Online Learning of Linear Predictors for Real-Time Tracking

Stefan Holzer[1], Marc Pollefeys[2], Slobodan Ilic[1], David Tan[1], and Nassir Navab[1]

[1]Department of Computer Science, Technische Universität München (TUM), Boltzmannstrasse 3, 85748 Garching, Germany
{holzers,slobodan.ilic,tanda,navab}@in.tum.de
[2]Department of Computer Science, ETH Zurich, CNB G105, Universitatstrasse 6, CH-8092 Zurich, Switzerland
marc.pollefeys@inf.ethz.ch

Abstract. Although fast and reliable, real-time template tracking using linear predictors requires a long training time. The lack of the ability to learn new templates online prevents their use in applications that require fast learning. This especially holds for applications where the scene is not known a priori and multiple templates have to be added online. So far, linear predictors had to be either learned offline [1] or in an iterative manner by starting with a small sized template and growing it over time [2]. In this paper, we propose a fast and simple reformulation of the learning procedure that allows learning new linear predictors online.

Key words: template tracking, template learning, linear predictors

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 by estimating 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.

Most approaches to template tracking are based on energy minimization [3–11], where the image intensity differences between template areas of two consecutive frames have to be minimized in terms of the template warping parameters. In many cases, analytical derivation of the Jacobian is used in order to provide real-time tracking capabilities. Alternative approaches to template tracking are based on learning [1, 2, 12–17], where the relation between image intensity differences and template warping parameters is learned. While energy minimization approaches are flexible at run-time, learning based methods have proven to allow much faster tracking.

A very successful learning based template tracker was proposed by Jurie and Dhome [1]. It is based on learning linear predictors to efficiently compute template warp parameter updates. Thanks to extensive training, this approach is very fast and tends to avoid local minima. The costly learning phase, however, prohibits this method from computing templates online.

In many applications, such as simultaneous localization and mapping (SLAM), the ability to learn new templates at run-time is crucial, since they have to deal with data, which is not available for prior offline learning. Contrary to the current development of methods for highly parallelized systems, which *e.g.* rely on modern graphics cards, most consumer-oriented applications, especially those placed on mobile devices, do not have such a huge processing power available.

We, therefore, propose a reformulation of the linear predictor learning step that drastically improves the learning speed. Although this way of training brings a small decrease in tracking robustness, it helps to improve robustness against image noise. However, we demonstrate how the tracking robustness can be increased online during tracking and how the tracking performance of the original approach of Jurie and Dhome [1] can be achieved.

2 Related Work

Since the seminal work of Lucas and Kanade [3], a large variety of template tracking approaches have been presented. They can be classified into three main categories: tracking-by-detection (TBD) [18–22], energy minimization [3–11] and learning [1, 2, 12–17]. In contrast to others, TBD-based approaches track a template over the whole image independent from the previous position. Nonetheless, they often require a time consuming training procedure and can hardly achieve the processing speed of frame-to-frame tracking. Furthermore, their possible pose space is limited.

On the other hand, the latter two categories use frame-to-frame tracking. Between them, energy minimization-based approaches are generally more flexible at run-time while learning-based approaches enable higher tracking speed. Additionally, Jurie *et al.* [12] demonstrated that Linear Predictors (LPs) are superior to Jacobian approximation and Holzer *et al.* [2] showed an experiment where LPs are superior to Efficient Second-order Minimization (ESM) [11].

Tracking-by-Detection-based approaches. Özuysal *et al.* [18] presented a TBD-based approach called FERNs where they extract keypoints from an image and match them using a classification-based approach by estimating the probability on which class the keypoints belong. However, it needs a time consuming learning stage and requires a sufficient number of visible keypoints which makes it less useful in tracking small regions. Another approach is DTTs from Holzer *et al.* [19] which builds on finding closed contours and matches them using a similar approach as the keypoint matching in [18]. Thus, this also needs a time consuming learning stage while detection speed was reported at only 10 fps. Furthermore, prominent advances in this field are reflected from the works of Hinterstoisser *et*

al. [20–22]. Their earlier works called Leopard [20] and Gepard [21] make use of the image patch that surrounds a keypoint. These patches are then used for matching and pose estimation. Hence, their methodology suggests that they benefit from any advancement in template tracking. Moreover, although they achieve a near real time performance, these approaches heavily rely on the repeatability of the underlying keypoint detector. In their recent work, DOT [22] aims to overcome the dependency on keypoint detection. It is a template matching based approach that learns templates for every pose. Due to this, it restricts the application space and is comparably slow in contrast to frame-to-frame tracking.

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 [3–8], different orders of approximation of the error function [10, 11], and occlusion and illumination change handling [5]. 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]. Among these types, the interesting component in the inverse additive and inverse compositional approach is that it switches the functions of the reference and current image. As a consequence, it is possible to transfer some of the computation to the initialization phase and to make the tracking computationally more efficient. Faster convergence rates for larger convergence areas can be additionally obtained by using a second-order instead of a first-order approximation of the error function [10, 11]. Lastly, Hager and Belhumeur [5] established a method to compensate for illumination changes and occlusions. A more detailed overview of energy-based tracking methods is given by Baker and Matthews [9].

Learning-based approaches. In contrast to energy minimization 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. This simplifies the tracking process from the previous approach by using a matrix vector multiplication. Here, the "Jacobians" are computed once for the whole method. Furthermore, the same authors extended their approach to handle occlusions [12]. Other authors such as Gräßl *et al*. [23] demonstrated how linear predictors can be made invariant to illumination changes. In addition, to further increase accuracy in tracking, they [13] also formulated a method on how to select the points for sampling from the image data. 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 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 Background and Terminology

This section aims to introduce our notations and to summarize the fundamental aspects of the template tracking approach proposed by Jurie and Dhome [1], which is used in comparison to our approach as introduced in Sec. 4.

3.1 Template and Parameter Description

Without loss of generality, we define a template as a rectangular region in the first frame of a video sequence, which defines the region of interest that we want to track. Note that the method is not limited to rectangular regions and is capable of dealing with arbitrary shapes. The location of the region within the image is defined by the variable $\boldsymbol{\mu}$. In this paper, $\boldsymbol{\mu}$ is an 8×1 vector that stores the position of the four 2D corner points of the template region in the image. Thus, $\boldsymbol{\mu}$ has to be estimated in every new frame. Furthermore, to find the image intensities in the template, n_p sample points are positioned on a regular grid instead of using all the pixels in the template region. These intensities are stored in an $n_p \times 1$ vector $\mathbf{i}$, where $\mathbf{i} = \left(i_1, i_2, \ldots, i_{n_p}\right)^\top$.

3.2 Template Tracking based on Linear Predictors

Given a template region in a reference image, the corresponding initial parameter values and reference image intensities are stored in $\boldsymbol{\mu}_R$ and $\mathbf{i}_R$, respectively. The template parameter values $\boldsymbol{\mu}_C$ define the location of the template in the current image; henceforth, tracking is done by computing $\boldsymbol{\mu}_C$. The value of $\boldsymbol{\mu}_C$ depends on the previously computed parameter values $\boldsymbol{\mu}_{C-1}$, the image intensities in the reference image $\mathbf{i}_R$ and the image intensities in the current image $\mathbf{i}_C$. Jurie and Dhome [1] simplified this relation as:

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

where $\delta\boldsymbol{\mu}$ are the template parameter updates, $\delta\mathbf{i} = \mathbf{i}_C - \mathbf{i}_R$ and $\mathbf{A}$ is a linear predictor matrix. It is important to mention that the image intensities $\mathbf{i}_C$ are extracted from the current image using the sample points from the previously computed parameter values $\boldsymbol{\mu}_{C-1}$. Therefore, in order to compute the update of the template parameters $\delta\boldsymbol{\mu}$, one needs to pre-compute the matrix $\mathbf{A}$. In this case, $\mathbf{A}$ is called a linear predictor since it establishes a linear relation between the image differences and the parameter updates.

$\mathbf{A}$ is a constant matrix of size $8 \times n_p$, which is computed during the learning phase. The learning process uses n_t random transformations on the reference

template, where n_t is much larger than n_p. These transformations are small disturbances $\delta\boldsymbol{\mu}_i$, $i = 1, \ldots, n_t$, to the reference parameters $\boldsymbol{\mu}_R$. As a consequence, this introduces a change in the image intensities $\delta\mathbf{i}_i = \mathbf{i}_i - \mathbf{i}_R$ for each random transformation. The vectors of those small disturbances to the template position parameters are concatenated into an $8 \times n_t$ matrix $\mathbf{Y}$, while the corresponding image intensity differences are stored in an $n_p \times n_t$ matrix $\mathbf{H}$. These can be written as $\mathbf{Y} = \left(\delta\boldsymbol{\mu}_1, \delta\boldsymbol{\mu}_2, \ldots, \delta\boldsymbol{\mu}_{n_t}\right)$ and $\mathbf{H} = (\delta\mathbf{i}_1, \delta\mathbf{i}_2, \ldots, \delta\mathbf{i}_{n_t})$. Using $\mathbf{Y}$ and $\mathbf{H}$, Eq. (1) is modified and becomes:

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

Finally, $\mathbf{A}$ is learned by minimizing:

$$\arg\min_{\mathbf{A}} \sum_{k=1}^{n_t} \left(\delta\boldsymbol{\mu}_k - \mathbf{A}\delta\mathbf{i}_k\right)^2 \tag{3}$$

which results in the closed-form solution:

$$\mathbf{A} = \mathbf{Y}\mathbf{H}^\top \left(\mathbf{H}\mathbf{H}^\top\right)^{-1}. \tag{4}$$

This leads to an inverse-compositional tracking approach, where the parameter updates, obtained from Eq. (1), have to be applied to the reference parameters $\boldsymbol{\mu}_R$ and a corresponding transformation has to be estimated. The inverse of this transformation is then used to update the current template parameters. In our implementation, we compute a homography to represent the current perspective distortion.

To improve invariance to illumination changes, normalization is used on the extracted image data by imposing zero mean and unit standard deviation. As a consequence, zero mean makes $\mathbf{H}$ lose one rank and the resulting $\mathbf{H}\mathbf{H}^\top$ rank-deficient. In order to prevent this rank-deficiency, random noise is added to $\mathbf{H}$ after normalization.

3.3 Multi-Layered Tracking

In order to make tracking more robust, we use a multi-predictor approach where we compute n_l linear predictors: $\mathbf{A}_1, \ldots, \mathbf{A}_{n_l}$. Among the linear predictors, $\mathbf{A}_1$ has learned large template distortions, while $\mathbf{A}_{n_l}$ has learned smaller parameter changes. Intuitively, $\mathbf{A}_1$ accounts for large movements of the template and the subsequent linear predictors further refine the results of the previous predictor. In practice, each linear predictor is utilized several times before the next level is used. In this paper, we used $n_l = 5$ and three iterations for each predictor.

4 Fast Learning Strategy

Considering Eq. (4), it is evident that the computation of the linear predictor $\mathbf{A}$ using Jurie and Dhome [1] is time-consuming due to the pseudo-inverse of $\mathbf{H}$. This involves the inverse of an $n_p \times n_p$ matrix $\mathbf{H}\mathbf{H}^\top$.

To increase the speed, we propose to use the pseudo-inverse of $\mathbf{Y}$, instead of $\mathbf{H}$, in order to generate a much faster learning process. Using this approach, Eq. (2) leads to:

$$\mathbf{I} = \mathbf{A}\mathbf{H}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1} = \mathbf{A}\mathbf{B}, \tag{5}$$

where $\mathbf{B} = \mathbf{H}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1}$ is an $n_p \times 8$ matrix; henceforth, to learn $\mathbf{A}$, we compute:

$$\mathbf{A} = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top. \tag{6}$$

The pseudo-inverse is applied differently in Eqs. (5) and (6), since for matrix $\mathbf{Y}$, the rows are linearly independent while for matrix $\mathbf{B}$, the columns are linearly independent; and therefore, computing it the same way leads to a rank-deficient inversion in one of the two cases [24].

It is noteworthy to mention that the computation of the matrix $\mathbf{A}$ involves two matrix inverse, but both $\mathbf{Y}\mathbf{Y}^\top$ and $\mathbf{B}^\top\mathbf{B}$ are 8×8 matrices. However, computing the inverse of two 8×8 matrices is much faster in comparison to computing the inverse of an $n_p \times n_p$ matrix. In fact, $\mathbf{Y}\mathbf{Y}^\top$ can be precomputed. Therefore, only a single 8×8 matrix has to be inverted online.

Since the linear mapping denoted by the linear predictor should never encode fixed offsets, we normalize $\mathbf{Y}$ such that each parameter has zero mean and unit standard deviation; while de-normalizing $\delta\boldsymbol{\mu}$ when solving Eq. (1) in tracking. It is interesting to note that unlike the normalization used in Sec. 3.2 to obtain invariance on changes in lighting conditions, this normalization does not generate a rank-deficient matrix $\mathbf{Y}\mathbf{Y}^\top$ because the normalization is applied on the rows of $\mathbf{Y}$. The difference in performance using normalized and unnormalized $\mathbf{Y}$ is shown in Sec. 5.

Moreover, solving Eq. (4) in the approach of Jurie and Dhome [1] actually corresponds to approximating $\mathbf{Y}$ by orthogonally projecting it on $\mathbf{H}$. On the other hand, solving Eq. (6) in our approach approximates $\mathbf{H}$ by orthogonally projecting it on $\mathbf{Y}$. Given that we project $\mathbf{H}$ on $\mathbf{Y}$, all noise outside of the low-rank space represented by $\mathbf{Y}$ has no effect; while in case of Jurie and Dhome, the noise has more effect. This makes their approach more sensitive to noise. We also prove this in our experiments.

Updating Linear Predictors. Hinterstoisser *et al.* [20] showed that new training samples can be added to a linear predictor even after learning by making use of the Sherman-Morrison formula. It relies on the original way of computing linear predictors as $\mathbf{A} = \mathbf{Y}\mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)^{-1}$ and efficiently updates the inverse $\mathbf{S} = (\mathbf{H}\mathbf{H}^\top)^{-1}$. In contrast to this, our method does not compute for $\mathbf{S}$ in the learning phase as Jurie and Dhome [1] do. We propose to derive $\mathbf{S}$ from an existing linear predictor $\mathbf{A}$ using Eq. (4):

$$\mathbf{A} = \mathbf{Y}\mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)^{-1} = \mathbf{Y}\mathbf{H}^\top\mathbf{S} = \mathbf{D}\mathbf{S}, \tag{7}$$

where $\mathbf{D} = \mathbf{Y}\mathbf{H}^\top$ is an $8 \times n_t$ matrix. From this, $\mathbf{S}$ can be computed using the pseudo-inverse of $\mathbf{D}$:

$$\mathbf{S} = \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{A}. \tag{8}$$

In this computation, we are using the matrix inverse of $\mathbf{D}\mathbf{D}^\top$. Again, this is an 8×8 matrix and can be inverted very fast.

Sec. 5 shows that updating $\mathbf{S}$ by adding training samples using the Sherman-Morrison formula helps to further improve the tracking performance. The update is done by:

$$\hat{\mathbf{S}} = \left(\mathbf{S}^{-1} + \delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^\top\right)^{-1} = \mathbf{S} - \frac{\mathbf{S}\delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^\top\mathbf{S}_I}{1 + \delta\mathbf{i}_{n_t+1}^\top\mathbf{S}\delta\mathbf{i}_{n_t+1}}, \tag{9}$$

where $\delta\mathbf{i}_{n_t+1}$ is a vector of image value differences obtained from a new random transformation applied to the sample points. Note that before computing the updated linear predictor using Eq. (7), we also have to update the matrices $\mathbf{H}$ and $\mathbf{Y}$ by concatenating them with the new training samples. For the normalization of the parameter differences, we use the normalization as applied to the original learning.

5 Experiments

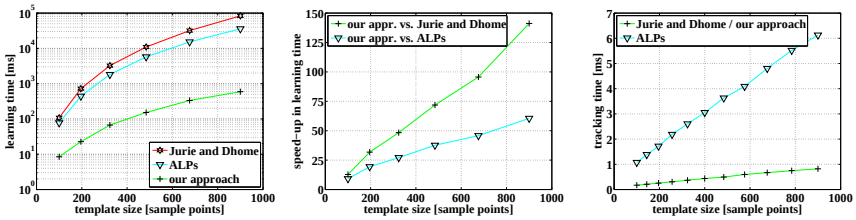
In this section, we evaluate our proposed approach for efficient learning of linear predictors by comparing it to the original learning approach proposed by Jurie and Dhome [1] and the iterative approach of Holzer *et al.* [2]. These comparisons are done using two kinds of evaluation – timing and accuracy. The former shows the difference in learning and tracking times; while the latter involves the computation of tracking robustness with respect to different types of motion as well as its sensitivity to noise. Additionally, we also compare the accuracy of our approach to the non-linear method of Benhimane *et al.* [11]. Moreover, we show several qualitative results from real video sequences. They show the algorithm used on a mobile phone for learning and tracking a single template as well as handling multiple templates. This demonstrates the need for fast learning in unknown environments.

All the algorithms are implemented in C++. For the implementation of Holzer *et al.* [2], we used the binaries provided by the authors; while the implementation of Benhimane *et al.* [11] is from the publicly available binaries[1]. The evaluation of these algorithms are conducted using a notebook with a 2.26 GHz Intel(R) Core(TM)2 Quad CPU and 4 GB of RAM, where only one core is used for the computation.

5.1 Computational Complexity

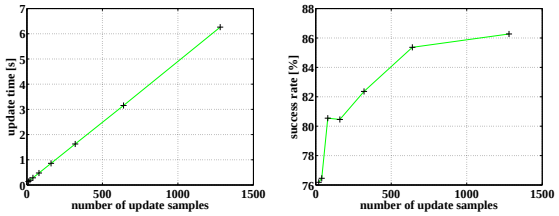
In the first evaluation we investigate the computational complexity of our approach in contrast to the approach of Jurie and Dhome [1] and Holzer *et al.* [2]. Our algorithm is divided into three parts – learning linear predictors, tracking using the learned linear predictors and updating the linear predictors while tracking. This section mainly focuses on the amount of time that each part requires to finish in relation to the number of sample points used.

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



(a) (b) (c)

Fig. 1. (a) Comparison of the necessary learning time with respect to the number of sample points used within the template for the approach proposed by Jurie and Dhome [1], by Holzer *et al.* [2] (referred as "ALPs") and our approach. **(b)** The corresponding speed-up in learning obtained by our approach. **(c)** The tracking time per frame with respect to the number of sample points used for the template.



(a) (b)

Fig. 2. (a) The time necessary to update an existing tracker with respect to number of update samples. This update can be performed in parallel with tracking. **(b)** The corresponding improvement in success rates with increasing number of updates.

Learning. Our main contribution is reflected on the learning time. We show in Fig. 1 (a) 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]. This difference is emphasized in Fig. 1 (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 120 times faster, than Jurie and Dhome [1] and more than 50 times faster than the approach of Holzer *et al.* [2].

Tracking. Both the original approach [1] and our approach have similar tracking time because the time needed to de-normalize the parameter updates in our approach is negligible. Furthermore, the measure of tracking time per frame with respect to template size in Fig. 1 (c) demonstrates that our approach can easily reach frame rates higher than 1000 fps even with large templates. In contrast to Holzer *et al.* [2], their method is slightly slower and the necessary time for tracking increases faster as the template size increases. In comparison to this, the non-linear approach of Benhimane *et al.* [11] takes about 10 ms for tracking the same template.

Updating. The updating process is a way of adding new training samples to a learned linear predictor during tracking. Fig. 2 (a) shows the time necessary to update an existing tracker with respect to the number of update samples, where the number of update samples corresponds to the number of random transformations applied to the template. Note that this is the same template as used for the initial learning. This result illustrates that by adding a small number of training samples at each time, we can keep the computational cost low while improving the performance of the tracker over time. In Fig. 2 (b), we show an exemplary improvement of tracking robustness when updating the linear predictors with a specific number of update samples. Sec. 5.2 discusses more on the tracking robustness in updating.

5.2 Robustness

In this section, we analyze the influence of our learning approach on the robustness of tracking with respect to different movements and different levels of noise. We measure accuracy by finding the correct location of the template after inducing random transforms to several test images. The images used in the evaluation are taken from the Internet (see supplementary material[2]). Moreover, the random transforms include translation, rotation, scale and viewpoint change. Using the test images and random transforms, tracking is considered successful if the mean pixel distance between the reference template corner points and the tracked template corner points, that is back-projected into the reference view, is less than 5 pixels. Hence, robustness is measured as the percent of successfully tracked templates after applying several random transforms to each test image. For measuring the robustness in relation to noise we corrupted the image with noise sampled from a Gaussian distribution before applying the random transform.

At this point, it is important to mention that the goal of this type of evaluation is to generate more accurate comparison with the ground truth measurements. Indeed, there are other methods of testing such as using markers on real scenes to find the camera motion. However, this approach includes markers that generate its own error and limit 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. We can also specify the amount of change to fairly evaluate at which value the algorithm failed.

Here, we compare our approach to the methods of Jurie and Dhome [1], Holzer *et al.* [2], and Benhimane *et al.* [11]. For our approach, we considered three different cases:

- **Unnormalized:** we do not normalize the parameter differences;
- **Normalized:** we normalize the parameter differences before learning the linear predictors and de-normalize them during tracking; and,

[2] The supplementary material, which includes the images used for evaluation as well as videos, can be found at `http://campar.in.tum.de/Main/StefanHolzer`.

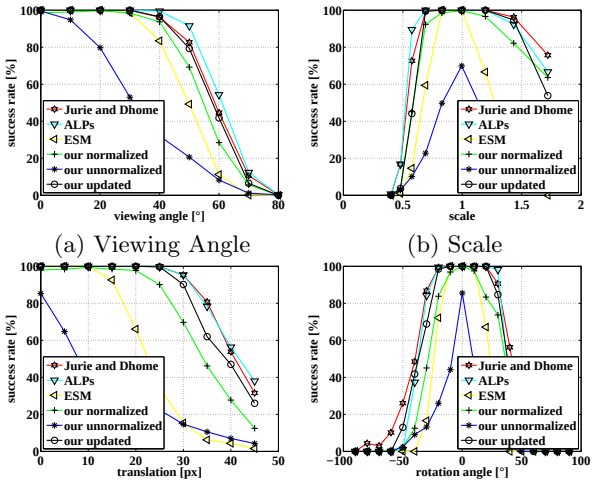
– **Updated:** we normalize the parameter differences and update the linear predictors with 1000 training samples before performing the experiments.

Given the learned linear predictor of a test image, the experiment starts by applying a random transform to the image and use the linear predictor to track this movement. This transform includes translation, rotation, scale and viewing angle. Therefore, after imposing several random transforms to a set of images, robustness is measured as the percent of successful estimation of the applied motions. For the evaluation with respect to noise, we corrupted the test image with Gaussian noise before applying the random transforms. Unless otherwise stated, the experiments are applied on templates of size 150×150 pixels with 18×18 sample points, and the initial learning of the linear predictors use $3 \cdot 18 \cdot 18 = 972$ training samples; while for the non-linear approach of Benhimane *et al.* [11], we use the complete template without subsampling.

Normalization of parameter differences. As we mentioned in Sec. 4, normalizing the parameter difference matrix $\mathbf{Y}$ before learning is important for our approach. To emphasize this, we included the results of the unnormalized approach in Fig. 3. It clearly shows that the unnormalized approach is not suitable for tracking. In contrast to that, the normalized approach gives results which are close to the original approach while the updated approach gets even closer to the results of Jurie and Dhome [1]. On the other hand, the outcome from Holzer *et al.* [2] also shows that it performs similarly well as Jurie and Dhome [1]. All the learning based approaches, except for the unnormalized version of our approach, give superior results compared to the non-linear approach of Benhimane *et al.* [11].

Number of samples points. In Fig. 4, we compare the tracking robustness in relation to the number of sample points. It is important to note that all the results show similar behavior across different transformations. Our normalized approach replicates the results of Jurie and Dhome [1] when the number of sample points per template is above 325. In all the results, the updated approach does not lose tracking robustness and performs consistently equal to the original approach.

Updating. Fig. 2 (b) depicts the change in robustness when we add new training samples to a learned linear predictor with normalization during tracking. In this experiment, we applied several random translations of approximately 30 pixels on the set of test images. After applying the updated linear predictors to the transformed images, we checked how often the translation was correctly estimated. This was done for linear predictors updated with different numbers of update samples, where the number of update samples is the number of random transformations applied to the template. The results show that tracking robustness increases as the number of update samples increases. We also show in Fig. 3 that updating brings the tracking performance closer to the original learning approach of Jurie and Dhome [1].



(a) Viewing Angle (b) Scale

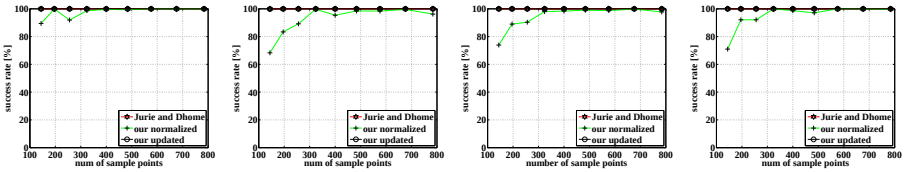
(c) Translation (d) In-plane Rotation

Fig. 3. Comparison of the approach of Jurie and Dhome [1], Holzer *et al.* [2] (referred as "ALPs"), Benhimane *et al.* [11] (referred as "ESM"), as well as our approach with and without normalization, and with updated predictors. We consider four different types of motions as specified. The success rate indicates the percent of successful estimation of the applied motions.

Sensitivity to Noise. A comparison among the different methods with respect to noise sensitivity is presented in Fig. 5. This experiment corrupts the input image by Gaussian noise with zero mean and varying standard deviation. After that, we impose a small translation to the corrupted image and measure the accuracy of the tracker for each algorithm. The noise parameter in Fig. 5 corresponds to the standard deviation of the Gaussian noise and the image intensity of the uncorrupted image ranges from 0 to 255.

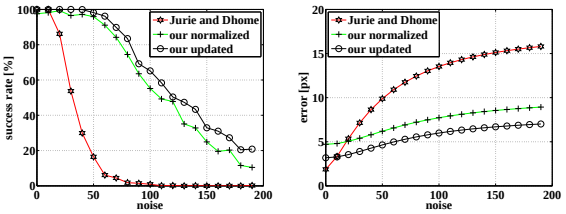
While our approach had a slightly worse tracking robustness for large motions as shown in Fig. 3, we illustrate in Fig. 5 (a) that the tracking robustness of our approach outperforms the original approach in terms of sensitivity to noise. An evidence for this is shown in Fig. 5 (b) where we analyze the average distance between the reference template corner points and the predicted template corner points that is back-projected into the reference view. Based on the figure, the prediction error of Jurie and Dhome [1] is smaller compared to our approach for small noise levels, but rapidly increases as the level of noise increases. Contrary to this, our approach has a higher error if no or only a small amount of noise is present, but the error increases slower when more noise is added. For a discussion on why our approach is less sensitive to noise in comparison to Jurie and Dhome [1], refer to Sec. 4.

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.



(a) Viewing Angle (b) Scale (c) Translation (d) Rotation

Fig. 4. Comparison of the success rate in tracking with respect to the number of sample points for the approach of Jurie and Dhome [1], as well as our approach with normalization and with updated predictors.



(a) (b)

Fig. 5. Comparison of our approach to the approach of Jurie and Dhome [1] with respect to sensitivity to noise in tracking. **(a)** shows the success rate for different noise levels. **(b)** shows the average error in the predicted corner points of the template.

5.3 Application: Tracking on a Mobile Phone

Due to the high efficiency of learning and tracking using the proposed approach, it is optimally suited for applications running on mobile devices. In order to demonstrate this, we implemented it on a mobile phone with a 1.2 GHz dual core processor and 1 GB of RAM. Note that we only used a single core for learning and tracking, and that we directly used our implementation without optimizing it for the special processor technology used in mobile phones. Sample images of tracking using a mobile phone are shown in Fig. 6, and a video that demonstrates the learning and tracking can be found in the supplementary material.

Exemplary learning times for [1] are approximately 18000 ms for a template with 16×16 sample points, whereas our approach needs only approximately 350 ms. Therefore, our approach is more than 50 times faster than the approach of Jurie and Dhome [1], but more importantly, allows interactive applications to start tracking almost immediately. For tracking, both approaches need about 2.5 ms per frame for a template with 16×16 sample points.

5.4 Application: Tracking of Multiple Templates Simultaneously

In Fig. 7, we demonstrate the simultaneous tracking of multiple templates. The first row in this figure shows the tracking of three templates on a mobile phone while the second row demonstrates tracking of a large number of templates and shows that the use of multiple templates helps to handle occlusions. Because of

Fig. 6. Tracking on a mobile phone. The upper row shows tracking a non planar surface while the lower row shows tracking a planar scene.

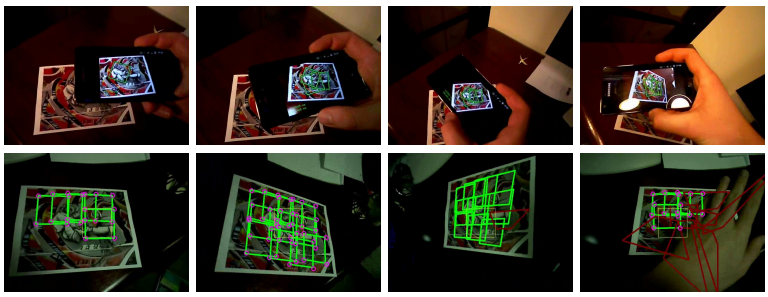


Fig. 7. Learning and tracking of multiple templates. The upper row shows tracking of multiple templates on a mobile phone while the lower row shows tracking of a large number of templates on a standard PC. This helps to handle occlusions.

the fast learning characteristic of our approach, we are able to learn such a large number of templates online. This can be useful for a SLAM and similar systems where a patch-based reconstruction of a scene is performed.

6 Conclusion

We introduced an efficient method for online learning of linear predictors for real-time template tracking by reformulating the original learning procedure presented by Jurie and Dhome [1]. This removes the time consuming inversion of large matrices and dramatically reduces the learning time. In addition, our approach yields tracking results comparable to those of the standard approach while sensitivity to image noise is reduced. Furthermore, the robustness in tracking can be increased by adding new training samples to an already learned tracker. Lastly, we demonstrated the usefulness of the proposed learning approach in a tracking application for mobile devices, where online learning is necessary.

References

1. Jurie, F., Dhome, M.: Hyperplane approximation for template matching. PAMI (2002)

2. Holzer, S., Ilic, S., Navab, N.: Adaptive linear predictors for real-time tracking. In: CVPR, 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. IJCV (2000)
5. Hager, G., Belhumeur, P.: Efficient region tracking with parametric models of geometry and illumination. PAMI (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. PAMI (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. IJCV (2004)
10. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: ICRA. (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: BMVC. (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. PAMI (2009)
18. Özuysal, M., Fua, P., Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. In: CVPR, Minneapolis, MI, USA (2007)
19. Holzer, S., Hinterstoisser, S., Ilic, S., Navab, N.: Distance transform templates for object detection and pose estimation. In: CVPR. (2009)
20. Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., Lepetit, V.: Online learning of patch perspective rectification for efficient object detection. In: CVPR. (2008)
21. Hinterstoisser, S., Kutter, O., Navab, N., Fua, P., Lepetit, V.: Real-time learning of accurate patch rectification. In: CVPR. (2009)
22. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of texture-less objects. In: CVPR. (2010)
23. Gräßl, C., Zinßer, T., Niemann, H.: Illumination insensitive template matching with hyperplanes. In: Proceedings of Pattern recognition: 25th DAGM Symposium, Magdeburg, Germany (2003)
24. Penrose, R.: A generalized inverse for matrices. In: Proceedings of the Cambridge Philosophical Society. (1955)