Efficient Learning of Linear Predictors using Dimensionality Reduction

Stefan Holzer, Slobodan Ilic, David Tan, Nassir Navab

Department of Computer Science, Technische Universität München (TUM),
Boltzmannstrasse 3, 85748 Garching, Germany

Abstract. Using Linear Predictors for template tracking enables fast and reliable real-time processing. However, not being able to learn new templates online limits their use in applications where the scene is not known a priori and multiple templates have to be added online, such as SLAM or SfM. This especially holds for applications running on low-end hardware such as mobile devices. Previous approaches either had to learn Linear Predictors offline [1], or start with a small template and iteratively grow it over time [2]. We propose a fast and simple learning procedure which reduces the necessary training time by up to two orders of magnitude while also slightly improving the tracking robustness with respect to large motions and image noise. This is illustrated in an exhaustive evaluation where we compare our approach with state-of-the-art approaches. Additionally, we show the learning and tracking in mobile phone applications which demonstrates the efficiency of the proposed approach.

1 Introduction

Template tracking is an extensively studied field in Computer Vision with a wide range of applications such as augmented reality, human-computer interfaces, medical imaging, surveillance, vision-based control and visual reconstruction. The main task of template tracking is to follow a template in an image sequence. This is done by estimating the parameters of the template warping function that defines how the pixel locations, occupied by the template, are warped to the next frame of the image sequence. Examples for such warping functions are affine transformations or homographies.

Recently, tracking-by-detection methods became popular since they reached a state where they are able to track close to or at real-time performance. However, they show some limitations which we further address in Sec. 2. In frame-to-frame template tracking, image intensity differences between template areas of two consecutive frames have to be minimized in terms of the template warping parameters. Most of them are based on energy minimization [3–11] and in many cases, an analytical derivation of the Jacobian is used in order to provide real-time tracking capabilities. Alternative approaches are based on learning [1, 12–17, 2] where the relation between image intensity differences and template

warping parameters is learned. While energy minimization is flexible at run-time, learning based methods have proven to allow much faster tracking.

Jurie and Dhome [1] proposed a very successful learning based template tracker which learns Linear Predictors to efficiently compute template warp parameter updates. This is very fast in tracking and tends to avoid local minima. But, due to the computationally expensive learning phase, online-creation of templates is hardly possible. This, however, is a crucial ability for many applications that have to deal with data which is not available for prior offline learning. Some examples for such applications are Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM). We address this limitation by introducing a more efficient learning procedure for creating Linear Predictors. This not only improves the learning speed drastically, but also brings a small improvement in robustness of tracking with respect to large motions and image noise.

The remainder of the paper is structured as follows: first, we discuss related work on template tracking (Sec. 2) and introduce the original approach of Jurie and Dhome (Sec. 3). This is followed by a detailed description of our approach (Sec. 4) and an extensive quantitative testing (Sec. 5.1 and Sec. 5.2). Finally, we demonstrate that the proposed approach can be used for efficient template learning and tracking on mobile phones, as well as applications similar to SLAM where multiple templates are being tracked simultaneously (Sec. 5.3).

2 Related Work

The existing template tracking approaches can be categorized in mainly three different sets of methods: tracking-by-detection (TBD) [18–22], template tracking based on energy minimization [3–11, 23, 24], and methods that utilize learning [1, 12–17]. While tracking-by-detection methods are able to track a template over the whole image independent of the previous position, they hardly achieve the processing speed of frame-to-frame tracking. Additionally, they often require a time consuming training procedure and are limited in their possible pose space. For frame-to-frame tracking, energy minimization-based approaches are generally more flexible at run-time by allowing fast creation and modification of templates, while learning-based approaches enable higher tracking speed. Looking at tracking performance, it has been shown in the past that learning-based approaches outperform methods based on energy minimization. Jurie *et al.* [12] demonstrated that Linear Predictors are superior to Jacobian approximation and Holzer *et al.* [2] showed an experiment where Linear Predictors are superior to Efficient Second-order Minimization (ESM) [11]. We further fortify the latter by showing additional comparisons in Sec. 5.2.

Tracking-by-Detection-based approaches. Some of the most prominent work on patch-based TBD was recently proposed by Hinterstoisser *et al.* [18–20]. Their former two methods, called Leopard [18] and Gepard [19], use the patch around detected keypoints for matching and pose estimation. While these methods enable near real-time performance, they heavily rely on the repeatability of the

underlying keypoint detector. Additionally, they apply template tracking approaches for pose refinement, which means that these approaches also benefit from advances in template tracking. To overcome the dependency on keypoint detectors, they proposed a template matching based approach (DOT) [20]. However, this requires to learn templates for every possible pose, which restricts the application space and makes it comparably slow in contrast to frame-to-frame tracking. Özuysal *et al.* (FERNs) [22] extract keypoints and match them using a classification-based approach by estimating the probability on which class the keypoints belong to. Although this gives real-time performance, it includes a time consuming learning stage and needs a sufficient number of keypoints visible. This makes it less useful to track small regions. Holzer *et al.* (DTTs) [21] proposed a detection based approach which builds on finding closed contours and matches them using a similar approach as [22] used for keypoint matching. However, this includes a time consuming learning stage and detection speed was reported at 10 fps only.

Energy minimization-based approaches. Numerous approaches have followed the work of Lucas and Kanade [3]. They consist of different update rules of the warp function [3, 5, 6, 4, 7, 8], handling of occlusions and illumination changes [5], as well as considering different orders of approximation of the error function [10, 11]. The different update rules of the warp function can be classified into four types, namely, the additive approach [3], the compositional approach [4], the inverse additive approach [5, 6] and the inverse compositional approach [7, 8], where the inverse approaches switch the roles of the reference and current image. As a consequence, it is possible to transfer some of the computation to the initialization phase, which makes the tracking computationally more efficient. Compensation of illumination changes and occlusions was addressed by Hager and Belhumeur [5]. Faster convergence rates as well as larger convergence areas can be additionally obtained by using a second-order instead of a first-order approximation of the error function [10, 11]. A more detailed overview of energy-based tracking methods is given by Baker and Matthews [9].

Learning-based approaches. Jurie and Dhome [1] proposed a method that learns Linear Predictors using randomly warped samples of the initial template, while using the learned Linear Predictors to predict the parameter updates in tracking. Here, the "Jacobians" are computed once for the whole method and a parameter update is computed using a simple matrix multiplication. More details on this are given in Sec. 3.2. The same authors extended their approach to handle occlusions [12]. Invariance to illumination changes was introduced by Gräßl *et al.* [25]. They [13] also formulated a method on how to select the points for sampling from image data to further increase accuracy in tracking. Zimmermann *et al.* [17] use numerous small templates and track them individually. Based on the local movements of these small templates, they estimate the movement of a large template. Holzer *et al.* [2] start with a small template and grow it until a large template is constructed online. This idea showcased a way to adapt existing Linear Predictors to modify the shape of a template at run-time. Mayol

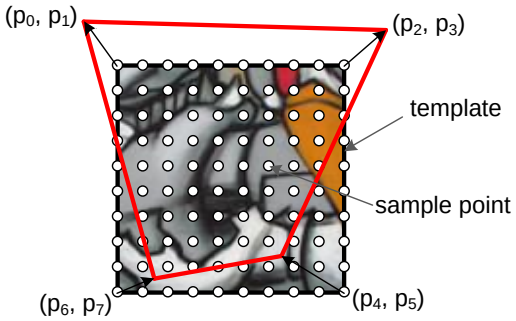


Fig. 1. A template is represented by a set of regularly placed sample points. Its pose is parameterized using its four corner points.

and Murray [16] stepped back from Linear Predictors by presenting an approach that fits the sampling region to pre-trained samples using general regression.

All the proposed learning approaches, however, are not able to learn large templates online. To overcome this limitation, we introduce a learning scheme, which is different to the one proposed by Jurie and Dhome [1] and enables online learning of templates.

3 Tracking Framework

Our proposed template tracking approach is based on the work of Jurie and Dhome [1]. While we introduce a new learning method in Sec. 4, the tracking stage itself stays the same as in [1]. Therefore, we first introduce our notations and review the method proposed by Jurie and Dhome [1].

3.1 Template and Parameter Description

Without loss of generality, we consider a $w \times h$ template with an area of $n_s = w \cdot h$ pixels within an image. Instead of using the full-resolution template, we apply a uniform subsampling as shown in Fig. 1 to obtain a grid of n_p sample points. However, neither the approach of Jurie and Dhome [1] nor our approach is restricted to this sample point arrangement or rectangular shapes.

The pose of the template is described using the parameter vector $\boldsymbol{\mu}$. Within this paper, we use a homography to represent the current perspective distortion of a planar template and parameterize it using the four corner points of the template. This leads to an 8-dimensional vector $\boldsymbol{\mu} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)^\top$ (see Fig. 1). Note that our approach is not limited to this type of transformations and can be easily adapted to any other parameterizable template deformation.

3.2 Template Tracking based on Linear Predictors

The goal of template tracking is to follow a reference template over a sequence of images. This reference template is defined by an initial parameter vector $\boldsymbol{\mu}_R$

that corresponds to the location of the template in the reference image, and a vector $\mathbf{i}_R = (i_{R,1}, i_{R,2}, \cdots, i_{R,n_p})^\top$ that corresponds to the image intensity at the sample points of the template.

Assuming that the reference template is located in the first frame of a video sequence, the location of the sample points is defined by the initial parameter vector $\boldsymbol{\mu}_R$ while the parameter vector $\boldsymbol{\mu}_C$ defines the location of the template in the current image. Henceforth, tracking is done by computing $\boldsymbol{\mu}_C$. The basic approach for this is to first compute a vector $\delta\mathbf{i} = \mathbf{i}_C - \mathbf{i}_R$ of image differences and then to use this to compute a parameter update $\delta\boldsymbol{\mu}$ which accounts for the present pose difference. Note that the vector $\mathbf{i}_C$ stores the image values extracted from the current image and is extracted by computing the sample point locations using the template pose $\boldsymbol{\mu}_{C-1}$ of the previous image frame.

Instead of explicitly minimizing an error function, *e.g.* by iteratively solving a first- or second-order approximation of the function, Jurie and Dhome [1] use a learned matrix $\mathbf{A}$ (also called as Linear Predictor) to compute $\delta\boldsymbol{\mu}$ based on the vector $\delta\mathbf{i}$ as:

$$\delta\boldsymbol{\mu} = \mathbf{A}\delta\mathbf{i}. \tag{1}$$

In order to compute $\delta\boldsymbol{\mu}$, one needs to precompute the matrix $\mathbf{A}$. This is done by collecting a set of n_t random transformations, where n_t is significantly larger than n_p, together with its corresponding image difference vectors. These random disturbances $\delta\boldsymbol{\mu}_i$ and image difference vectors $\delta\mathbf{i}_i$ are then combined in two matrices $\mathbf{Y} = \left(\delta\boldsymbol{\mu}_1, \delta\boldsymbol{\mu}_2, \ldots, \delta\boldsymbol{\mu}_{n_t}\right)$ and $\mathbf{H} = (\delta\mathbf{i}_1, \delta\mathbf{i}_2, \ldots, \delta\mathbf{i}_{n_t})$. Using these matrices, Eq. (1) can be written as:

$$\mathbf{Y} = \mathbf{A}\mathbf{H} \tag{2}$$

which solves for $\mathbf{A}$ using a closed-form solution:

$$\mathbf{A} = \mathbf{Y}\mathbf{H}^\top \left(\mathbf{H}\mathbf{H}^\top\right)^{-1}. \tag{3}$$

In practice, we normalize the extracted image data with zero mean and unit standard deviation. This increases the robustness against illumination changes. Note that, to prevent $\mathbf{H}\mathbf{H}^\top$ from being rank-deficient due to the zero mean of the data, we have to add random noise to the obtained image value difference vectors.

3.3 Multi-Layered Tracking

For improved tracking performance, we use a multi-predictor approach where multiple Linear Predictors $\mathbf{A}_1, \ldots, \mathbf{A}_{n_l}$ are learned for one template. Among these, $\mathbf{A}_1$ is trained for large distortions and the subsequent ones for smaller distortions. Intuitively, $\mathbf{A}_1$ accounts for large motions but is less accurate, while $\mathbf{A}_{n_l}$ can handle only small template motions but has improved accuracy. During tracking, each Linear Predictor is utilized several times before the next level is used. Within this paper, we use $n_l = 5$ and three iterations for each predictor.

4 Efficient Predictor Learning using Dimensionality Reduction

As we show in Sec. 5.1, the original approach for learning Linear Predictors as proposed by Jurie and Dhome [1] is very time consuming and not applicable for learning on the fly. Therefore, we propose a simple yet powerful way of learning Linear Predictors that is much faster than [1]. The main idea behind our new learning approach is to compress the image difference vectors $\delta\mathbf{i}_i$ before using them to learn the Linear Predictor matrix. By reducing the dimensionality of $\delta\mathbf{i}_i$ from n_p to n_r, the size of $\mathbf{H}\mathbf{H}^\top$ gets reduced to $n_r \times n_r$ and therefore, the necessary matrix inversion $(\mathbf{H}\mathbf{H}^\top)^{-1}$ becomes less computational expensive.

We propose to reduce the dimensionality of $\delta\mathbf{i}_i$ by using Discrete Cosine Transform (DCT). This transform is known to give good results for compressing image data by removing DCT coefficients that correspond to high frequencies. Keeping only low-frequency information makes it well-suited for template tracking, since high-frequency information tends to de-stabilize tracking. In the following, we first introduce the 2-dimensional DCT, then show how we apply it on the 1-dimensional vectors $\delta\mathbf{i}_i$ which are sampled from the 2-dimensional templates. Mathematically, the 2-dimensional DCT $\mathbf{U}$ of a $k \times k$ matrix $\mathbf{V}$ is:

$$\mathbf{U} = \mathrm{DCT}(\mathbf{V}) = \mathbf{C}\mathbf{V}\mathbf{C}^\top \tag{4}$$

where the elements of the matrix $\mathbf{C}$ are defined as:

$$\mathbf{C}_{i,j} = \sqrt{\frac{\alpha_i}{k}} \cos\left[\frac{\pi(2j+1)i}{2k}\right] \tag{5}$$

with

$$\alpha_i = \begin{cases} 1 & \text{if } i = 0, \\ 2 & \text{otherwise.} \end{cases} \tag{6}$$

After transforming $\delta\mathbf{i}_i$ as $\delta\hat{\mathbf{i}}_i = \mathrm{DCT}(\delta\mathbf{i}_i)$, we form $\hat{\mathbf{H}} = (\delta\hat{\mathbf{i}}_1, \delta\hat{\mathbf{i}}_2, \ldots, \delta\hat{\mathbf{i}}_{n_t})$. However, since we reshaped the samples from a 2D template into a vector, Eq. (4) can not be directly applied to $\delta\mathbf{i}_i$. Therefore, we create an $n_p \times n_p$ matrix $\mathbf{W}_{DCT}$ which maps the difference $\delta\mathbf{i}_i$ of the sampled vectors directly to their DCT counterparts $\delta\hat{\mathbf{i}}_i$. Assuming that the vector $\delta\mathbf{i}_i$ is reshaped from the 2D matrix $\mathbf{V}_i$ written as $\delta\mathbf{i}_i = \mathrm{reshape}(\mathbf{V}_i)$, we compute $\mathbf{W}_{DCT}$ as:

$$\mathbf{W}_{DCT} = \left(\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{n_p}\right) \tag{7}$$

where $\mathbf{b}_m = \mathrm{reshape}(\mathbf{C}\mathbf{B}_m\mathbf{C}^\top)$ and $\mathbf{B}_m$ is a matrix with all elements set to 0 except for the m-th element which is set to 1. By setting a single element to 1, the set of matrices $\{\mathbf{B}_1, \ldots, \mathbf{B}_{n_p}\}$ are a base of the image space of the template and the set of vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_{n_p}\}$ are the DCT projections of this base. This way, we can directly compute the 2-dimensional DCT of our image difference vectors as:

$$\delta\hat{\mathbf{i}}_i = \mathbf{W}_{DCT}\delta\mathbf{i}_i \quad \Rightarrow \quad \hat{\mathbf{H}} = \mathbf{W}_{DCT}\mathbf{H}. \tag{8}$$

In relation to the original learning formula in Eq. (3), we reformulate this by using the relation:

$$\mathbf{H} = (\mathbf{W}_{DCT})^{-1}\hat{\mathbf{H}}. \tag{9}$$

Thus, we subsitute $\mathbf{H}$ from Eq. (9) to Eq. (2) and solve for the Linear Predictor matrix $\mathbf{A}$ as follows:

$$\begin{aligned}
\mathbf{A}\mathbf{W}_{DCT}^{-1}\hat{\mathbf{H}} &= \mathbf{Y} \\
\mathbf{A}\mathbf{W}_{DCT}^{-1} &= \mathbf{Y}\hat{\mathbf{H}}^{\top}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}^{\top}\right)^{-1} \\
\mathbf{A}\mathbf{W}_{DCT}^{-1}\mathbf{W}_{DCT} &= \mathbf{Y}\hat{\mathbf{H}}^{\top}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}^{\top}\right)^{-1}\mathbf{W}_{DCT} \\
\mathbf{A} &= \mathbf{Y}\hat{\mathbf{H}}^{\top}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}^{\top}\right)^{-1}\mathbf{W}_{DCT}
\end{aligned} \tag{10}$$

To reduce the necessary computational load, we apply a dimensionality reduction by defining an $n_r \times n_p$ submatrix $\mathbf{W}_{DCT}^{(n_r)}$ with $n_r < n_p$, such that the necessary matrix inversion is no longer applied to an $n_p \times n_p$ matrix but rather to an $n_r \times n_r$ matrix. The final Linear Predictor is then computed as:

$$\mathbf{A}^{(n_r)} = \mathbf{Y}\hat{\mathbf{H}}^{(n_r)\top}\left(\hat{\mathbf{H}}^{(n_r)}\hat{\mathbf{H}}^{(n_r)\top}\right)^{-1}\mathbf{W}_{DCT}^{(n_r)}. \tag{11}$$

with $\hat{\mathbf{H}}^{(n_r)} = \mathbf{W}_{DCT}^{(n_r)}\mathbf{H}$. We show in Sec. 5.1 that by keeping n_r small, the learning time for large templates is significantly reduced. Moreover, depending on the size of n_r, the reduction in learning even increases tracking robustness.

5 Experiments

In this section, we evaluate our approach for efficient learning of Linear Predictors, as proposed in Sec. 4, by comparing it to the original learning approach proposed by Jurie and Dhome [1], the iterative approach of Holzer *et al.* [2] which is also referred to as Adaptive Linear Predictors (ALPs), as well as the approach of Benhimane *et al.* [11] known as Efficient Second-order Minimization (ESM). For the comparison, we use two kinds of evaluation – timing and tracking performance. The former shows the difference in learning and tracking times (see Sec. 5.1); while the latter involves the computation of tracking robustness with respect to different types of motion as well as its sensitivity to noise (see Sec. 5.2). Finally, we demonstrate the usefulness of fast template learning using tracking on mobile devices (see Sec. 5.3).

All algorithms used in this experiment are implemented in C++. For our approach, we consider three different instances where the difference is in the number of DCT coefficients used for learning. Specifically, we use varying DCT coefficients with values of 25, 49, and 81. The evaluation of Holzer *et al.* [2] is performed using binaries kindly provided by the authors while the approach of Benhimane *et al.* [11] is evaluated using publicly available binaries[1]. The

[1] See version 0.4 available at http://esm.gforge.inria.fr/ESM.html

Fig. 2. The data set used for synthetic experiments. These images are randomly taken from the Internet.

evaluations of these algorithms are conducted using a notebook with a 2.26GHz Intel(R) Core(TM)2 Quad CPU and 8 GB of RAM, where only one core is used for the computations. The images used for evaluation on synthetic data are taken from the Internet (see Fig. 2). For all synthetic experiments, the template size is 150×150 pixels. A template is located at the center of the image and tracking is applied on its warped versions.

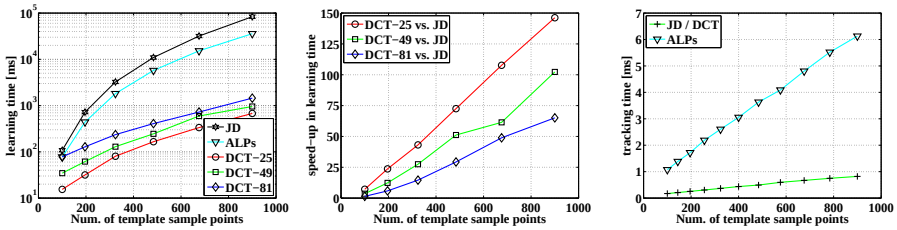
We want to emphasize that the reason for focusing on synthetic experiments in Sec. 5.1 and 5.2 is to generate a more accurate comparison using ground truth measurements. Using other methods of testing, such as using markers on real scenes to find the camera motion, generates its own error and limits the amount of motion available for testing. In addition, our evaluation also has the benefit of control which means that it is done by changing only variables that are being tested while keeping the others constant throughout the experiment. Furthermore, it allows to precisely specify the amount of change to fairly evaluate at which value the algorithm failed.

In addition to the synthetic evaluation, we also show several qualitative results from real video sequences in Sec. 5.3. These demonstrate the proposed approach used on a mobile phone for learning and tracking templates in unknown environments.

5.1 Computational Complexity

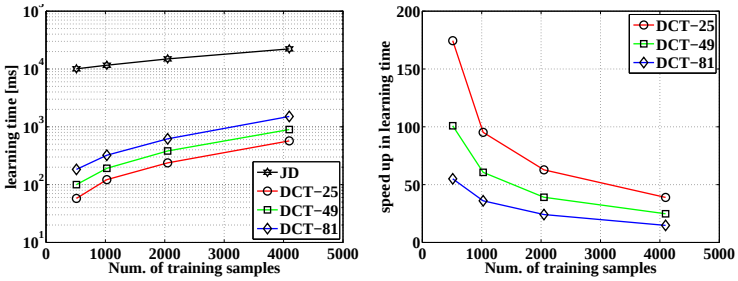
In the first evaluation, timing is measured by counting the amount of time to finish a specific part of the algorithm, *i.e.* learning and tracking. We compare the computational complexity of our approach with the approach of Jurie and Dhome [1] as well as Holzer *et al.* [2].

Learning. Our main contribution is reflected on the learning time. In Fig. 3 (a), we evaluate the amount of time necessary for learning with respect to the number of sample points. It shows that as the amount of sample points increases, the time required for learning using our approach increases much slower in comparison to the approach of both, Jurie and Dhome [1] and Holzer *et al.* [2]. As



(a) (b) (c)

Fig. 3. Comparison of timings for the approach proposed by Jurie and Dhome [1] ('JD'), the approach of Holzer *et al.* [2] ('ALPs'), and our approach ('DCT-x'). **(a)** Comparison of learning time. **(b)** Obtained speed-up of our approach with respect to Jurie and Dhome [1]. **(c)** Comparison of tracking time.



(a) (b)

Fig. 4. Evaluation of learning time depending on the number of training samples used for training. **(a)** Learning time of our approach ('DCT-x') in comparison to the approach of Jurie and Dhome [1] ('JD') and **(b)** the speed-up obtained by our approach with respect to the number of DCT coefficients used for training. The experiments were performed with a template-size of 150×150 pixels, where 22×22 sampling points were used.

expected, using less DCT coefficients for learning decreases the necessary time. This difference is emphasized in Fig. 3 (b) where it is evident that for templates with more than 800 sample points (*e.g.* 30×30), our approach is more than two orders of magnitude faster, *i.e.* almost 150 times faster, than Jurie and Dhome [1] if 25 DCT coefficients are used. Using 81 DCT coefficients, it is still approximately 70 times faster.

Fig. 4 compares the necessary learning time with the number of random samples used for training. Reducing the number of random samples drastically reduces the necessary time for learning Linear Predictors. Although this comes hand in hand with a decrease in tracking performance, we show in Sec. 5.2 that our approach is much more robust against this kind of reduction compared to the original approach of Jurie and Dhome [1].

Tracking. Both the original approach [1] and our approach share the same approach for tracking and therefore, have equal tracking times. Furthermore, the measure of tracking time per frame with respect to template size in Fig. 3(c) demonstrates that our approach can easily reach frame rates higher than 1000 fps even for templates with a high number of sample points. In contrast to Holzer *et al.* [2], their approach is slightly slower and the necessary time for tracking increases significantly faster as the template size increases. Considering a non-linear template tracking approach, the method of Benhimane *et al.* [11] takes about 10 ms for tracking the same templates.

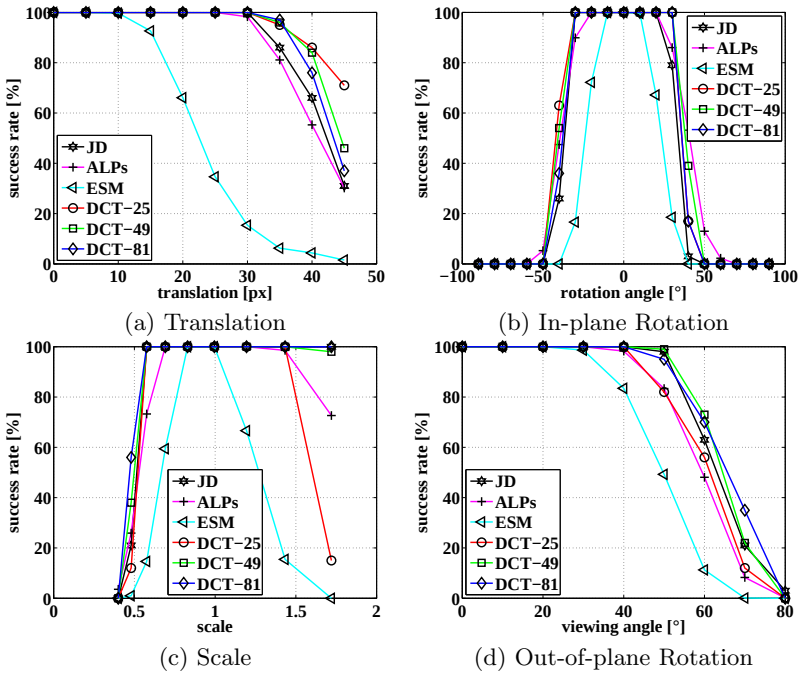
5.2 Robustness

In this section, we measure the tracking performance by finding the correct location of the template after inducing random transformations and noise to several test images. These random transformations include translation, rotation (in-plane rotation), scale and viewpoint changes (out-of-plane rotation). For the experiments on the influence of noise, we corrupted the images with noise sampled from a Gaussian distribution before applying the random transformation.

Applying these disturbances to the test images, tracking is considered successful if the mean pixel distance between the reference template corner points and the tracked template corner points, which are back-projected into the reference view, is less than 5 pixels. Hence, robustness is measured as the percentage of successfully tracked templates after applying several random disturbances to each test image.

Number of sample points. In Fig. 5, we compare the tracking robustness using different types of transformations in relation to the number of sample points. Here, we evaluate our approach with different numbers of DCT coefficients as well as the approach of Jurie and Dhome [1], Holzer *et al.* [2], and the non-linear approach of Benhimane *et al.* [11]. Hereby, the training stage as it is applied for the methods based on Linear Predictors leads to significantly better results than obtained using the non-linear approach of Benhimane *et al.* [11]. Comparing our approach with that of Jurie and Dhome [1] reveals that our approach is always better or comparable, except for the variant where we use only 25 DCT coefficients. This gives slightly worse results for large changes in viewing angle and scale. The approach of Holzer *et al.* [2] tends to give slightly worse results than that of Jurie and Dhome [1]. The improvement in tracking robustness using our approach can be explained by the fact that only low-frequency data is kept during the compression using the DCT and high-frequency data of the template is removed. As a result, noise and fine details, which tend to de-stabilize tracking, are removed.

Having a look at the tracking performance when varying the number of random transformations used for training (see Fig. 6), we see that the approach of Jurie and Dhome [1] lacks robustness when reducing the number of training samples while our approach still keeps high tracking performance even with re-



(a) Translation (b) In-plane Rotation

(c) Scale (d) Out-of-plane Rotation

Fig. 5. Comparison of tracking performance for the approaches proposed by Jurie and Dhome [1] ('JD'), Holzer *et al.* [2] ('ALPs'), Benhimane *et al.* [11] ('ESM'), and our approach ('DCT-x'). Four different types of motions are considered: **(a)** translation, **(b)** in-plane rotation, **(c)** scale and **(d)** out-of-plane rotation. The experiments were performed with a template size of 150×150 pixels, where 20×20 sampling points are used for JD, ALPs and our approach. ESM uses the complete template. For training we used 1200 training samples.

duced training examples. This property is useful to even further decrease the learning time if necessary, as we showed in Fig. 4.

Sensitivity to Noise. The results presented in Fig. 7 compare our proposed approach with Jurie and Dhome [1] with respect to sensitivity to noise, where the noise parameter specifies the standard deviation of the Gaussian noise and is with respect to an image value range from 0 to 255. Fig. 7 (a) shows that increasing the number of used DCT coefficients also increases the robustness against noise. Using 81 DCT coefficients, we obtain a template tracking approach which is more robust against noise than the one of Jurie and Dhome [1]. Looking at Fig. 7 (b), we see that our approach, in general, gives a smaller mean error in the tracking results.

It is noteworthy that being less sensitive to noise is an advantage in environments with bad lighting conditions, *e.g.* at night when the signal-to-noise ratio of cameras usually decreases. This is especially the case for cameras used in mobile devices.

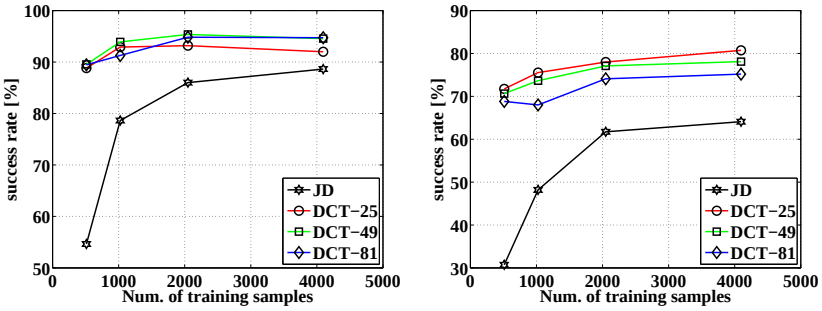


Fig. 6. Evaluation of tracking success rate with respect to the number of training samples. The left graph shows success rates for random translations in the range of 30 to 40 pixels while the right one shows them for random translations in the range of 35 to 45 pixels. For these experiments we used templates with 22×22 sample points.

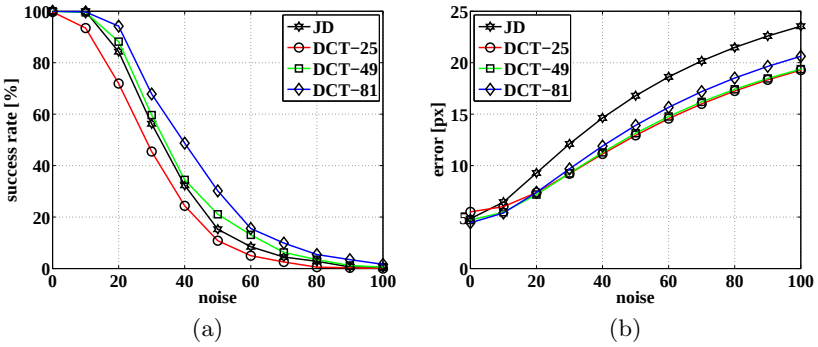


Fig. 7. Comparison of sensitivity to noise for the approach proposed by Jurie and Dhome [1] and our approach.

5.3 Exemplary Applications on Mobile Phones

To demonstrate the efficiency of the template learning and tracking, we implemented it on a standard mobile phone with a 1.2 GHz dual core processor with 1 GB of RAM. Note that we did not optimize the implementation for processor specific technology and used only a single core for the learning and tracking.

Tracking of a Single Template In Fig. 8, we show exemplary images demonstrating the tracking of a single template on a mobile phone. Learning times for a single template are approximately 18000 ms for the original approach of Jurie and Dhome [1] and about 600 ms for our proposed approach. To estimate the learning time, we trained a template with 16×16 sample points, $16 \cdot 16 \cdot 3 = 768$ training samples, and used 25 DCT coefficients in our approach. As a result, the tracking takes about 2.5 ms for both approaches.

Fig. 8. Tracking of a single template on a mobile phone.

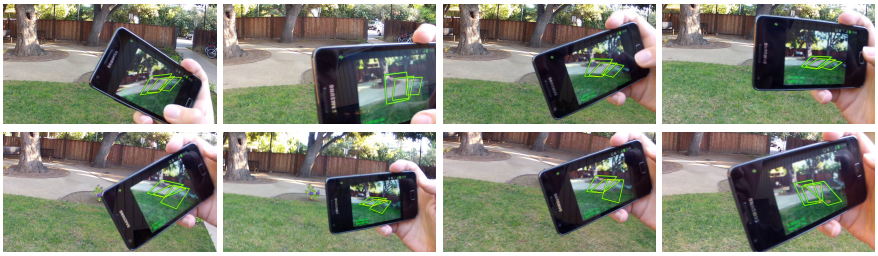


Fig. 9. Tracking of multiple templates on a mobile phone. The most right template shown in the first row failed during tracking and was replaced by a new template in the second row.

Tracking of Multiple Templates Fig. 9 shows the tracking of multiple templates. This can be useful for applications like SLAM or similar systems where a patch-based reconstruction of a scene is performed.

6 Conclusion

We proposed an efficient method for learning Linear Predictors for real-time template tracking by making use of the Discrete Cosine Transform for dimensionality reduction. This reduces the necessary computation dramatically and enables to learn Linear Predictors at run-time. We demonstrated that the introduced learning procedure leads to an improvement in handling of large motions and image noise, and showed its usefulness for mobile applications.

References

1. Jurie, F., Dhome, M.: Hyperplane approximation for template matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (2002)
2. Holzer, S., Ilic, S., Navab, N.: Adaptive linear predictors for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA (2010)
3. Lucas, B., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: International Joint Conference on Artificial Intelligence. (1981)
4. Shum, H.Y., Szeliski, R.: Construction of panoramic image mosaics with global and local alignment. International Journal of Computer Vision (2000)
5. Hager, G., Belhumeur, P.: Efficient region tracking with parametric models of geometry and illumination. IEEE Transactions on Pattern Analysis and Machine Intelligence (1998)

6. Cascia, M., Sclaroff, S., Athitsos, V.: Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2000)
7. Dellaert, F., Collins, R.: Fast image-based tracking by selective pixel integration. In: ICCV Workshop of Frame-Rate Vision. (1999)
8. Baker, S., Matthews, I.: Equivalence and efficiency of image alignment algorithms. In: Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA (2001)
9. Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision (2004)
10. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: IEEE International Conference on Robotics and Automation. (2004)
11. Benhimane, S., Malis, E.: Homography-based 2d visual tracking and servoing. International Journal of Robotics Research (2007)
12. Jurie, F., Dhome, M.: Real time robust template matching. In: British Machine Vision Conference. (2002)
13. Gräßl, C., Zinßer, T., Niemann, H.: Efficient hyperplane tracking by intelligent region selection. In: Image Analysis and Interpretation. (2004)
14. Parisot, P., Thiesse, B., Charvillat, V.: Selection of reliable features subsets for appearance-based tracking. Signal-Image Technologies and Internet-Based System (2007)
15. Matas, J., Zimmermann, K., Svoboda, T., Hilton, A.: Learning efficient linear predictors for motion estimation. In: Computer Vision, Graphics and Image Processing. (2006)
16. Mayol, W.W., Murray, D.W.: Tracking with general regression. Journal of Machine Vision and Applications (2008)
17. Zimmermann, K., Matas, J., Svoboda, T.: Tracking by an optimal sequence of linear predictors. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)
18. Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., Lepetit, V.: Online learning of patch perspective rectification for efficient object detection. In: Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska (2008)
19. Hinterstoisser, S., Kutter, O., Navab, N., Fua, P., Lepetit, V.: Real-time learning of accurate patch rectification. (2009)
20. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of texture-less objects. (2010)
21. Holzer, S., Hinterstoisser, S., Ilic, S., Navab, N.: Distance transform templates for object detection and pose estimation. (2009)
22. Özuysal, M., Fua, P., Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. In: Conference on Computer Vision and Pattern Recognition, Minneapolis, MI, USA (2007)
23. Dame, A., Marchand, E.: Accurate real-time tracking using mutual information. In: Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on. (2010) 47 –56
24. Richa, R., Sznitman, R., Taylor, R., Hager, G.: Visual tracking using the sum of conditional variance. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. (2011) 2953 –2958
25. Gräßl, C., Zinßer, T., Niemann, H.: Illumination insensitive template matching with hyperplanes. In: Proceedings of Pattern recognition: 25th DAGM Symposium, Magdeburg, Germany (2003)