Learning Local Patch Orientation with a Cascade of Sparse Regressors

Alain Pagani
http://av.dfki.de/~pagani
Didier Stricker
http://av.dfki.de

DFKI
Augmented Vision Lab
Technical University of Kaiserslautern
Kaiserslautern, Germany

Abstract

We present a new method for infering the local 3D orientation of keypoints from their appearance. The method is based on the idea that the relation between keypoint appearance and pose can be learnt efficiently with an adequate regressor. Using one reference view of a keypoint, it is possible to train a keypoint-specific regressor that takes the point appearance as input and delivers the local perspective transformation as output. We show that an elegant choice of regressor is a set of sparse regressors applied sequentially in a cascade. In our case, we use a set of parametrized multivariate relevance vector machines (MVRVM) to learn the local 8-dimensional homography from the patch normalized pixel values. We show that using a cascade of regressors, ranging from coarse pose approximation to fine rectifications, considerably speeds up the identification and pose estimation process. Moreover, we show that our method improves the precision of classical points detectors, as the location of the point is rectified together with the homography. The resulting system is able to recover the orientation of patches in real time.

1 Introduction

Many computer vision applications need to identify properties of points of interest, or *keypoints*. Keypoint matching [9], for example, associates an identity to each keypoint, creating correspondences which can be used to compute the camera pose. Affine region detectors [11] additionally derive an approximation of the local image transformation. In general, retrieving accurate information about keypoints allows for considerably reducing the complexity of subsequent steps and is therefore highly desirable. Recently, a learning-based method for retrieving the local perspective warping of a patch has been presented [7]. This method uses a classifier which classifies patches into pre-defined quantized pose estimates. An adequate quantization of the pose space is crucial for the succes of the method.

In this paper, we advocate the use of a regressor to learn the pose as a function of the patch appearance. Thanks to the regressor, smooth variations in the patch appearance result in smooth variations of the pose. Our method relies on a learning stage, where examples of randomly warped patches are used to train a set of Relevance Vector Machines (RVM) [13]. RVMs have the clear advantage that they are described through a sparse set of a few vectors (the *relevance vectors*). The sparsity allows for a very fast prediction step, with

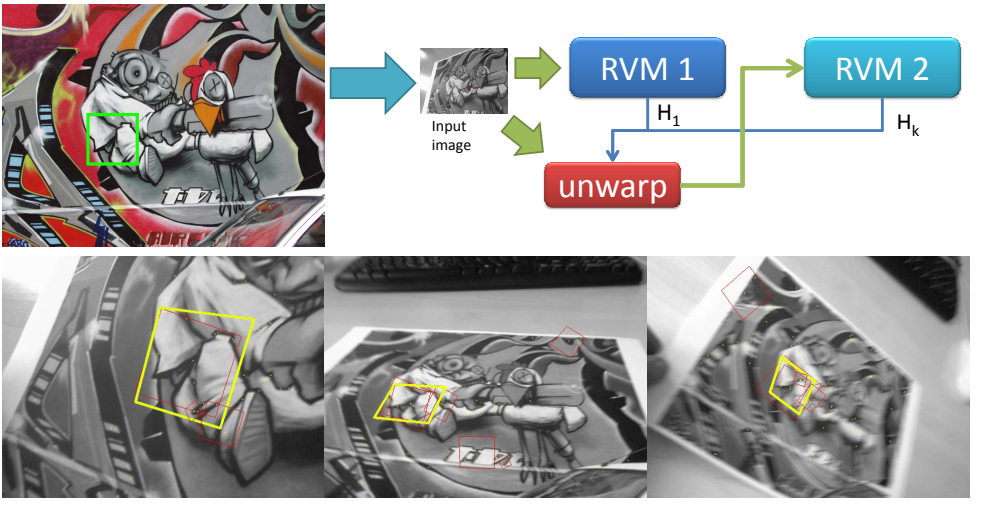


Figure 1: Overview of our approach. From a single reference image (top left), we train a cascade of regressors (top right) to infer the patch orientation from its appearance (bottom).

maintained accuracy. Our method performs in several steps (see figure 1). In a first step, we apply a first RVM, which has been trained to be fast and infer a rough pose estimate from a local square image patch. The local neighborhood of the keypoint is warped using this first estimate, and only a few point hypotheses are kept, based on the similarity of the patch with the reference one. For each of the selected hypotheses, the warped patch is used as input for a second RVM specialized in small variations. The second RVM yields a better estimate of the patch. This last step is repeated until the changes in the pose are small (usually less than 10 iterations). We call this successive use of RVMs a Cascade of RVMs, and our method Caspar, for CAscade of SPArse Regressors. Caspar works for multiple patches, in which case a complete cascade of regressors has to be learnt for each patch.

In the remainder of the paper, we first discuss related work on patch rectification and use of RVMs in computer vision in section 2. We present the different steps of our method in section 3 and show our experimental results in section 4.

2 Related work

2.1 Relevance Vector Machine

The relevance vector machine has been proposed by Tipping [13] to adapt the main ideas of Support Vector Machines (SVM) to a Bayesian context. The results have been shown to be as accurate and sparse as SVMs yet fit naturally into a regression framework and yield full probability distributions as output. RVMs learn a mapping bewteen input vectors $\mathbf{z}$ and output state vectors $\mathbf{x}$ of the form:

$$\mathbf{x} = \mathbf{W}\phi(\mathbf{z}) + \xi, \tag{1}$$

where ξ is a Gaussian noise vector with 0 mean and diagonal covariance matrix. ϕ is a vector of basis functions of the form $\phi(\mathbf{z}) = (1, k(\mathbf{z},\mathbf{z}_1), k(\mathbf{z},\mathbf{z}_2), \ldots, k(\mathbf{z},\mathbf{z}_n))^T$, where k is the kernel

function (any function that compares two input vectors), and $\mathbf{z}_1$ to $\mathbf{z}_n$ are the example input vectors from the training set. The weights of the basis functions are written in the matrix $\mathbf{W}$. In the RVM framework, the weights of each input example are governed by a set of hyperparameters, which describe the posterior distribution of the weights. During training, a set of input-output pairs $(\mathbf{x}_i, \mathbf{z}_i)$ are used to learn the optimal function from equation (1). To achieve this, the hyperparameters are estimated iteratively. Most hyperparameters go to infinity, causing the posterior distributions to effectively set the corresponding weights to zero. This means that the matrix $\mathbf{W}$ only has few non-zero columns. The remaining examples with non-zero weights are called *relevance vectors*.

Tipping's original forumlation only allows regression from multivariate input to univariate output. Here, the patch appearance is used as input vector, and the 8 parameters of the local homography is learnt as output. We therefore use a multivariate extension of the RVM, called MVRVM, proposed by Thayananthan *et al*. [12], and using an EM type algorithm for the training.

RVMs have been used in association with computer vision for different tasks. Agarwal and Triggs [1] presented a method for computing a 3D human pose from silouhettes and shape context distributions. Williams *et al*. used separate RVMs to track the four parameters of a 2D similarity transform of an image region [15]. Both underline the sparsity of RVMs as a noticable advantage in comparison to other techniques like Nearest Neighbor Search, where the entire database has to be kept in memory. Zimmermann *et al*. also propose to solve the region tracking problem using linear regressors in [16], where the idea of using sequentially several regressors is presented. Unlike our method however, only small displacements between two consecutive frames are learnt.

2.2 Local patch orientation

The idea of deducing geometric information from the object appearance through learning was proposed in [4] where Support Vector Machines are used to learn the position of the corners of the object bounding box in relation to its appearance. The 3D orientation problem has been adressed using affine region detectors [11], as they provide an affine approximation of the local transformation of a patch and allow for correcting the distortions induced by this local warping. Since the pioneering work of Baumberg [3], several approaches have been proposed. Among them, the Hessian-Affine detector by Mikolajczyk and Schmid [11] and the MSER detector by Matas *et al.* [10] have been shown to be the most reliable ones. However, the obtained transformation is affine only, and the fully perspective transformation (a homography in the case of a planar patch) cannot be found.

Recently, a method for computing the local homography called *patch rectification* has been presented [6, 7]. In this method, as in ours, a training phase is used to learn to estimate the local transformation of the patch. However, this method uses a classifier to learn the local homography. To achieve this, the pose space has to be quantized in a finite number of classes, which poses the problem of the pose space tesselation: a high number of classes increases the classifier complexity while a small number of classes can lead to poor estimation results.

Our argumentation is that the relation between patch appearance and patch pose is continuous by nature, and we suggest in this paper the use of a regressor to infer the pose of the patch from its appearance. In the next section we present our regression-based method and the cascading approach.

3 Cascade of sparse regressors

3.1 General approach

We consider a set of n 3D points of interest $\{\mathbf{M}^i\}_{i=1}^n$ lying on the surface of a given object. The aim is to find for each of these points a function $\mathbf{f}^i$ which maps the point's appearance, $\mathbf{p}$, to the local homography of the point, $\mathbf{h}^i$. The homographies are parametrized using the projected image coordinates of the 4 corners of a unit square centered at the point (as recommended by [2]), and thus has 8 parameters. Let us first assume that such a set of functions is known. In this case, we can easily identify learnt points *and* compute their local orientation in a unknown image by applying following steps:

1. Apply a rotation and scale invariant corner detector on the image.
2. For each corner: Extract a orientation- and scale-corrected square patch $\mathbf{p}$ of size $s \times s$ from the image. Normalize the patch intensity values and apply steps 3. through 5.
3. For each point of interest $\mathbf{M}^i$: apply steps 4. and 5.
4. Estimate the corresponding homography with the mapping function $\mathbf{h}^i = \mathbf{f}^i(\mathbf{p})$.
5. Using the estimated homography, unwarp the image patch and compute its normalized cross-correlation ncc^i with a canonical view of the point $\mathbf{M}^i$.
6. If $\max_i ncc^i > \tau$ then the *identity* of the point is $I = \arg\max_i ncc^i$ and its local orientation is given by $\mathbf{h}^I$. τ is a correlation threshold which can be high. In practice we used $\tau = 0.9$.

As the pseudo-code shows, this algorithm tests every 2D corner with the mapping function of every 3D point of interest. Therefore, it can be quite time-consuming if the functions $\mathbf{f}^i$ are costly. We solve this problem by introducing two key ingredients in our method: *sparse regression* and *regression cascading*. With sparse regression, we mean using a machine that needs to retain only a fraction of the examples in the training set, like *e.g.* Support Vector Machines [5], leading to an efficient prediction step. With cascading, we mean that the function $\mathbf{f}^i$ is implemented as a cascade of functions with increasing precision and complexity, which permits an early termination in the spirit of the Attentional Cascade of [14]. In the next subsections, we assume a 3D point of interest $\mathbf{M}$ is given, and we describe our choice for the corresponding mapping function $\mathbf{f}$ and its training.

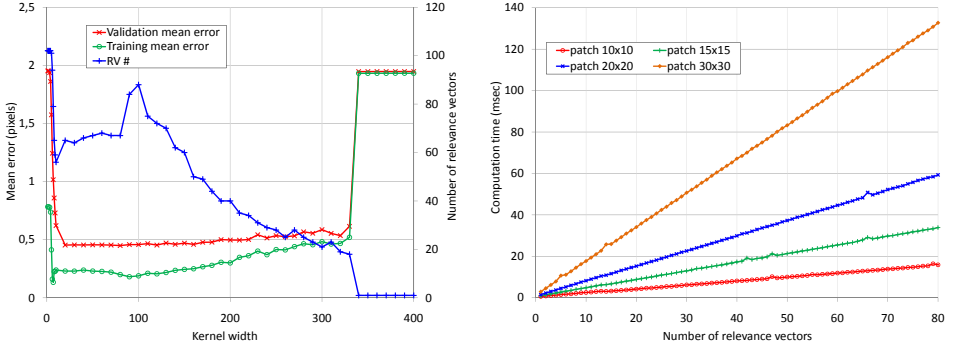
3.2 Cascade levels

The input of a single cascade level k is the current estimate of the homography. For the first level, this estimate is a (orientation- and scale-corrected) 2D square patch of a predefined constant size s_1. For the other levels, the estimate is the output of the previous level. Using this estimate, the image patch is unwarped to a square patch, resampled to a level-dependent size s_k and normalized to have a zero mean and unit standard deviation. The normalization adds robustness to light changes. For each cascade level, we train one Multivariate Relevance Vector Machine (MVRVM), using a quadratic kernel functions of the form

$$if \quad \|\mathbf{p}_1 - \mathbf{p}_2\| < 2, \;\; k(\mathbf{p}_1, \mathbf{p}_2) = 1 - \frac{\|\mathbf{p}_1 - \mathbf{p}_2\|^2}{4\sigma_k^2} \quad , else \;\; k(\mathbf{p}_1, \mathbf{p}_2) = 0 \tag{2}$$

where σ_k is a level-dependent parameter governing the width of the kernel.

When building the regressor cascade, we can use the two level-dependent parameters s_k and σ_k to create regressors with different properties: Using a small patch size s_k reduces the size of the input vector and thus increases the prediction speed, but reduces the regressor precision as the input patch has a smaller resolution. On the contrary, a large patch size s_k will produce a slower but more precise regressor. The parameter σ_k governs the width of the kernel and thus the sparsity of the RVM: If it is large the number of relevance vectors is reduced and the regressor faster, at the cost of possibly poor results. Conversely, if σ_k is smaller, the RVM is more precise while being slower. However, care has to be taken that σ_k is not too small to prevent overfitting, which leads to poor generalization properties. Figure 2 shows the effect of σ_k on the precision and the number of relevance vectors.



(a) (b)

Figure 2: (a): Effects of the kernel width σ_k on the training / validation error and on the number of relevance vectors. Increasing the kernel width slighlty increases the validation error, but significantly reduces the number of relevance vectors needed, up to a critical point (here around 300). (b): Predicition time for 150 candidate keypoints w. r. t. the number of relevance vectors for different patch sizes. The computation time is experimentally shown to be proportional to the number of RVs and to the square of the patch size. Using such tests allows us to find an optimal σ_k for a given regressor.

Our idea is to design the cascade in such a way that the first regressor is very fast while providing a very coarse estimate of the homography. A similarity test based on the normalized cross correlation rejects most of the candidate already after the first step. The remaining hypotheses are then pushed to more complex regressors with a increasing precision upon several levels. More specifically, let $\mathbf{p}_1$ be a orientation and scale corrected patch of size s_1 found in the image. We start by applying the first, fast and coarse regressor $\mathbf{f}_1$:

$$\mathbf{h}_1 = \mathbf{f}_1(\mathbf{p}_1) \tag{3}$$

and compute the score of the patch

$$S_1 = ncc(\mathbf{p}^*, u(\mathbf{p}_1, \mathbf{h}_1)) \tag{4}$$

where $u(\mathbf{p},\mathbf{h})$ is the function unwarping the patch $\mathbf{p}$ using the homography $\mathbf{h}$, ncc is the normalized cross correlation between two patches, and $\mathbf{p}^*$ is a canonical view of the patch. If S_1 is smaller than a threshold τ_1, then the patch is rejected. Otherwise, the patch is processed

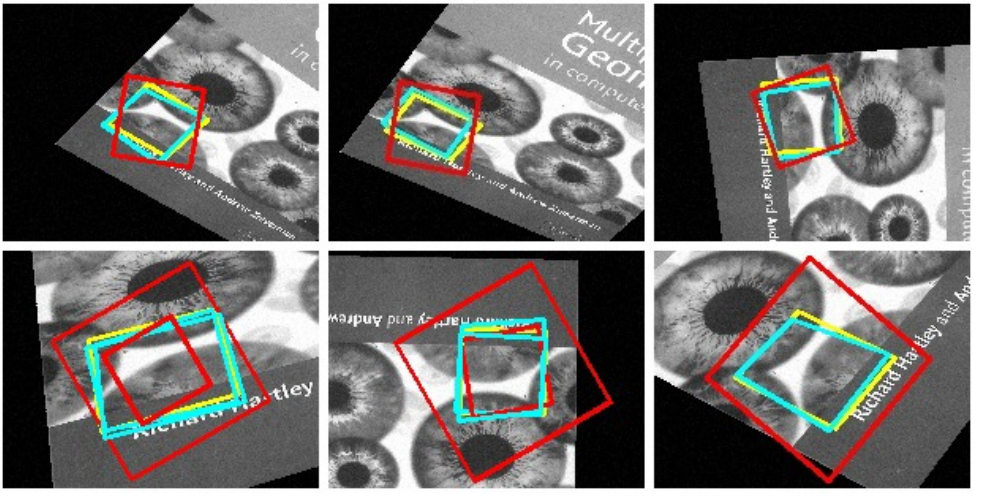


Figure 3: Training data. The yellow patch is the true homography. The red squares are the 2D patches used as input for the first cascade level. The blue patches are the disturbed homographies used to learn the subsequent cascade levels. Several red squares indicates that the point has been detected at several scales.

by the next cascade level. Thus, the entire cascade can be described by repeated applications of tests:

$$\textit{if } ncc(\mathbf{p}^*, u(\mathbf{p}_k, \mathbf{h}_k)) > \tau_k, \quad \textit{then } \mathbf{h}_{k+1} = \mathbf{f}_{k+1}(u(\mathbf{p}_k, \mathbf{h}_k)), \quad \textit{else reject patch} \tag{5}$$

Thus, each level is governed by the set of parameters (s_k, σ_k, τ_k). These parameters are set during learning as described in the next subsection.

3.3 Learning a cascade of regressors

The training set consists of examples obtained by randomly rendering hundreds of views of the object with known random poses, adding realistic noise level and intensity changes, and storing the normalized unwarped patch as input and the known homography as output. Figure 3 shows examples of training data. In practice, we collect 200 examples for training and 200 for verification for each point of interest. We distinguish between the first cascade level where no homography approximation is known and the subsequent levels where an estimate is provided.

For the first level of the cascade, the input is simply the 2D square patch. We store as output the real homography computed from the known pose. For the subsequent levels, we construct rectification regressors by adding a given level of disturbance to the real homography. To this aim, we simply add Gaussian noise to the real patch center and to the real patch corners.

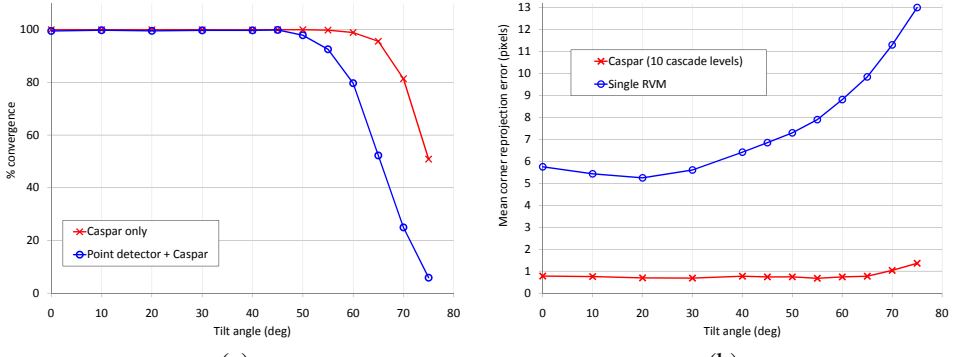
During learning, we automatically adapt the kernel width σ_k through training/validation experiments to achieve a given type of regressor: the first level should be fast, so we set a small value for s_1 and adapt σ_1 to keep the number of relevance vectors under a given maximum. Figure 2 shows the precision and number of relevance vectors obtained for different

values of σ_k. Using such tests allows us to choose an optimal value for σ_k. In practice, we used $s_1 = 20$ and a maximum number of relevance vectors of 25. The correlation threshold τ_1 is set to keep the number of hypotheses lower than 5 (in practice we used $\tau_1 = 0.4$). For the remaining levels ($k = 2$ to 10), the patch size is set to $s_k = 30$ and we add Gaussian noise to the patch corners and center with $\sigma_k^r = 2$. The kernel size σ_k is automatically adapted to keep the number of relevance vectors under 100.

4 Evaluation and results

In this section, we present the experimental results of our tests with different objects. For a given object, a set of points of interest have to be defined first. In order to find good points of interest, we render a large number of views of the object and apply the rotation and scale invariant point detector of [8] on the rendered image. The found 2D points are back-projected onto the object and feed a 3D point accumulator. The 3D points accumulating the most votes are selected as good points of interest, having a large probability to be found by the 2D point detector. We then train our casade of regressors for one or several points of the object. In all our experiments, we use a cascade trained for tilts up to 70 degrees, using a set of 200 random examples for the training.

4.1 Robustness to viewpoint change



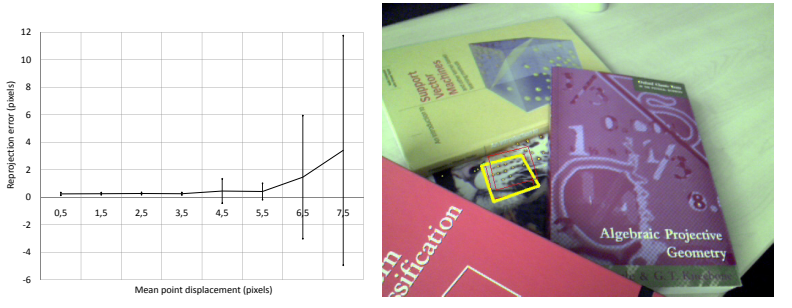
(a) (b)

Figure 4: Robustness to viewing angle variations. (a) Percentage of correctly retrieved orientations. (b) Mean corner reprojection error for Caspar and a direct regression approach (Single RVM). See text for details.

In a first experiment, we measure the robustness to viewpoint change by generating synthetic views with random orientation and a given tilt angle ranging from 0 to 75 degrees. For each view, we apply our method to identify a patch and find its orientation. We repeat this test 2000 times for each tilt angle, and report the percentage of correctly computed homographies. In figure 4 (a), we measure the ratio of correctly retrieved orientations for two cases: the red curve (Caspar) shows the success rate obtained by Caspar only, *i.e.* the ratio of correctly retrieved orientations among all test images where at least one point has been

found by the corner detector in the vicinity of the real position. The blue curve (Point detector of [8] + Caspar) shows the success rate obtained when considering the corner detector and Caspar (including the cases where no corner has been found in the vicinity of the real point). Caspar has a high succes rate when a corner has been found, even for large tilts: For a tilt of 60 degrees, our method retrieves the right homography over 98.9% of the time. This is much better than the method of [8] which reports convergence rates between 79% and 95% for the same tilt. Note the good convergence rate of 50% for extremely large tilts of 75 degrees. However, when considering the point detector of [8] and Caspar together, the convergence rate drops when the tilt exceeds 50 degrees. This shows that the point detector of [8] is not adequate for large tilts, as it does not detect the point of interest robustly in this case. Figure 4 (b) shows the mean reprojection error for the 4 corners of the patch using one single RVM and our cascaded approach. Our method has a mean reprojection error of less than 1 pixel over almost the entire tilt range, whereas a single RVM achieves at best 5 pixels reprojection error (an up to 13 for large tilts). This justifies the use of a convergence cascade.

4.2 Improving the point detector's precision



(a) (b)

Figure 5: (a) Improving the point detector's precision. Our method converges to the right point even if up to 4 pixels displacement is added to the correct point. (b) Our method applied to a covered image of a jaguar.

To assess the robustness to point detector imprecision, we applied our method to a ground truth keypoint with a small amount of displacement, for different ranges of distances. Figure 5 shows the reprojection error after convergence for increasing distance means (for each mean, random keypoint locations were generated in a unit range centered on the mean). The graph shows the mean reprojection error for the center of the patch after convergence, and twice the standard deviation around this mean. This experiment clearly shows that our method is able to recover from point detector errors up to 4 pixels.

4.3 Application

We applied our method to identify a patch and infer its orientation for objects with different textures. Figure 1 shows the example of an image from the Graffiti set of [11], figure 5 (b) shows a patch retrieved on a picture of a jaguar even under severe occlusion, and figure 6 shows several frames of a video sequence with fast movements, motion blur and large scale variations. Several complete video sequences are available on the authors' website at

http://av.dfki.de/~pagani. Please note that every frame is treated independently, whithout using information from the previous frames. Nevertheless, our method does not produce jitter, which usually affects such single-frame methods. In the examples, the red patches are the hypotheses detected by the first cascade level. The yellow patch is the result returned by our system. The time needed to learn a patch is about 1.5 minutes for the first RVM and 3 minutes remaining levels. Once the system has been trained, our system is able to detect one patch and find its orientation in video real time (28 frames /sec).

Figure 6: Estimation of the local patch transformation. Our system detects the learnt point and estimates its pose in real time, even in presence of large scale variations. *A full video is available on the authors' website at* http://av.dfki.de/~pagani.

5 Conclusion

We have introduced a novel method for computing the local homography of an object that uses directly its appearance as input for a regression scheme. The efficiency of our method relies on two key ideas: first, the usage of *sparse* regressors (in our case, multivariate relevance vector machines), and second, the design of the regressor as *cascade*. Both ideas contribute to speeding up the orientation recovery process. We have shown that using a cascade instead of a single regressor improves the precision, and that the main bottleneck of our approach is the lack of repeatability of the point detector for large tilts. In our future work, we will therefore investigate the use of better point detectors. Because we use Baysian regressors, another direction of research is the the fusion of multiple patches orientations for the computation of the global camera pose.

Acknowledgement

This work has been partially funded by the BMBF project AVILUSplus (01IM08002).

References

[1] Ankur Agarwal and Bill Triggs. Recovering 3D human pose from monocular images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 28:44–58, 2006.

[2] Simon Baker, Ankur Datta, and Takeo Kanade. Parameterizing homographies. Technical Report CMU-RI-TR-06-11, Robotics Institute, Carnegie Mellon University, 2006.

[3] Adam Baumberg. Reliable feature matching acrosswidely separated views. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[4] Matthew B. Blaschko and Christoph H. Lampert. Learning to localize objects with structured output regression. In *European Conference on Computer Vision (ECCV)*, 2008.

[5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Workshop on Computational Learning Theory*, 1992.

[6] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *British Machine Vision Conference (BMVC)*, 2008.

[7] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online learning of patch perspective rectification for efficient object detection. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[8] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 28:1465–1479, 2006.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60:91–110, 2004.

[10] J. Matas, O. Chum, M.Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference (BMVC)*, 2002.

[11] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. *Int. Journal of Computer Vision*, 65:43–72, 2005.

[12] A. Thayananthan, R. Navaratnam, B. Stenger, P. H. S. Torr, and R. Cipolla. Multivariate relevance vector machines for tracking. In *European Conference on Computer Vision (ECCV)*, 2006.

[13] Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[15] Oliver M. C. Williams, Andrew Blake, and Roberto Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 27:1292–1304, 2005.

[16] Karel Zimmermann, Jiri Matas, and Tomas Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 31:677–692, 2009.