mathbf{p}, H)$. To be complete, for this property to be true, we have to make sure that no new parts appear in $\mathbf{w}(\mathbf{p}, H)$, and we consider a large enough original patch $\mathbf{p}$. (b) To experimentally check this property, we took a large set of pairs of original patches and their warped version under a given pose, and computed the matrix $\mathbf{M}$ by regression. (c) By applying $\mathbf{M}$ to the intensity vectors of new original patches, we retrieve the vectors for their warped versions, which validates the hypothesis.

The permutation transformation itsself is a linear function. When a more complex interpolation method is used, warping is not as simple as a permutation. However most of the intensity interpolation functions, if not even all, are linear in the pixel intensities, and the function $\mathbf{w}(., H)$ is therefore still linear. Even if a more complex intensity interpolation function is considered and $\mathbf{w}(., H)$ is not "exactly linear", the linear approximation is good enough for our purpose.

One issue remains. When warping the image to create a new patch, new parts can sometimes appear in the generated patch, compared to the original one, making the function $\mathbf{w}(., H)$ nonlinear. To solve this issue, we can simply take the original patch larger than the warped patch, and large enough so that there are never parts in the generated patch which were not present in the original patch. The function $\mathbf{w}(., H)$ then becomes a permutation followed by a projection, and this composition remains a linear transformation.

To convince the reader of the linearity of the warping function, we performed the following experiment, illustrated by Fig. 3. We first picked a transformation H, and warped many random patches under this transformation. This gave us a set of pairs of original patches and their warped versions. We then computed by regression the matrix that transforms the vectors made of the intensities of the original patches into the vectors made of the intensities of the patches after warping. By applying this matrix to the intensity vectors of new original patches, we retrieve the vectors for their warped versions, which validates the hypothesis.

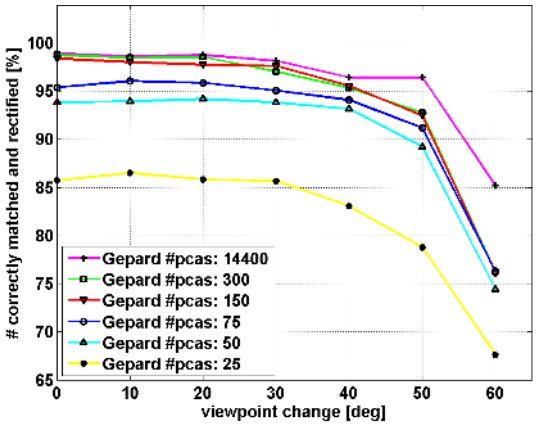

Figure 4. Comparison of correct identity and coarse pose estimation percentage against viewpoint change on the Graffiti image set, for different numbers of principal components. Using only 150 components over 14400 gives results comparable to the full method but is more than 70 times faster.

3.3. An Efficient Way to Compute the Means

Since $\mathbf{w}(., H)$ is a linear function, Eq.(5) becomes

$$\overline{\mathbf{p}_{i,h}} \propto \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\overline{\mathbf{v}}+\sum_{l=1}^{L}\alpha_{i,l}\mathbf{v}_l, H_{j,h}) \quad (6)$$

$$= \frac{1}{N}\sum_{j=1}^{N}\left(\mathbf{w}(\overline{\mathbf{v}}, H_{j,h})+\sum_{l=1}^{L}\alpha_{i,l}\mathbf{w}(\mathbf{v}_l, H_{j,h})\right) \quad (7)$$

$$= \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\overline{\mathbf{v}}, H_{j,h})+\sum_{l=1}^{L}\frac{\alpha_{i,l}}{N}\sum_{j=1}^{N}\mathbf{w}(\mathbf{v}_l, H_{j,h}) \quad (8)$$

$$= \overline{\overline{\mathbf{v}}_h}+\sum_{l=1}^{L}\alpha_{i,l}\overline{\mathbf{v}_{l,h}} \quad (9)$$

where $\overline{\overline{\mathbf{v}}_h}$ is the mean of the patches created by warping $\overline{\mathbf{v}}$ under poses from $\mathcal{H}_h$. The $\overline{\mathbf{v}_{l,h}}$s are similar but obtained from the $\mathbf{v}_l$s:

$$\overline{\mathbf{v}_{l,h}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\mathbf{v}_l, H_{j,h})\,. \quad (10)$$

This implies that we can precompute the $\overline{\mathbf{v}_{l,h}}$s during an offline stage, and when we have to insert a new feature point in the database at run-time, we simply have to project it into the eigenspace to compute its $\alpha_{i,l}$s coefficients, and compute its associated means as the linear combinations given by Eq.(9).

Computing the means $\overline{\mathbf{p}_{i,h}}$ becomes therefore very fast, and does not depend on the number of pose samples $H_{j,h}$. In addition, we can limit the number of components to a small value. In practice, as the graph of Fig. 4 shows, using only 150 components and 575 means on 3 different scales already gives reasonably good results. The computation time is then 15 milliseconds for 12×12 patches including the α_is computation while using directly Eq.(2) takes 1.1 seconds. This should also be compared to the computation time for geometric blur. In [4], the authors claim a computation time of "less of 1 second" for only one mean, which makes our approach much faster. Using the GPU we can reduce the processing time even further to only 5.5 milliseconds [1].

3.4. Refinement Step

Once the coarse pose $H_{\widehat{h}}$ is obtained for an incoming point with assumed identity $\widehat{id}$, we use the Inverse Compositional (IC) [1] algorithm together with the hyperplane approximation of [7] to obtain a fine estimate of the pose. This step is similar to what was done in [5, 6]. We describe it quickly for completeness but more details can be found in these papers. We obtain the corrective homography parameters $\mathbf{x}$ using the formula:

$$\mathbf{x} = \mathbf{A}_{\widehat{id}}\left(n(\mathbf{w}(\mathbf{p}, H_{\widehat{h}}^{-1})) - n(\mathbf{p}_{\widehat{id}})\right)\,, \quad (11)$$

where $\mathbf{A}_{\widehat{id}}$ is the matrix of a linear predictor that depends on the patch identity $\widehat{id}$, $n(\mathbf{p}_{\widehat{id}})$ is the normalized vector of the patch in the reference image, and $n(\mathbf{w}(\mathbf{p}, H_{\widehat{h}}^{-1}))$ is a normalized vector of the patch in the input image, rectified using the coarse pose estimate, as done in the IC algorithm.

This method is very efficient and robust. The $\mathbf{A}_i$ matrices are computed by regression from a set of couples made of small random transformations H_s and the corresponding warped patches $\mathbf{w}(\mathbf{p}_i, H_s^{-1})$. The $\mathbf{A}_i$ matrices must be computed at run-time, for each new feature point inserted in the database. In practice, we precompute the transformations H_s and the warped pixel locations in order to compute very quickly the $\mathbf{w}(\mathbf{p}_i, H_s^{-1})$ patches. An ultimate refinement is done using the ESM algorithm [3].

In practice, for one patch we train four matrices $\mathbf{A}_i$ with different ranges of variation from coarse to fine, using downscaled patches of $13 \times 13 = 169$ pixels and 300 training samples. The whole process takes about 29 milliseconds [1].

Finally, as it was done in [5, 6], we select the correct match consisting of the retrieved pose and the feature point identity based on the cross-correlation between the normalized reference patch $n(\mathbf{p}_i)$ and the normalized warped one after refinement. Furthermore, we apply a threshold on the correlation score to remove wrong matches or wrong pose estimates. Thanks to the high accuracy of the retrieved transformation, we can set this threshold quite high, and we use a value of 0.9 in practice.

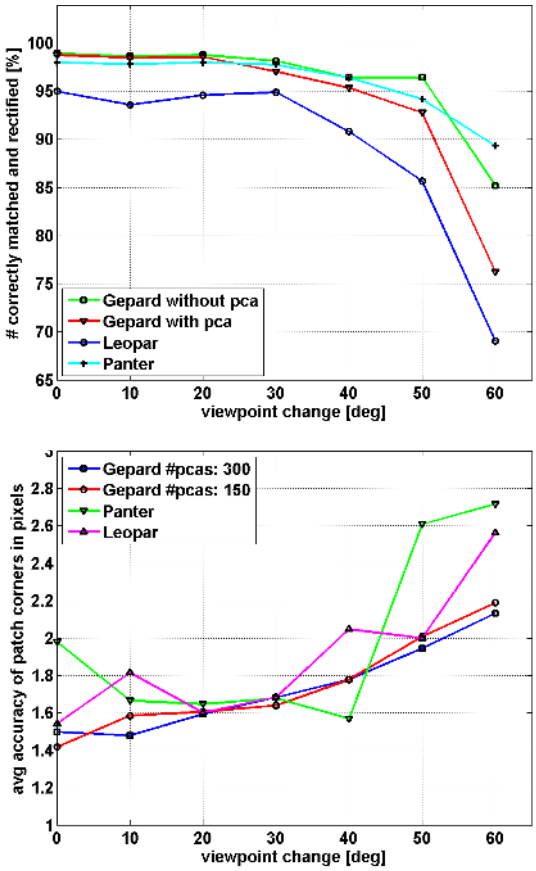

Figure 5. Comparison of our approach with Leopar and Panter. Note these methods have already been proved superior to affine region methods [6, 5]. **Top graph:** Correct identity and coarse pose estimation percentage against viewpoint change on the Graffiti image set. Our approach performs similarly to Panter and better than Leopar (but learning is much faster). **Bottom graph:** Average error of patch corners locations in pixels, against viewpoint change. Thanks to the ESM final refinement, Gepard performs even more accurate than Leopar and Panter.

4. Experimental Results

We limited the comparison of our approach to the other learning-based methods called Leopar and Panter because these methods have already been proved superior to affine region methods in the related papers [6, 5]. We did this comparison on many synthetically warped Graffiti images [9] to obtain a statistically significant statement. Additionally to the warping, we added random synthetic pixel noise and affine illumination change. To compare the performance of the different approaches without taking into account the performance of the initial point detector, we used for each test image the warped feature point location to which we added a uniform noise in the range of [-5;+5] pixels. The results are shown in the three graphs of Fig. 5 and Fig. 4. Our performance measure for the graph in Fig. 4 and for the top graph in Fig. 5 is the percentage of correctly matched and rectified patches against the viewing angle. In the bottom graph of Fig. 5 we show the average accuracy of the corners of all detected patches in pixels. The top graph in Fig. 5 compares the results obtained with our approach, which we call Gepard, and Leopar and Panter. Gepard performs similarly to Panter [5] and better than Leopar [6]. Accuracy was another advantage of Leopar and Panter compared to affine regions detector since they retrieve very accurate full perspective poses. As the bottom graph in Fig. 5 shows, Gepard performs slightly better even with small number of principal components, due to the final ESM refinement.

Leopar [6]	1.05 seconds[1]
Panter [5]	180 seconds
Gepard without PCA	1.1 seconds
Gepard with PCA (CPU)	15 miliseconds
Gepard with PCA (GPU)	5.5 miliseconds

Table 1. Average learning time for the first step for the different approaches. Our approach is more than 70 times faster when the CPU is used.

The advantage of Gepard against Leopar and Panter lies in the learning time. As Table 1 shows, it is much faster than Leopar and Panter. Moreover, the fast version, in which only 150 principal components out of 14400 are used, is more than 70 times faster while it performs only slightly worse (see Fig. 4). When the GPU is used, learning time drops to 5.5 milliseconds, which is largely fast enough for frame rate learning. Learning the refinement stage can be done in additional 29ms on the CPU.

Our current implementation runs at about 10 frames per second using 10 keypoints in the database and 70 candidate keypoints in the input image, on a standard notebook with an Intel Centrino Processor Core2Duo with 2.4GHz and 3GB RAM. We do not use any special data structure for nearest neighbor search and using for example KD-trees [2] would speed it up. Due to the method robustness, one detected patch is enough to detect the target object and to estimate its pose reliably. Some applications are shown in Figs 6, 7, 8, and 9, respectively SLAM localisation, poorly textured object detection, and deformable object detection.

5. Conclusion

We showed how to learn in real-time a method that quickly, robustly and accurately estimates the pose of feature points. We demonstrated our approach on SLAM applications, low-textured object detection and deformable objects registration. However, many other applications could benefit from it, such as object recognition, image retrieval or robot localization.

Acknowledgment: This project was funded by the BMBF project AVILUSplus (01IM08002).

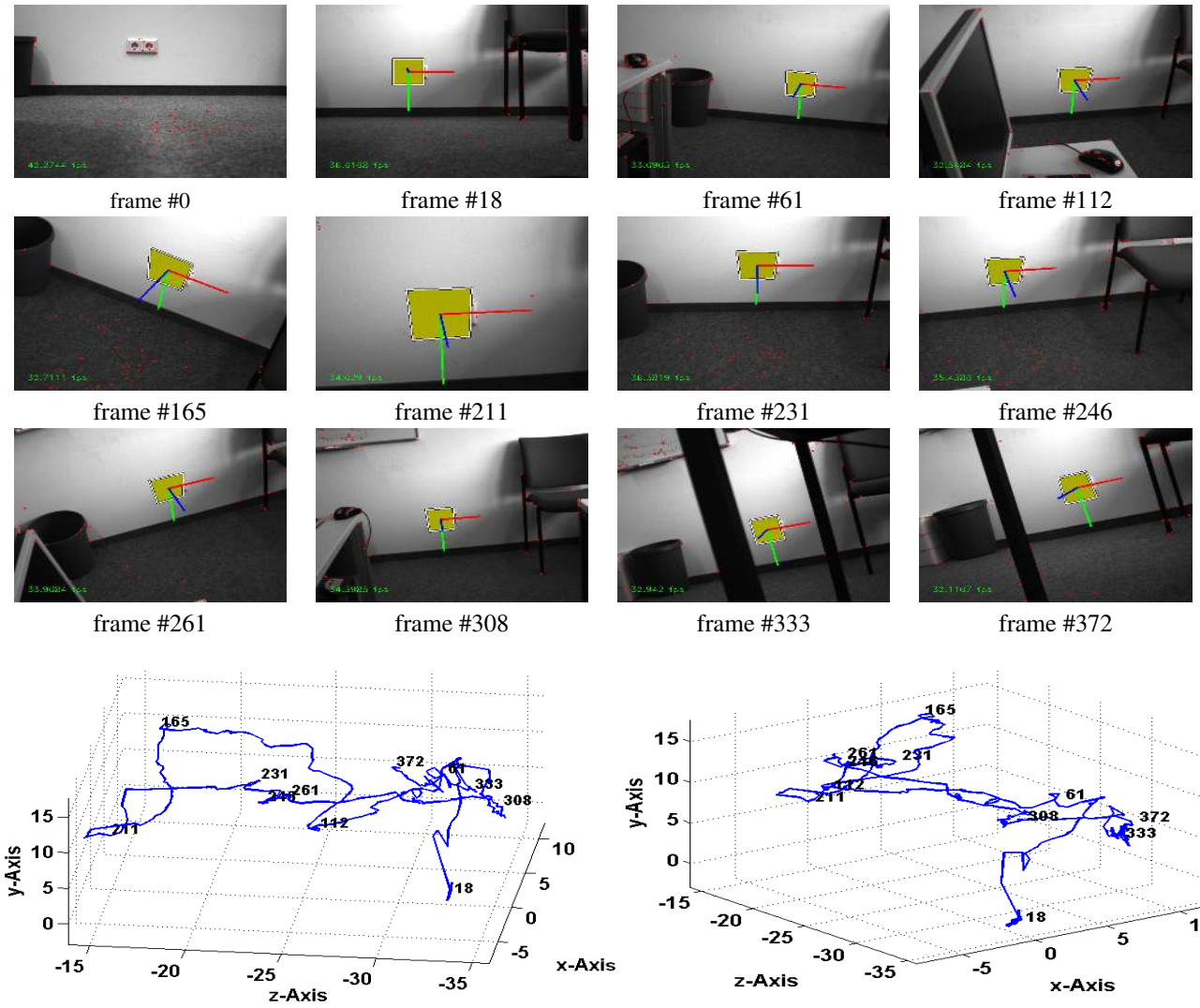

Figure 6. Tracking an outlet. We can retrieve the camera trajectory through the scene despite very limited texture and large viewpoint changes. Since the patch is detected and its poses estimated in every frame independently, the method is very robust to fast motion and occlusion. The two graphs show the retrieved trajectory. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*

References

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56:221–255, March 2004.

[2] J. Beis and D. Lowe. Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *CVPR*, 1997.

[3] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *IJRR*, 26(7):661–676, July 2007.

[4] A. Berg and J. Malik. Geometric blur for template matching. In *CVPR*, 2002.

[5] S. Hinterstoisser, S. Benhimane, V. Lepetit, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *BMVC*, 2008.

[6] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online learning of patch perspective rectification for efficient object detection. In *CVPR*, 2008.

[7] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *PAMI*, 24(7):996–100, 2002.

[8] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004.

[9] 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.

[10] Š. Obdržálek and J. Matas. *Toward Category-Level Object Recognition*. J. Ponce, M. Herbert, C. Schmid, and A. Zisserman (Editors). Springer-Verlag, 2006.

[11] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *CVPR*, June 2007.

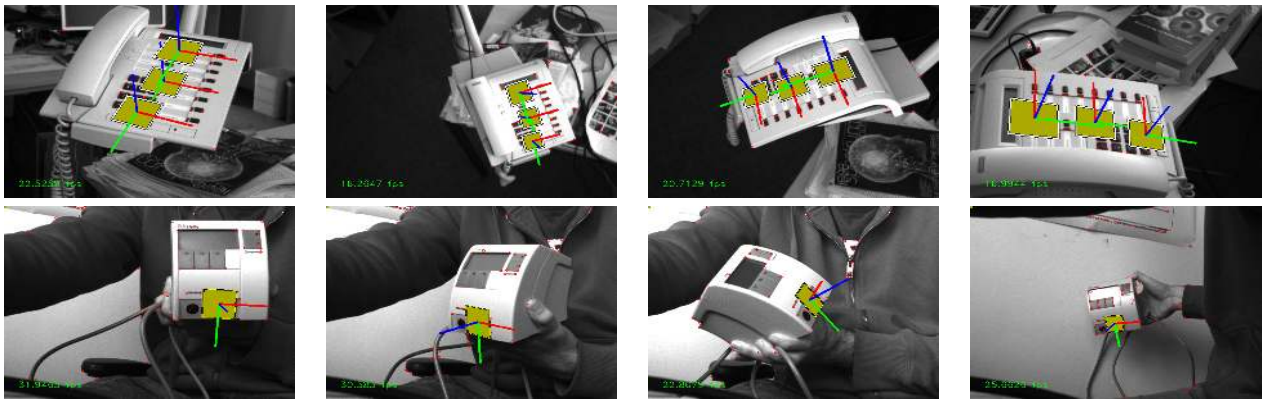

Figure 7. Application to tracking-by-detection of poorly textured objects under large viewing changes. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*

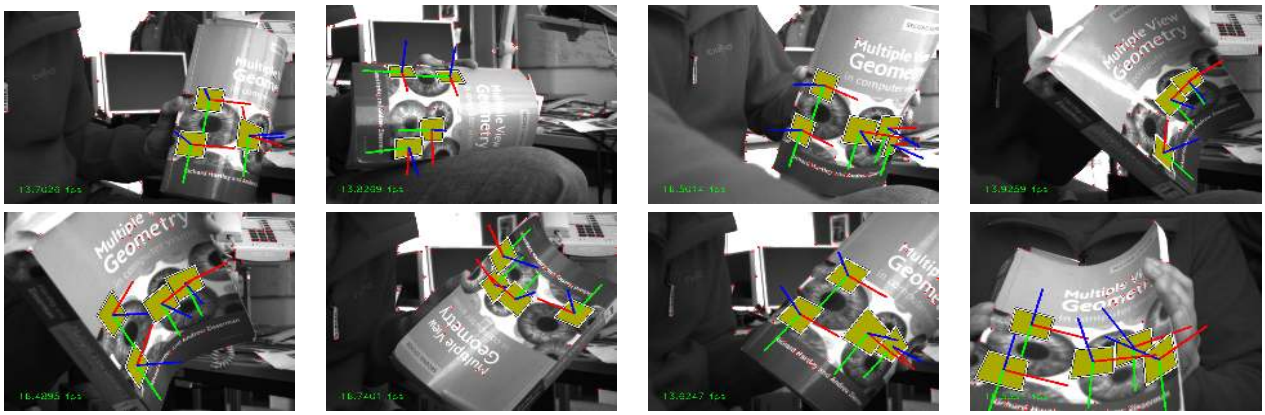

Figure 8. Application to a deformable object. We can retrieve an accurate pose even under large deformations. While it is not done here, such cues would be very useful to constrain the 3D surface estimation. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*

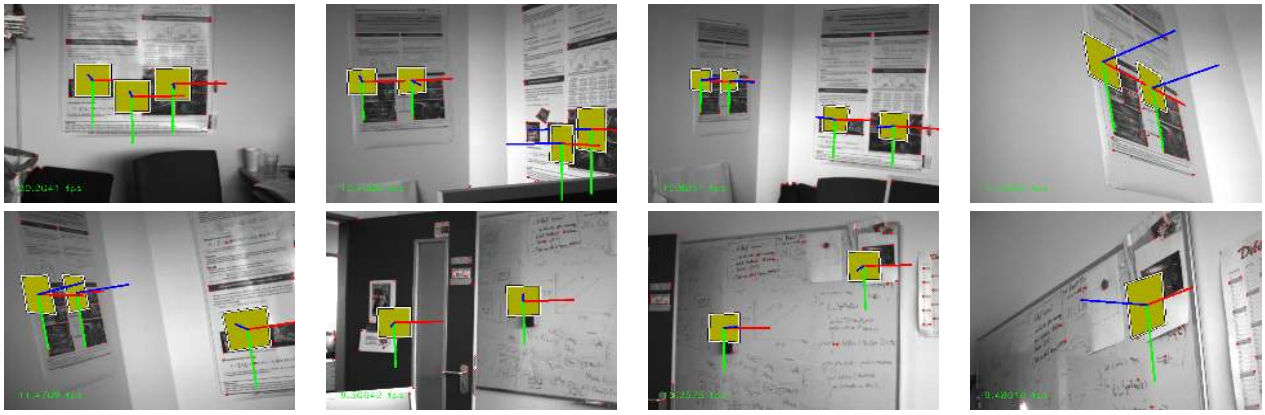

Figure 9. Another example of SLAM relocalisation, using 8 different patches. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*