Learning Real-Time Perspective Patch Rectification

Stefan Hinterstoisser · Vincent Lepetit · Selim Benhimane · Pascal Fua · Nassir Navab

Abstract We propose two learning-based methods to patch rectification that are faster and more reliable than state-of-the-art affine region detection methods. Given a reference view of a patch, they can quickly recognize it in new views and accurately estimate the homography between the reference view and the new view. Our methods are more memory-consuming than affine region detectors, and are in practice currently limited to a few tens of patches. However, if the reference image is a fronto-parallel view and the internal parameters known, one single patch is often enough to precisely estimate an object pose. As a result, we can deal in real-time with objects that are significantly less textured than the ones required by state-of-the-art methods.

The first method favors fast run-time performance while the second one is designed for fast real-time learning and robustness. However, they follow the same general approach: First, a classifier provides for every keypoint a first estimate of its transformation. Then, the estimate allows carrying out an accurate perspective rectification using linear predictors. The last step is a fast verification—made possible by the accurate perspective rectification—of the patch identity and its sub-pixel precision position estimation. We demonstrate the advantages of our approach on real-time 3D object detection and tracking applications.

Keywords Patch rectification · Tracking by detection · Object recognition · Online learning · Real-time learning · Pose estimation

1 Introduction

Retrieving the poses of patches around keypoints in addition to matching them is an essential task in many applications such as vision-based robot localization (Goedeme and Tuytelaars 2004), object recognition (Rothganger et al. 2006) or image retrieval (Chum and Matas 2006; Philbin et al. 2007) to constrain the problem at hand. It is usually done by decoupling the matching process from the keypoint pose estimation: The standard approach is to first use some affine region detector (Mikolajczyk et al. 2005) and then rely on SIFT (Lowe 2004) or SURF (Bay and Tuytelaars 2006) descriptors on the rectified regions to match them.

Recently, it has been shown that taking advantage of a training phase, when possible, greatly improves the speed and the rate of keypoint recognition tasks (Grabner 2007; Ozuysal et al. 2009). Such a training phase is possible when the application relies on some database of keypoints, such as object detection or SLAM. By contrast with Mikolajczyk et al. (2005), these learning-based approaches usually do not rely on the extraction of local patch transformations in order to handle larger perspective distortions.
The advantages of learning for patch recognition and pose estimation. (a) Given a training images or a video sequence, our method learns to recognize patches and in the same time to estimate their transformation. (b) The results are very accurate and mostly exempt of outliers. Note we get the full perspective pose, and not only an affine transformation. (c) Hence a single patch is often sufficient to detect objects and estimate their pose very accurately. (d) To illustrate the accuracy, we use the ‘Graffiti 1’ image and the ICCV booklet cover respectively to train our method and detect patches in the ‘Graffiti 6’ image and in the real scene respectively. We then superimpose the retrieved transformations with the original patches warped by the ground truth homography. (e) Even after zooming, the errors are still barely visible. (f) By contrast, the standard methods retrieve comparatively inaccurate transformations, which are limited to the affine transformation group.

but on the ability to generalize well from training data. The drawback is they only provide a 2-D location, while using an affine region detector provides additional constraints that proved to be useful (Rothganger et al. 2006; Chum and Matas 2006).

To overcome this problem we introduce an approach illustrated in Fig. 1 that can provide not only an affine transformation but the full perspective patch rectification and that is still real-time thanks to a learning stage. We show this is very useful for object detection and SLAM applications: Applying our approach on a single keypoint is often enough to estimate the 3-D pose of the object that the keypoint lies on, provided that a fronto-parallel view of the keypoint is given for training. As a result, we can robustly handle very poorly textured or strongly occluded objects.

More specifically, we propose two methods based on this approach. The first method was called LEOPAR in Hinterstoisser et al. (2008) where it was originally published, and will be refer as ALGO 1 in this paper. It is the faster method, and still more accurate than affine region detectors. The second method was called GEPARD in Hinterstoisser et al. (2009), and will be refered as ALGO 2. ALGO 2 produces the more reliable results and requires only a very fast training stage. Choosing between the two methods depends on the application at hand.

Both methods are made of two stages. The first stage relies on a classifier to quickly recognize the keypoints and provide a first estimate of their poses to the second stage. This second stage uses relatively slower but much more accurate template matching techniques to refine the pose estimate. The difference between the two methods lies in the way the first stage proceeds.

ALGO 1 first retrieves the patch identity and then a coarse pose using an extended version of the Ferns classifier (Ozuysal et al. 2009): To each keypoint in our database we correspond several classes, where each class covers its possible appearances for a restricted range of poses. Due to the Fern structure the computations can be done in almost no time which results in a very fast runtime performance.

Unfortunately the Ferns require a long training stage and a large amount of memory. ALGO 2 dramatically decreases the training time and the memory requirements and is even more accurate; however, it is slower than ALGO 1 at runtime. In ALGO 2, the Ferns classifier is replaced by a sim-