Noname manuscript No.
(will be inserted by the editor)

Efficient Learning of Linear Predictors for Template Tracking

Stefan Holzer · Slobodan Ilic · David Tan · Marc Pollefeys · Nassir Navab

Received: date / Accepted: date

Abstract The research on tracking templates or image patches in a sequence of images has been largely dominated by energy-minimization-based methods. However,since its introduction in [22], the learning-based approach called Linear Predictors has proven to be an efficient and reliable alternative for template tracking, demonstrating superior tracking speed and robustness. But, their time intensive learning procedure prevented their use in applications where online learning is essential. Indeed, [18] presented an iterative method to learn Linear Predictors; but it starts with a small template that makes it unstable at the beginning.

Therefore, we propose three methods for highly efficient learning of full-sized Linear Predictors – where the first one is based on Dimensionality Reduction using the Discrete Cosine Transform; the second is based on an efficient reformulation of the learning equations; and, the third is a combination of both. They show different characteristics with respect to learning time and tracking robustness, which makes them suitable for different scenarios.

S. Holzer, S. Ilic, D. Tan, N. Navab
Technische Universität München
Fakultät für Informatik, I-16
Boltzmannstr. 3
85748 Garching b. München, Germany
E-mail: {holzers, Slobodan.Ilic, tanda, navab}@in.tum.de

M. Pollefeys
Computer Vision and Geometry lab (CVG)
Institute of Visual Computing
Department of Computer Science - ETH Zurich
CNB G105
Universitatstrasse 6
CH-8092 Zurich, Switzerland
E-mail: marc.pollefeys@inf.ethz.ch

Keywords Template Tracking · Linear Predictors · Learning · Dimensionality Reduction · Discrete Cosine Transform (DCT)

1 Introduction

Due to its large range of applications, template tracking has been extensively studied in the past. The main task of template tracking is to follow a specified template over time, *i.e.* over a sequence of consecutive images. This is done by estimating the parameters of the template warping function, defining how the image data occupied by the template is warped from one frame to another. Examples of such warping functions are affine transformations or homographies. Applications can be found in areas such as augmented reality, human-computer interfaces, medical imaging, surveillance, vision-based control, or visual reconstruction.

While energy minimization has become a common technique for estimating template parameters from frame to frame, tracking-by-detection methods became popular recently, as they reached a state where template parameter estimation is possible at or close to frame rate. A further alternative are learning based approaches, where the relation between image intensity differences and template parameters is learned using exemplary data. While energy minimization approaches are flexible at run-time and tracking-by-detection methods do not put constraints on inter-frame motions, learning based techniques have shown to allow much faster online tracking.

One of the most successful learning based approaches for template tracking are Linear Predictors, or template tracking using hyper-plane approximation, proposed by Jurie and Dhome [22]. Although fast and

robust, it contains a computationally expensive learning phase which prohibits it from being used in scenarios where new scenes need to be handled online. We address this limitation by presenting three learning approaches which lead to learning times that are up to two orders of magnitude faster than [22].

In the remainder of this paper, we first discuss related work (Sec. 2) and introduce the original approach as proposed by Jurie and Dhome [22] (Sec. 3). We then present our learning approaches starting with a dimensionality reduction approach which makes use of the Discrete Cosine Transform (Sec. 4), introduced in [20]; followed by a reformulation of the learning process (Sec. 5), introduced in [21]; and finally a combination of the two (Sec. 6). New training data is added by updating an existing Linear Predictor (Sec. 7) [12]. We then demonstrate the usefulness of our proposed approaches using mobile applications (Sec. 9). In the experiments (Sec. 8) we analyze the characteristics of the proposed approaches under different scenarios and with respect to the related work.

2 Related Work

Template tracking approaches can be categorized in three different sets of methods: tracking-by-detection (TBD) [12,14,15,13,16,17,33], template tracking based on energy minimization [29,36,11,5,7,1,2,30,4,6,35], as well as learning based methods [8,24,22,23,10,34, 31,32,38,18,19]. While tracking-by-detection methods do not put constraints on the possible location of the template to track, *i.e.* the template is searched in the whole image independent of its location in the previous frame, they are, in general, significantly slower than frame-to-frame tracking approaches. The time consuming training procedure and the restrictions in the possible pose space further limit these approaches. Energy minimization based approaches usually have the advantage in creating and modifying templates online, while learning based methods are significantly better in tracking speed. Additionally, Jurie *et al.* [22] showed that Linear Predictors are superior to Jacobian approximation in terms of tracking speed and robustness, and we showed in [18] that Linear Predictors outperform methods based on energy minimization such as Efficient Second-order Minimization [4] (ESM).

Tracking-by-Detection-based approaches. We can further categorize TBD approaches into methods that are based on keypoint detection and on a sliding window which is also known as template matching. Özuysal *et al.* [33] introduced FERNs where keypoints are extracted and then classified by estimating the probability of each keypoint falling under a specific class. Unfortunately, the corresponding learning process is time consuming and is error-prone if the number of available keypoints is limited, which is the case if small regions are considered. Holzer *et al.* [17] presented an approach called Distance Transform Templates (DTTs), where the distance transform is used to extract closed contours, and to describe and match the extracted contours using FERNs [33]. This approach needs a significant amount of time in learning, and detects objects only at about 10 frames per second. In [12,14], Hinterstoisser *et al.* proposed two patch-based TBD methods, where keypoints are used as starting point to match patches and estimate their pose. Although the use of keypoints enables processing at almost video framerate, it constrains its repeatability by the underlying keypoint detector. In their recent work [15,13,16], Hinterstoisser *et al.* overcome the limitations of keypoint detectors by using sliding window approaches. Indeed, this allows a robust object detection; however, it imposes restrictions on the possible pose space since the necessary processing increases with the amount of covered poses. Even with reasonable constraints on the pose space, it is still significantly slower than frame-to-frame tracking.

Keyframe-based approaches. This type of approach relies on creating a set of keyframes that represent the object or scene and are used in an optimization in order to obtain stable pose estimation [37] and, if the model of the object or scene is not available, create a map of the tracked environment [25]. The keyframes are hereby either created in an offline process from a known model [37] or selected online in case the model needs to be created while tracking [25]. These keyframe-based approaches usually rely on keypoints or other discriminative image features, *e.g.* edges [26], which are tracked from frame to frame and matched against keyframes. Similar to some of the Tracking-by-Detection-based methods, keyframe-based approaches tend to become error-prone if the number of available keypoints or features is limited. This especially holds if the object or region of interest is small. However, if sufficient image features are available, these approaches provide very stable and robust tracking.

Energy minimization-based approaches. These approaches build upon the early work of Lucas and Kanade [29]. Improvements since then include: different update rules of the warp function [29,11,5,36,7,1], handling of occlusions and illumination changes [11], as well as considering different orders of approximation of the error function [30,4]. When looking at the different

update rules that have been proposed over time, we can classify them into four categories, which are the additive apporach [29], the compositional approach [36], the inverse additive approach [11,5] and the inverse compositional approach [7,1]. The inverse approaches hereby switch the roles of the reference and the current image, which allows to move some of the computations into the initialization phase, making the tracking phase more efficient. Hager and Belhumeur [11] addressed the problems in illumination changes and occlussions. The usage of second-order instead of first-order approximations led to a faster convergence speed and larger convergence areas [30,4]. For a more detailed and complete overview of energy-based tracking methods, we refer the readers to [2].

Learning-based approaches. Learning-based approaches can be divided into approaches that apply offline or online learning. Online learning is often used to track objects that are not known a-priori or change appearance over time. Prominent examples of an online learning approach is the tracking approach of Grabner *et al.* [8], as well as the Tracking-Learning-Detection approach of Kalal *et al.* [24]. Both approaches are semi-supervised approaches where the tracking process is split into a frame-to-frame tracker, a detector that localizes all observed appearances of the tracked object and corrects the tracker if necessary, as well as a learning procedure which estimates the errors of the detector and updates it in order to avoid them in future. While [8] uses a semi-supervised online boosting strategie for learning, [24] uses so called PN learning where two sets of experts are used to estimate missed detections and false alarms. These combinations of tracking, detection and learning allow to handle changes in appearance as well as occlusions. However, while they give good results when tracking a bounding box around an object they are not suited to track the full pose of objects.

A prominent approach in tracking based on offline learning is tracking using Linear Predictors. Linear Predictors for template tracking or template tracking using hyperplane approximation was first introduced by Jurie and Dhome [22]. Their proposed method used randomly warped samples of the initial template to learn Linear Predictors and applies them to predict the parameter updates in template tracking. In contrast to previous methods, the "Jacobians" are here computed once during the learning phase and the parameter updates are computed using simple matrix multiplications. A more detailed description of this method is in Sec. 3.2. An extension of this method in order to handle occlusions has been presented in [23]. Gräßl *et al.* showed how invariance with respect to illumination changes can be achieved [9] and how tracking accuracy can be increased by intelligently selecting the points for sampling from the image data [10]. Zimmermann *et al.* [38] moved away from using one big single template into numerous small templates, track them individually, and combine their seperate motion estimates into a single one. Holzer *et al.* [18,19] presented a method for adapting existing Linear Predictors in order to modify online the covered area of template while tracking. This reduces the initial learning time by start tracking with a small template and grow it over time. Instead of creating a predictor, Mayol and Murray [32] collect a set of training samples in order to fit the current sampling region to this pre-trained set using general regression.

Among the learning based approaches using Linear Predictors, none of them can learn large templates at run-time. Therefore, we present learning approaches with this criterion.

3 Tracking Framework

Our proposed template tracking approaches are built from the work of Jurie and Dhome [22]. While we introduce new learning procedures that significantly reduce the necessary learning time, we keep the tracking stage the same. In this section, we introduce our notations and review the method proposed by Jurie and Dhome [22].

3.1 Template and Parameter Description

Without loss of generality, we use a rectangular template that covers an area of $n_s = w \cdot h$ square pixels and locate sample points that uniformly subsample the template into a grid of n_p points as shown in Fig. 1. The sample points are used to efficiently describe the image intensities in the template instead of using its full resolution. From the template in Fig. 1, we define the parameter vector $\boldsymbol{\mu} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)^\top$ that contains the image coordinates of the four corners which are used to parameterize a homography. Note that neither our proposed approaches nor the approach of Jurie and Dhome [22] are limited to this type of sample point arrangement, rectangular shape or transformation.

3.2 Template Tracking based on Linear Predictors

Given a template in the reference image, the goal of template tracking is to follow the selected template from one frame to the next and to estimate its pose in each frame.

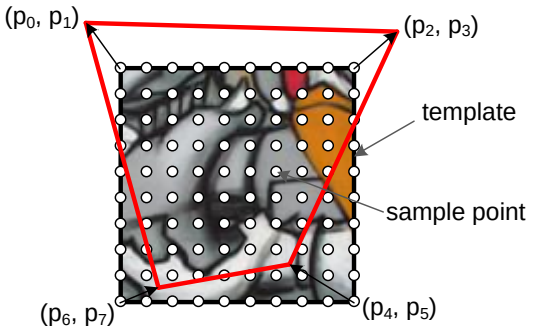


Fig. 1 A template is represented by a set of regularly placed sample points. Its pose is parameterized using its four corner points.

In the reference image, we define the initial location of the template using the parameter vector $\boldsymbol{\mu}_R$ that stores the eight parameters from the four corners of the template, and the image intensity vector $\mathbf{i}_R = (i_{R,1}, i_{R,2}, \cdots, i_{R,n_p})^\top$ that has the image intensity at the corresponding sample points; while we assign $\boldsymbol{\mu}_C$ as the current parameter vector and $\mathbf{i}_C$ as the image intensity vector in the current frame. However, unlike $\mathbf{i}_R$, the elements of $\mathbf{i}_C$ are the image intensities extracted from the current frame at the sample point locations of the template pose from the previous frame or previous iteration $\boldsymbol{\mu}_{C-1}$.

To track the location of the template and estimate the pose in the current frame, Jurie and Dhome [22] introduce the Linear Predictor matrix $\mathbf{A}$ that relates the parameter vectors and the intensity vectors as:

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

where $\delta\mathbf{i} = \mathbf{i}_C - \mathbf{i}_R$ is the image intensity difference and $\delta\boldsymbol{\mu}$ is the parameter update. This tracking algorithm is summarized in Alg. 1.

Algorithm 1 Tracking using Linear Predictors

```
function Track (in Image I,
  in TemplateParameters μ_{C-1},
  out TemplateParameters μ_C)
Compute homography T_μ from μ_{C-1}.
for level = 1 → n_l do
  for iteration = 1 → n_i do
    Extract image data from I at sample points warped
    with T_μ.
    Normalize image data.
    Compute image difference vector δi.
    Compute parameter update δμ = A_{level} δi.
    Compute homography T_{δμ} from δμ.
    T_μ ← T_μ T_{δμ}.
  end for
end for
Compute μ_C from T_μ.
```

The $8 \times n_p$ matrix $\mathbf{A}$ is precomputed by imposing n_t, where $n_t \gg n_p$, random transformations $\delta\boldsymbol{\mu}_i$ for $i = 1, \ldots, n_t$ on $\boldsymbol{\mu}_R$ and computing the corresponding image difference vectors $\delta\mathbf{i}_i$. These vectors are concatenated into an $8 \times n_t$ matrix $\mathbf{Y} = \left(\delta\boldsymbol{\mu}_1, \delta\boldsymbol{\mu}_2, \ldots, \delta\boldsymbol{\mu}_{n_t}\right)$ and $n_p \times n_t$ matrix $\mathbf{H} = (\delta\mathbf{i}_1, \delta\mathbf{i}_2, \ldots, \delta\mathbf{i}_{n_t})$, which are used to reformulate Eq. (1) as:

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

Using a closed-form solution, we solve for $\mathbf{A}$ as:

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

This formulation requires to invert an $n_p \times n_p$ matrix $\mathbf{H}\mathbf{H}^\top$ where a typical value of n_p is $20 \cdot 20 = 400$. Due to the large size of this matrix, learning Linear Predictors using Eq. (3) is computationally expensive, and limits their use in real-time applications where the environment is unknown and templates have to be learned online. Throughout the paper, we focus on reformulating the learning procedure to find $\mathbf{A}$ in Eq. (3) and to evaluate the robustness in tracking for each learning procedure.

In practice, the image data vector $\mathbf{i}$ is normalized such that the elements have zero mean and unit standard deviation, which increases the robustness against illumination changes. Furthermore, to prevent $\mathbf{H}\mathbf{H}^\top$ from being rank-deficient due to the zero mean of the vectors, we add random noise to the normalized image intensity difference vectors in $\mathbf{H}$.

3.3 Multi-Layered Tracking

To improve the tracking performance, we apply a multi-layer approach where we compute n_l Linear Predictors $\mathbf{A}_1, \ldots, \mathbf{A}_{n_l}$ and use one Linear Predictor per layer. Each of these Linear Predictors is trained for different amounts of motion where the first one is trained for large motions and the latter ones for consecutively smaller motions. Additionally, each of the Linear Predictors is applied multiple times in tracking. Within this paper, we use $n_l = 5$ and three iterations for each predictor. The complete algorithm for learning Linear Predictors is given in Alg. 2.

4 Efficient Predictor Learning using Dimensionality Reduction

Our first approach reduces the size of $\delta\mathbf{i}_i$ from n_p to n_r by applying the Discrete Cosine Transform (DCT). As a consequence, it reduces the number of rows in $\mathbf{H}$ from n_p to n_r and the size of $\mathbf{H}\mathbf{H}^\top$ from $n_p \times n_p$ to $n_r \times n_r$.

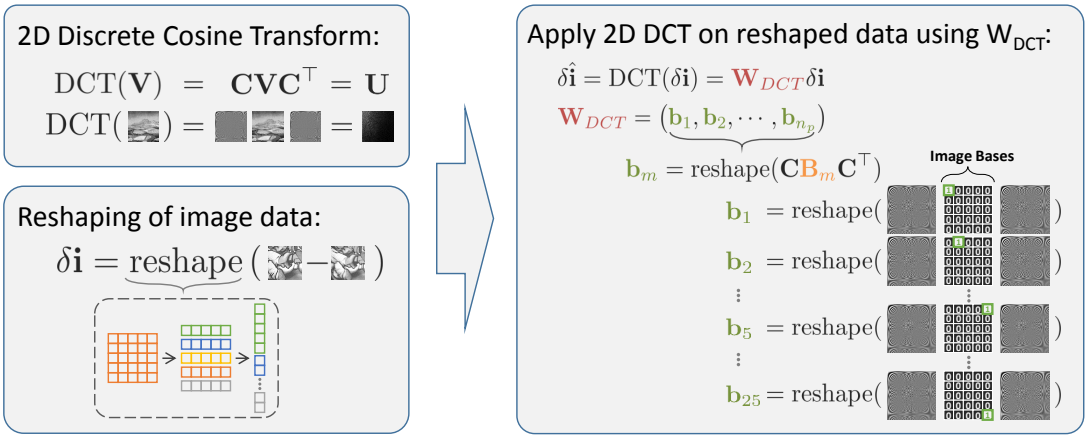


Fig. 2 Demostration of how the 2D-DCT is applied on reshaped image data.

Algorithm 2 Learning Linear Predictors

function *Learn* (***in*** Image $\mathbf{I}$,
in TemplateParameters $\boldsymbol{\mu}$,
out Set of Linear Predictors $\mathcal{A} = \{\mathbf{A}_i\}$)
for $level = 1 \rightarrow n_l$ **do**
Create set of random transformations and put them into the matrix $\mathbf{Y} = (\delta\boldsymbol{\mu}_1, \delta\boldsymbol{\mu}_2, \ldots, \delta\boldsymbol{\mu}_{n_t})$.
Apply random transformations on $\boldsymbol{\mu}$ and extract image differences $\mathbf{H} = (\delta\mathbf{i}_1, \delta\mathbf{i}_2, \ldots, \delta\mathbf{i}_{n_t})$ from $\mathbf{I}$.
Normalize image data and add random noise.
Compute Linear Predictor $\mathbf{A}_{level}$
Add $\mathbf{A}_{level}$ to $\mathcal{A}$.
end for

In general, DCT is known to give good results in image compression. This process involves transforming the image into the frequency space and removing the DCT coefficients that correspond to high frequencies. In relation to our approach, it is advantageous to filter the high frequencies because they tend to include noise and de-stabilize tracking.

Mathematically, the 2-dimensional DCT $\mathbf{U}$ of a $k \times k$ matrix $\mathbf{V}$ is defined as:

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

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

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

Contrary to Eq. (4) where the 2D DCT is applied on a 2D matrix, our approach needs to apply the DCT on a reshaped 1D vector $\delta\mathbf{i}_i$ as $\delta\hat{\mathbf{i}}_i = \mathrm{DCT}(\delta\mathbf{i}_i)$ to form the matrix $\hat{\mathbf{H}} = (\delta\hat{\mathbf{i}}_1, \delta\hat{\mathbf{i}}_2, \ldots, \delta\hat{\mathbf{i}}_{n_t})$.

Therefore, we define an $n_p \times n_p$ matrix $\mathbf{W}_{DCT}$ that directly maps the image difference vectors $\delta\mathbf{i}_i$ to their DCT counterparts $\delta\hat{\mathbf{i}}_i$ (see Fig. 2). By using a function that converts the elements of a 2D matrix $\mathbf{V}_i$ into a 1D column vector $\delta\mathbf{i}_i$ as $\delta\mathbf{i}_i = \mathrm{reshape}(\mathbf{V}_i)$, where $\mathbf{V}_i$ is the 2D template image data in frame i, we can 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. This makes the set of matrices $\{\mathbf{B}_1, \ldots, \mathbf{B}_{n_p}\}$ as the base of the template in the image space and the set of vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_{n_p}\}$ as their DCT projection. As a result, the 2D DCT of the image difference vectors is computed 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 order to integrate this into 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:

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

Note that by integrating the DCT computation directly into the Linear Predictor matrix $\mathbf{A}$ we do not have to modify the tracking procedure.

We induce the dimensionality reduction by defining an $n_r \times n_p$ submatrix $\mathbf{W}_{DCT}^{(n_r)}$ with $n_r < n_p$. In this case, 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. Hence, the final Linear Predictor is computed as:

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

where $\hat{\mathbf{H}}^{(n_r)} = \mathbf{W}_{DCT}^{(n_r)}\mathbf{H}$.

The experiments in Sec. 8.1 show that keeping n_r significantly small reduces the learning time for large templates. Moreover, depending on the size of n_r, the reduction in learning even increases tracking robustness.

5 Reformulating the Learning Equations

Another way to increase the learning speed is to reformulate the learning equations such that it is no longer necessary to compute the pseudo-inverse of $\mathbf{H}$. Starting from Eq. (2), we first apply the pseudo-inverse of $\mathbf{Y}$ from the right which leads to:

$$\mathbf{I}_{8\text{x}8} = \mathbf{A}\mathbf{H}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1} = \mathbf{A}\mathbf{B}, \tag{12}$$

where

$$\mathbf{B} = \mathbf{H}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1} \tag{13}$$

is an $n_p \times 8$ matrix. Hence, the Linear Predictor $\mathbf{A}$ is computed as:

$$\mathbf{A} = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top. \tag{14}$$

Note that the pseudo-inverse is applied differently in Eqs. (12) and (14) because the rows are linearly independent in $\mathbf{Y}$ while the columns are linearly independent in $\mathbf{B}$ [3].

Now, the learning process involves the inversion of the matrices $\mathbf{Y}\mathbf{Y}^\top$ and $\mathbf{B}^\top\mathbf{B}$ in Eqs. (12) and (14), respectively. However, both are 8×8 matrices and can be quickly calculated. Additionally, $\mathbf{Y}\mathbf{Y}^\top$ can be precomputed which requires us to invert a single 8×8 matrix online.

To avoid encoding of fixed offsets in the linear mapping, $\mathbf{Y}$ is normalized such that each parameter has zero mean and unit standard deviation. Accordingly, $\delta\boldsymbol{\mu}$ is de-normalized after using Eq. (1) in tracking. Unlike the normalization in Sec. 3.2 where we aim 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 $\mathbf{Y}$.

If we compare the formulation of $\mathbf{A}$, we can see that in Eq. (12) from our approach, $\mathbf{H}$ is approximated by orthogonally projecting it on $\mathbf{Y}$; while in Eq. (3) from the original approach of Jurie and Dhome [22], $\mathbf{Y}$ is approximated by orthogonally projecting it on $\mathbf{H}$. Given that we project $\mathbf{H}$ on $\mathbf{Y}$, all noise outside the low-rank space represented by $\mathbf{Y}$ has no effect; while in the case of Jurie and Dhome [22], the noise has more effect. Therefore, this learning process signigicantly improves tracking robustness with respect to noise as validated in Sec. 8.3.

6 Combining Dimensionality Reduction and Reformulation of Learning

To combine the dimensionality reduction in Sec. 4 with the reformulation in Sec. 5, we replace $\mathbf{H}$ in Eq. (13) by the dimensionality reduced version $\hat{\mathbf{B}} = \mathbf{W}_{DCT}^{-1}\hat{\mathbf{H}}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1}$; thus, similar to Eq. (14), we obtain:

$$\mathbf{A} = \left(\hat{\mathbf{B}}^\top\hat{\mathbf{B}}\right)^{-1}\hat{\mathbf{B}}^\top. \tag{15}$$

By assigning the $n_t \times 8$ matrix $\mathbf{Z}$ as:

$$\mathbf{Z} = \mathbf{Y}^\top\left(\mathbf{Y}\mathbf{Y}^\top\right)^{-1}, \tag{16}$$

we can write $\hat{\mathbf{B}}^\top\hat{\mathbf{B}}$ in the form of:

$$\hat{\mathbf{B}}^\top\hat{\mathbf{B}} = \mathbf{Z}^\top\hat{\mathbf{H}}^\top\mathbf{W}_{DCT}^{-\top}\mathbf{W}_{DCT}^{-1}\hat{\mathbf{H}}\mathbf{Z} \tag{17}$$

and since $\mathbf{W}_{DCT}$ is orthogonal (*i.e.* its inverse is equal to its transpose), this can be simplified to:

$$\hat{\mathbf{B}}^\top\hat{\mathbf{B}} = \mathbf{Z}^\top\hat{\mathbf{H}}^\top\hat{\mathbf{H}}\mathbf{Z}. \tag{18}$$

Therefore, the final learning equation then becomes:

$$\mathbf{A} = \left(\mathbf{Z}^\top\hat{\mathbf{H}}^\top\hat{\mathbf{H}}\mathbf{Z}\right)^{-1}\mathbf{Z}^\top\hat{\mathbf{H}}^\top. \tag{19}$$

Again, this formulation only requires us to invert an 8×8 matrix $\mathbf{Z}^\top\hat{\mathbf{H}}^\top\hat{\mathbf{H}}\mathbf{Z}$ which can be quickly calculated.

As we will show in the experiments, this leads to an compromise between the strenghts and weaknesses of both approaches.

7 Online Updating

After learning a Linear Predictor, new training samples can be added using the Sherman-Morrison formula as demonstrated in Hinterstoisser *et al.* [12]. However, this relies on the original approach of computing Linear Predictors by efficiently updating $\mathbf{S} = (\mathbf{H}\mathbf{H}^\top)^{-1}$. Since in our reformulated learning approach the matrix $\mathbf{S}$ is not computed as an intermediate result, we find a way to derive it from an existing Linear Predictor $\mathbf{A}$. Thus, using Eq. (3),

$$\mathbf{A} = \mathbf{Y}\mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)^{-1} = \mathbf{Y}\mathbf{H}^\top\mathbf{S} = \mathbf{D}\mathbf{S}, \tag{20}$$

where $\mathbf{D} = \mathbf{Y}\mathbf{H}^\top$ is a $8 \times n_p$ matrix. This implies that $\mathbf{S}$ can be directly computed using the pseudo-inverse of $\mathbf{D}$ as:

$$\mathbf{S} = \mathbf{D}^\top(\mathbf{D}\mathbf{D}^\top)^{-1}\mathbf{A}. \tag{21}$$

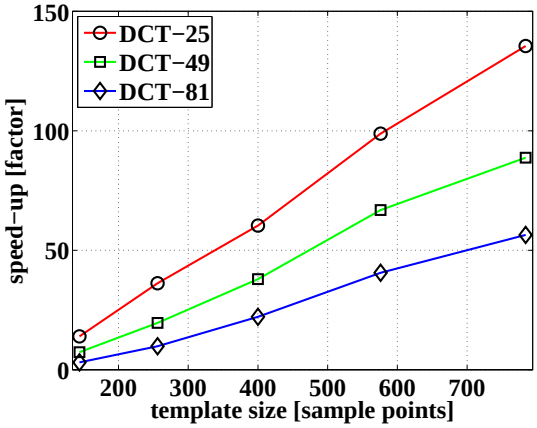
Here, $\mathbf{D}\mathbf{D}^\top$ is again an 8×8 matrix and therefore, can be efficiently inverted. Therefore, we update $\mathbf{S}$ as:

$$\hat{\mathbf{S}} = \left(\mathbf{S}^{-1} + \delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^\top\right)^{-1} \tag{22}$$

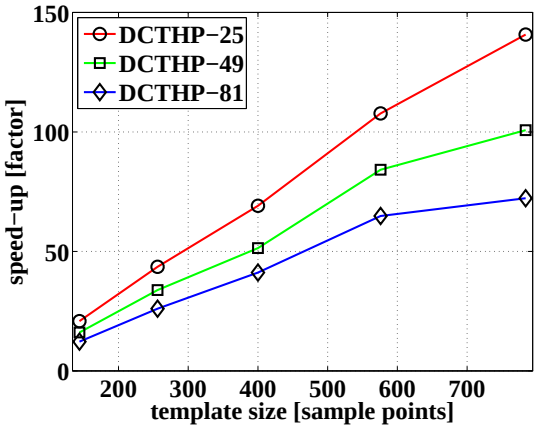
$$= \mathbf{S} - \frac{\mathbf{S}\delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^\top\mathbf{S}_I}{1 + \delta\mathbf{i}_{n_t+1}^\top\mathbf{S}\delta\mathbf{i}_{n_t+1}}, \tag{23}$$

where $\delta\mathbf{i}_{n_t+1}$ is a vector of image intensity differences obtained from a new random transformation applied to the sample points.

Note that we also have to update the matrices $\mathbf{H}$ and $\mathbf{Y}$ before computing the updated linear predictor in Eq. (20). This is done by concatenating them with the new training samples where the parameter differences are normalized in the same way as in the initial learning.



(a) Speed-up of DCT



(b) Speed-up of DCTHP

Fig. 5 Analysis of speed-ups for both DCT-based approaches with respect to the original approach of Jurie and Dhome [22]. **(a)** Speed-up of the approach only based on dimensiontality reduction (DCT). **(b)** Speed-up of the combined approach (DCTHP).

8 Experiments

In this section, we evaluate our approaches for efficient learning of Linear Predictors as proposed in Sec. 4 (DCT), Sec. 5 (HP) and Sec. 6 (DCTHP), and compare them to the original approach of Jurie and Dhome [22] (JD) as well as to Efficient Second-order Minimization (ESM) introduced by Benhimane *et al.* [4]. Our comparison is based on two kinds of evaluation – timing and tracking performance. The former is used to evaluate the differences in tracking and learning time while the latter reveals the impact gained from the learning performance on the tracking robustness with respect to different kinds of motions as well as its sensitivity to noise.

We used C++ implementations for all algorithms. The Efficient Second-order Minimization [4] is from the publicly available binaries[1] and the original approach of Jurie and Dhome [22] is from our own implementation where it is similarly optimized as the proposed approaches. For the evaluations, we used a standard notebook with a 2.26GHz Intel(R) Core(TM)2 Quad CPU and 8GB of RAM with only a single core used for computations.

All synthetic experiments are done using a template size of 150×150 pixels, the template is taken from the center of an image and tracking is applied on its warped versions. Fig. 3 shows the images used for the synthetic experiments which were selected randomly from the internet. The reason for focusing on synthetic experiments is that these allow an accurate comparison using ground truth. This also has the benefit that it is simple to estimate the influence of single parameters, compared to other methods of testing, like using markers on real scenes, which generate their own error and limit the amount of available motion. It further makes it pos-

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

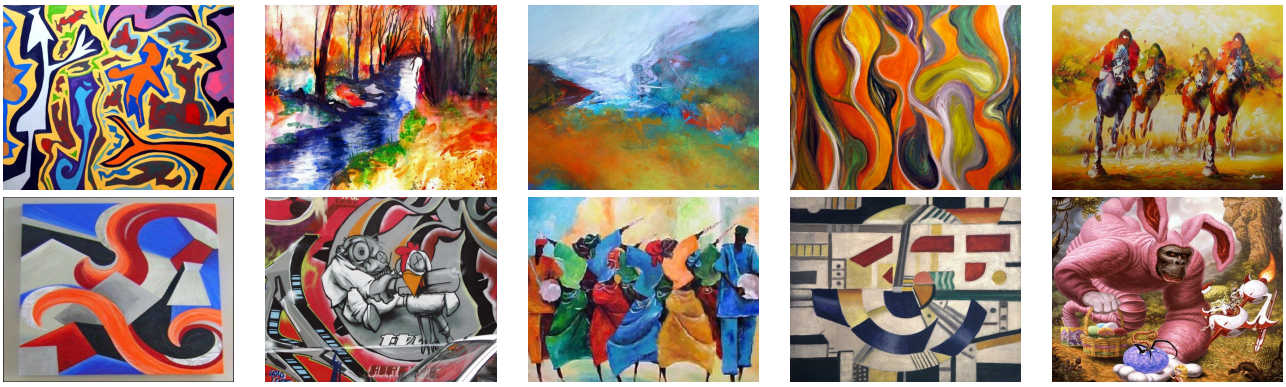
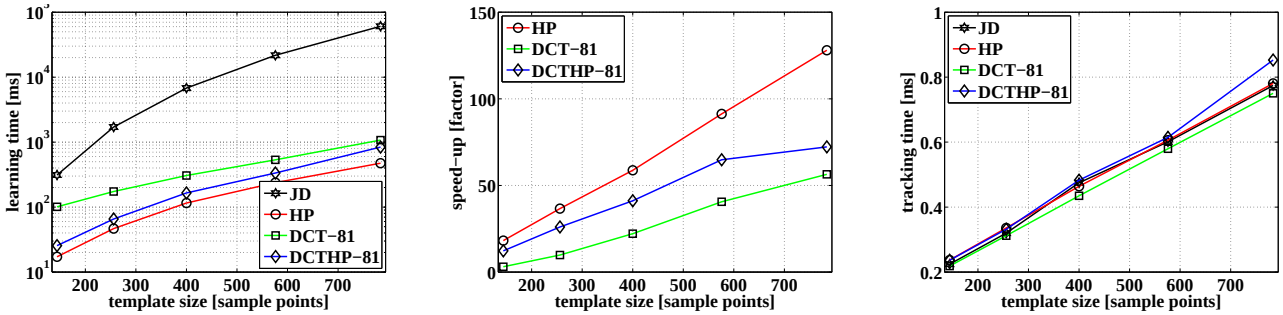


Fig. 3 A set of images selected from the internet, which is used for synthetic experiments.



(a) Learning times (b) Speed-up (c) Tracking times

Fig. 4 Evaluation of learning and tracking times. **(a)** Learning times. **(b)** Speed-up obtained by the proposed methods, where for the dimensionality reduction based approaches 81 DCT-coefficients are used. **(c)** Tracking times.

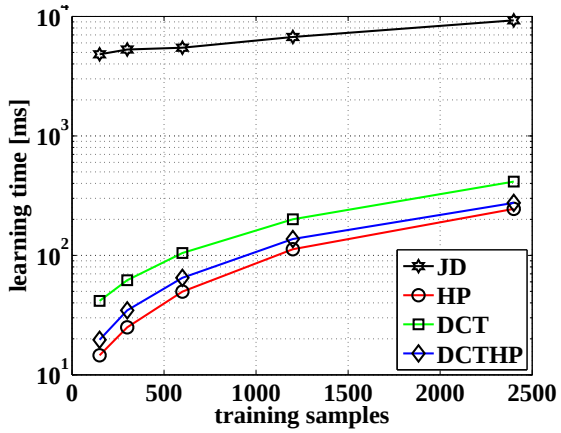


Fig. 9 Evaluation of the influence of the number of training samples on the necessary learning time.

sible to specify the exact amount of change. This allows a fair evaluation between the different approaches.

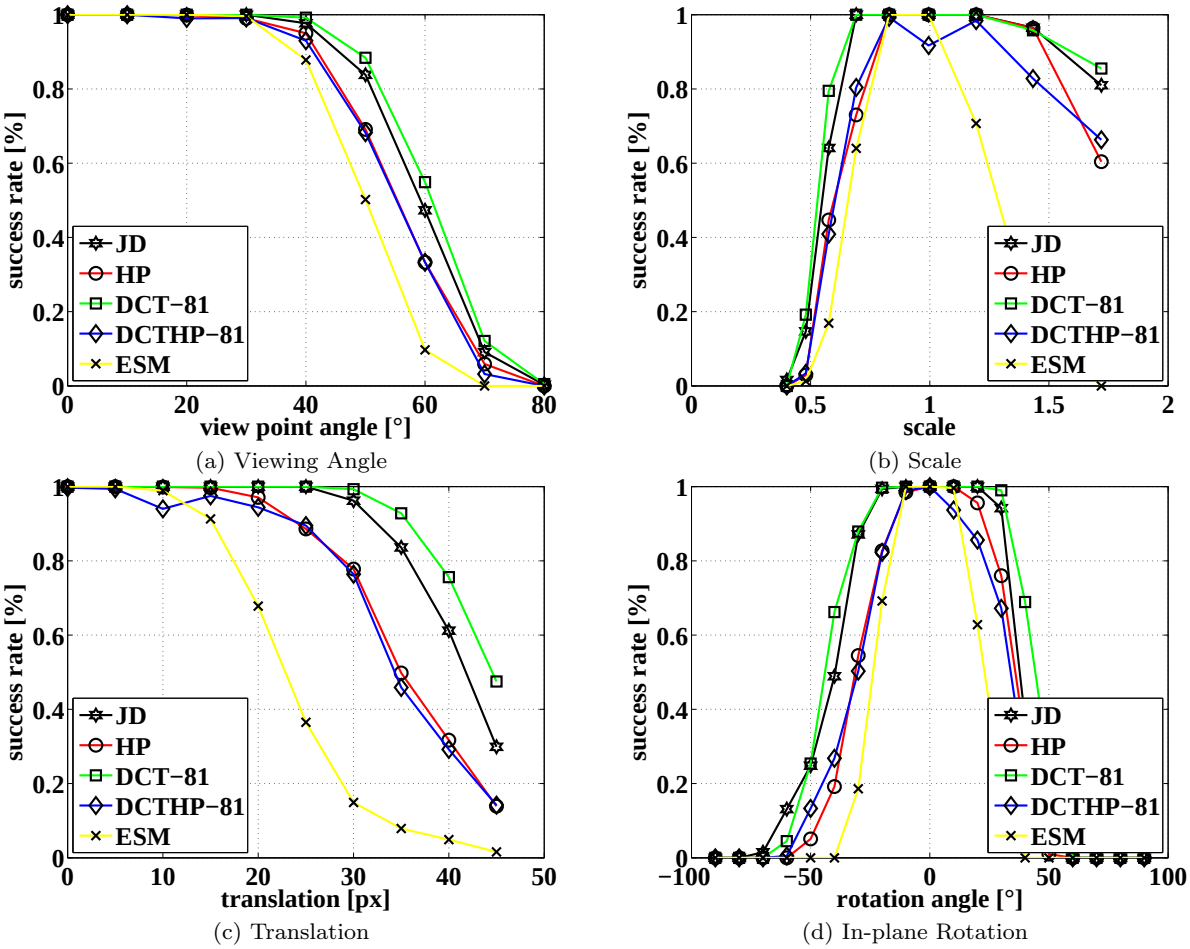
The homographies for the various tests are compute as follows:

- **Translations.** For a specified reference amount of translation t pixels we compute a set of random translations values in the range of $t \pm 5$ pixels. The translation is applied in a random direction.
- **Scale.** For a specified scale s we compute a set of random scale values in the range $[t, t \cdot 1.2]$. The scaleing is applied relative to the center of the template.
- **Rotation.** For a specified angle α in degree we compute a set of random rotation values in the range of $\alpha \pm 5$ degree. The center of the rotation is equal to the center of the template.
- **In-plane rotation.** For the in-plane rotation tests we rotate the plane of the template around a random axis which is lying in the template plane. For a specified angle β in degree we compute a set of random rotation values in the range of $\beta \pm 5$ degree. The axis of the rotation goes through the center of the template.

In Sec. 8.6 we analyse our approaches on real data and compare them to several state-of-the-art approaches [38, 18, 19, 23, 22, 29, 28, 24].

8.1 Computational Complexity

In order to analyze the computational complexity, we measure the time a specific phase of the approaches, *i.e.* learning and tracking, needs to complete.



(a) Viewing Angle (b) Scale (c) Translation (d) In-plane Rotation

Fig. 6 Evaluation of tracking success rate with respect to different types of motion.

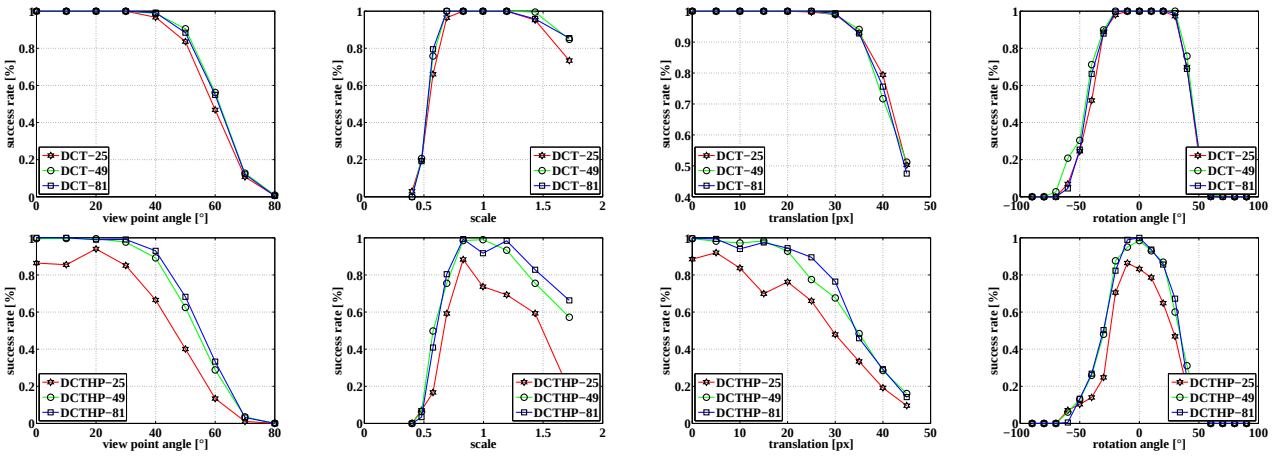
Learning The necessary learning time reflects our main contribution. In Fig. 4 (a), we compare the learning times with respect to the template size in number of sample points for the approach of Jurie and Dhome [22] (JD), our dimensiontality reduction approach with 81 DCT-coefficients (DCT-81), our approach with reformulated learning (HP), and our combined approach with 81 DCT-coefficients (DCTHP-81). All our proposed methods are significantly faster in learning than the original approach. Fig. 4 (b) compares the speed-up of the different proposed approaches with respect to Jurie and Dhome [22]. This also shows that for larger templates the difference between the learning times gets even bigger. In detail, the reformulation approach (HP) is the fastest if considering 81 DCT-coefficients for the other approaches, while the combined approach (DCTHP) is the second fastest, followed by the pure dimensionality reduction. Note that by using less DCT-coefficients, the DCT-based approaches can achieve a similar speed-up as the reformulation in Fig. 5. Considering template sizes with more than 800 sample points, *e.g.* for a 30×30 template, our approaches reach learning times which are more than two orders of magnitude faster than the approach of Jurie and Dhome [22].

Tracking When we compare the tracking times with respect to the template size in Fig. 4 (c), all proposed approaches need approximately the same amount of time for tracking as the approach of Jurie and Dhome [22] and can even track large templates at more than 1000 fps. In contrast to this, the energy minimization based approach of Benhimane *et al.* [4] needs approximately 10 ms to track the same template.

8.2 Robustness

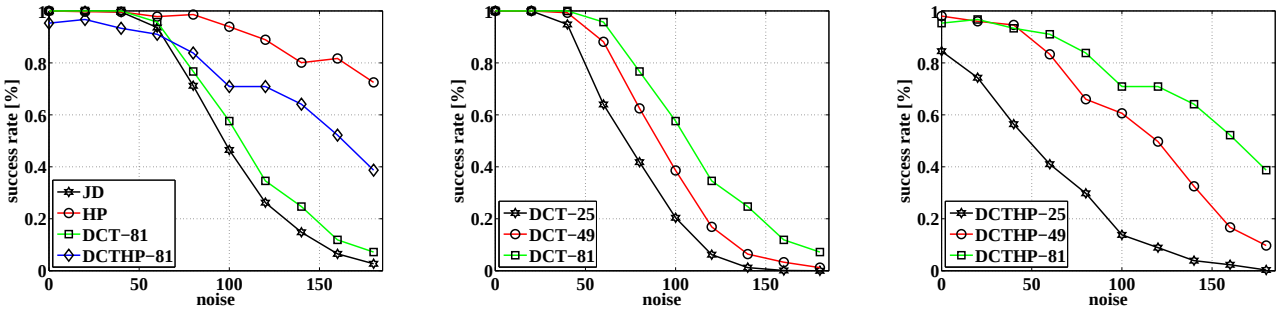
We analyze the success rates in tracking, where all approaches are considered in Fig. 6 and the effect of the number of DCT-coefficients n_r used in dimensionality reduction influences the tracking robustness of both DCT-based approaches is evaluated in Fig. 7.

The success rate in tracking is measured by finding the correct location of the template after the introduction of random transformations to several test images.



(a) Viewing Angle (b) Scale (c) Translation (d) In-plane Rotation

Fig. 7 Evaluation of tracking sucess rate for both DCT-based approaches in order to analyse the effect of the used number of DCT-coefficients. The first row shows the approach which is only based on dimensionality reduction and the second row the combined approach.



(a) (b) (c)

Fig. 8 Evaluation of the influence of noise on tracking success. **(a)** All approaches, where for the DCT-based approaches 81 DCT-coefficients are used. The evaluation of the influence of different numbers of DCT-coefficients for **(b)** the approach only based on dimensionality reduction and for **(c)** the combined approach. The shown results were compute from translation tests where a random translation of 10 ± 5 pixels is applied.

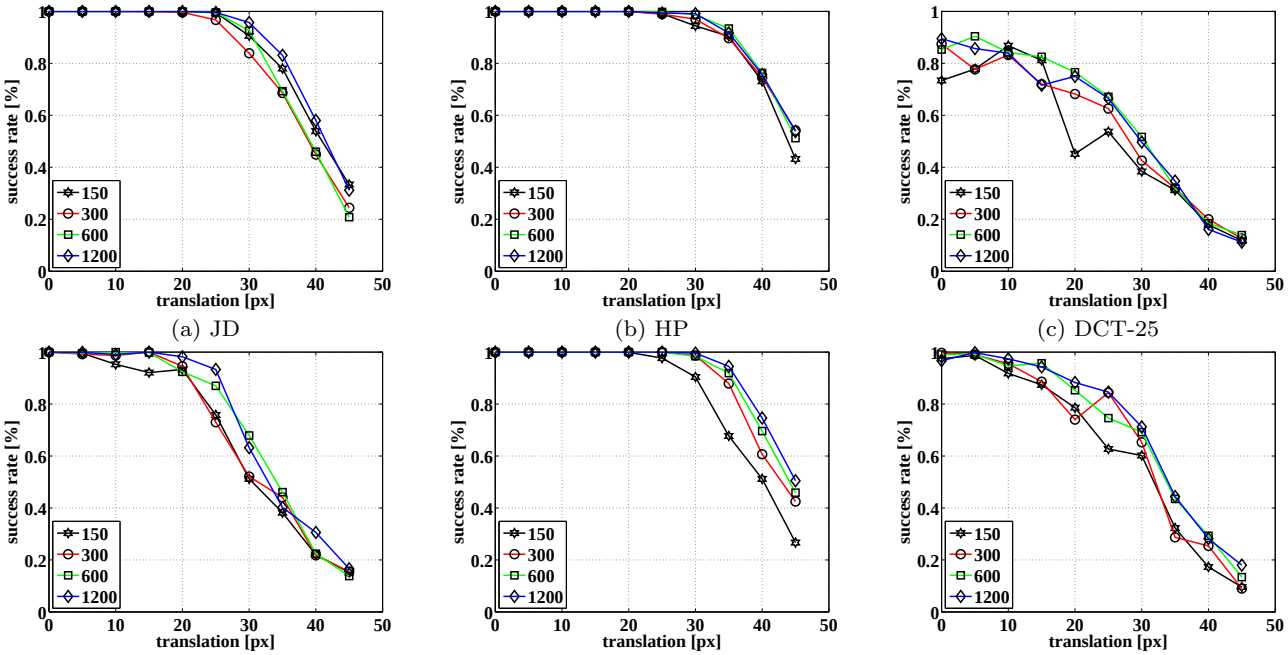
The considered random transformations contain translation, rotation, scale, and viewpoint change. Tracking is considered successfull by computing the mean pixel distance between the reference template corner points and the tracked template corner points, which are back-projected into the reference view. If this mean difference is less than 5 pixels then the tracking is considered to be successful. The template size for the following experiments is 150×150 pixels with 20×20 sample points, if not otherwise mentioned. The initial learning of the Linear Predictors is done using $3 \cdot 18 \cdot 18 = 972$ training samples. For the non-linear approach of Benhimane *et al.* [4] the complete template without sub-sampling is used.

Our experiments show that the approach of Benhimane [4] gives the worst results, our approach based on dimensionality reduction using the DCT gives the best results, followed by the original approach of Jurie and Dhome [22]. The approaches using the reformulation of the learning process give slightly worse results than Jurie and Dhome [22]. However, as we show in Sec. 8.5, this can be compensated by adding new training samples to the Linear Predictors.

When evaluating the influence of the number of used DCT-coefficients in Fig. 7, we see that the pure DCT-based approach gives similar results when using 81, 49, or 25 DCT-coefficients, while the combined approach shows a significant drop in success rate when using only 25 DCT-coefficients.

8.3 Noise

Fig. 8 shows the influence of noise on the tracking process. In order to measure the robustness of the considered approaches with respect to noise we corrupt the test images with noise drawn from a Gaussian distribution. The evaluation is done by increasing the standard deviation of the Gaussian distribution. This is similar to



(a) JD (b) HP (c) DCT-25

(d) DCT-81 (e) DCTHP-25 (f) DCTHP-81

Fig. 10 Analysis of the influence of the number of training samples on the tracking success rate for **(a)** Jurie and Dhome [22], **(b)** the reformulated learning, the dimensionality reduction based approach for **(c)** 25 DCT-coefficients and **(d)** 81 DCT-ceofficients, and for the combined approach for **(e)** 25 DCT-coefficients and **(f)** 81 DCT-coefficients.

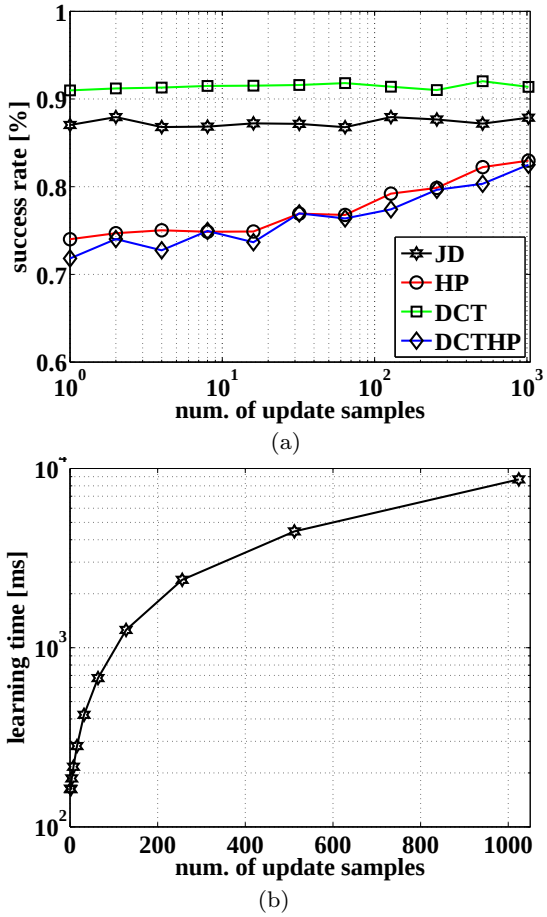
decreasing the signal-to-noise ratio. An example where this is important is in low-light conditions. This noise is added before we apply the random transformations. While Fig. 8 (a) compares all considered learning approaches, Fig. 8 (b) and Fig. 8 (c) focus on DCT and DCTHP, respectively, in order to show how the number of DCT-coefficients influences the robustness with respect to noise. Here, JD gives the worst results and HP clearly the best. The pure dimensionality reduction approach gives results that are slightly better than the approach of Jurie and Dhome [22] and the combined approach gives results approximately in the middle between DCT and HP. In both DCT-based approaches, using more DCT-coefficients helps to improve the robustness with respect to noise.

8.4 Number of Training Samples

In Fig. 9 we evaluate the necessary learning time with respect to the number of random training samples used for training. By reducing the number of training samples used for learning we can significantly reduce the necessary learning time. However, this can produce Linear Predictors that result in less stable tracking. When looking at Fig. 10, we see that reducing the number of training samples has different effects for different approaches. While JD, DCT, and DCTHP show a certain variance in the resulting tracking success rate, the pure HP approach shows only a marginal difference between using 150 training samples and 1200 training sample. The variance in the JD approach seems not to be related to the number of the training samples, while DCT and DCTHP show a clear relation between tracking robustness and the number of training samples.

8.5 Online Updating

In Fig. 11 we analyse the influence of updating Linear Predictors after learning, as described in Sec. 7. While the tracking robustness of JD and DCT is not influenced by updating, the HP-based approaches both significantly improve when being updated. However, this can be done online while tracking and therefore, does not influence the initial learning time. The reason for the improvement for the HP-based approaches can be seen in the fact that in the reformulated learning process of HP the training data is first projected onto a low-dimensional space, which makes the learning step less dependent on the number of training samples. However, the online updating is done in a different way where the training data is not projected onto a low-dimensional space and therefore, shows a more significant impact.



(a)

(b)

Fig. 11 **(a)** Evaluation of the influence of online updating on tracking robustness in case of random translations. The shown values are computed as average success rate of translation tests for translations ranging from 0 to 45 pixels. These average success rates are computed for different numbers of training samples. **(b)** Necessary time for updating.

8.6 Real Data

In this section, we evaluate our proposed approaches on real data and analyze their results in relation to other state-of-the-art approaches. This includes quantitative evaluations on the datasets of Zimmermann *et al.* [38] and Lieberknecht *et al.* [27].

8.6.1 Zimmermann Dataset

In Table 1, we evaluate our approaches on the datasets provided by Zimmermann *et al.* [38] and compare it to the results of other approaches as reported in [38,18, 19]. Apart from our approaches, we include the results from Zimmermann *et al.* [38] (NoSLLiP), Adaptive Linear Predictors [18] (ALPs), Multi-Layered Adaptive Linear Predictors [19] (ML-ALPs), an extension of the original Linear Predictor approach that is able to handle occlusions [23] (LLiP LS, LLiP LS 1/2), Lukas-Kanade template tracking [29] (LK), SIFT [28], as well as the tracking approach of Kalal *et al.* [24] (TLD). Figure 12 shows a representative image for each of the Zimmermann *et al.* [38] datasets.

Speed is the most obvious difference between our approaches and the other approaches. Similar to Jurie and Dhome [22], we achieve processing frame rates of about 1800 fps. While the ALPs approach still achieves almost 100 fps, all the others show frame rates below 25 fps and are therefore significantly slower. Note, that all three datasets (TOWEL, PHONE, MP) include frames where the template partly leaves the visible image region or is partly occluded. Therefore, approaches that are able to handle occlusions or out-of-image-scenarios (NoSLLiP, ALPs, ML-ALPs, LLiP LS 1/2) show the best loss-of-locks numbers.

For our algorithms, we see that the HP approach shows significant problems with the provided datasets. Among the datasets, it performs best in the MOUSEPAD sequence. The purely DCT-based approach performs well, compared to the original approach of Jurie and Dhome [22] (JD), in the TOWEL and MOUSEPAD sequences, but shows problems in the PHONE sequence.

As we confirm in Section 8.6.2, all presented approaches seem to have problems with repetitive textures. The DCTHP approach performs in a range between the HP and DCT approach, except for the phone sequence, where it fails in most cases. Since both, HP and DCT have problems with repetitive texture, this seems to add up in the combined approach. Again, similar observations can be made in Section 8.6.2.

In defense of TLD, we have to note that despite the high loss-of-locks numbers it is able to track the approximate location of the object of interest in the PHONE and MOUSEPAD datasets. But, it fails to do this most of the times in the TOWEL dataset. TLD is able to redected the tracked object when they lose tracking which poses a significant advantage. However, if other tracking approaches are combined with a detector a re-detection is possible for them too. While TLD is not able to track the full pose of the objects in the sequences, it also shows a tendency to drift apart from the initally selected template. Although it does not lose track of the template, the center of the bounding box can not be used to mark the center of the object in a stable way. The same holds for the extent of the bounding box which fluctuates over time. Its processing speed ranges within approximately $5 - 15$fps and therefore, our proposed approaches are up to two orders of magnitude faster in tracking than TLD. In general, TLD might be well suited for tracking deformable or unconstrained objects that change their appearance online.

Fig. 12 Representative images of the three datasets provided by Zimmermann *et al.* [38]. From left to right: the towel-, phone-, and mouse-pad-dataset.

However, if the change in appearance can be described via a model, *e.g.* a homography, other algorithms such as the ones shown in Table 1 seem more suitable.

8.6.2 Lieberknecht Dataset

For the evaluation in Table 2, we applied the original approach of Jurie and Dhome [22] as well as our proposed approaches on the datasets provided by Lieberknecht *et al.* [27]. For this evaluation, ground truth data is given every 250th frame and therefore, if tracking is lost, it can only be reinitialized at every 250th frame. The final evaluation results are obtained by providing the tracking results to the authors of the dataset. A frame is counted as successfully tracked if the average error of its tracked corners is less than 10 pixels. These corners are at an artificial position outside of the actual image data. Although this increases the effect of small misalignments, it prohibits from tracking data close to these control points in order to falsely improve results.

The Lieberknecht *et al.* [27] dataset considers four different texture scenarios (low, repetitive, normal, high) and five different motion and illumination scenarios (angle, range, fast far, fast close, illumination). For each texture scenario, two different scenes are provided (see Figure 13). Exemplary changes for the motion and illumination scenarios are shown in Figure 14. Before going into details of the evaluation, we want to note that a exact comparison between the success rates of the different approaches has to be done with caution, as the success percentage heavily depends on which of the 250 frame window the tracking failure occurs.

In general, the presented methods show good results on the low and normal texture scenarios. The success rate drops for DCT-based methods if the number of DCT-coefficients is reduced. In the 'Low-1' dataset, the original approach of Jurie and Dhome [22] (JD) is actually outperformed by the presented methods. In the case of 'Low-2', 'Normal-1', and 'Normal-2' similar results as achieved by JD are obtained. In the cases where the refumulated learning approach (HP) works well, it outperforms the other method in the illumination scenarios. This can be accounted to the increased robustness with respect to low signal-to-noise ratios, as shown in Section 8.3.

For the repetitive scenarios, we see that the reformulated learning (HP) shows significant problems. The purely DCT based methods still perform acceptable and in a similar range as JD. However, the combination of both approaches (DCTHP) also fails in these scenarios. The most challenging sequences are the high texture sequences. While DCT81 and DCT49 still achieve similar results as JD in 'High-2' all proposed methods show a significant drop in performance on 'High-1'. This can be explained by the fine details of the texture which are necessary to be considered for successfull tracking. Due to the compression applied in the DCT-based approaches, they tend to lose these details and therefore fail.

9 Applications

Due to their high efficiency, our proposed approaches are well-suited for applications using mobile devices. Therefore, we implemented them on a standard mobile phone with 1 GB of RAM and a 1.2 GHz dual core processor where no optimization for processor specific technology is implemented, and only a single core is used for learning and tracking.

The learning process for a template with 16×16 sample points and $16 \cdot 16 \cdot 3 = 768$ training samples took approximately 18000 ms for the original approach of Jurie and Dhome [22], about 600 ms for our proposed approach using dimensiontality reduction, and roughly 350 ms for the approach using the reformulation of the learning equation. Based on these results, the reformulated approach is more than 50 times faster than the approach of Jurie and Dhome [22] and allows interac-

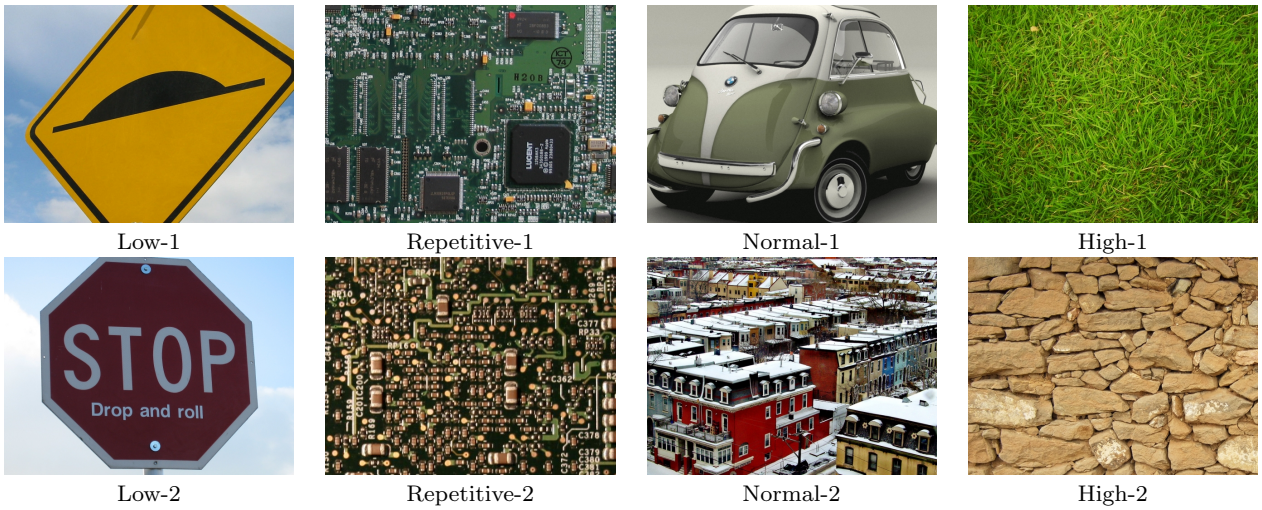


Fig. 13 Representative images of the datasets provided by Lieberknecht *et al.* [27]. From left to right column: low, repetitive, normal, and high texture.

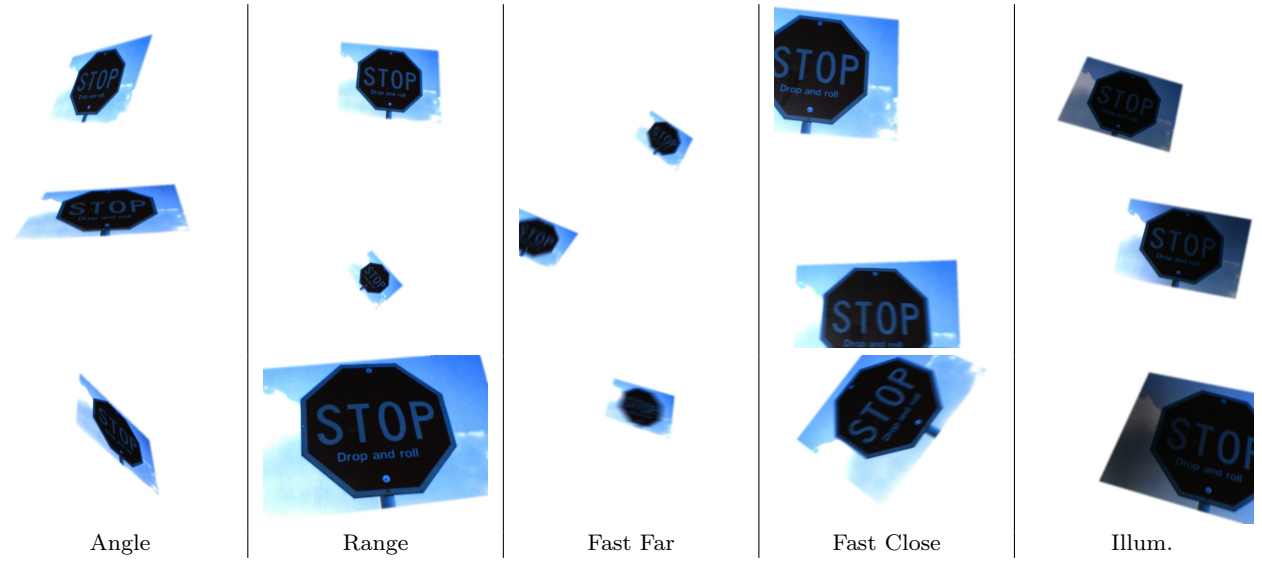


Fig. 14 Representative images of the Low-2 dataset provided by Lieberknecht *et al.* [27] to illustrate the different motion and illumination scenarios used in Table 2.

tive applications to start tracking almost immediately. On the other hand, tracking takes about 2.5 ms for all approaches.

10 Conclusions

Linear Predictors were first introduced by Jurie and Dhome [22] and enable robust tracking at very high frame rates. They are best suited for tracking the pose, *e.g.* as defined by a homography, of a textured object.

The main goal of this paper is to overcome the long learning time which was the main drawback of their approach. Thus, we presented three different methods to speed-up this learning procedure for template tracking. While they all significantly reduced learning time in two orders of magnitude range, they individually have different properties that makes them suited for different application schemes.

Our dimensionality reduction based approach in Sec. 4 gives the highest tracking success rate. By choosing appropriate number of DCT-coefficients in learning, it reaches similar learning speeds as the other approaches and even works with a very low number of training samples. However, if a significant amount of noise is present, the reformulation of the learning process in Sec. 5 gives the best results. But this approach

Low-1	Angle	Range	Fast Far	Fast Close	Illum.
JD	71.2%	57.8%	25.5%	12.2%	61.8%
HP	98.1%	71.5%	49.8%	15.8%	86.9%
DCT81	90.6%	70.6%	51.8%	12.9%	82.2%
DCT49	62.3%	53.2%	19.0%	12.2%	57.3%
DCT25	62.5%	50.1%	16.2%	11.0%	69.3%
DCTHP81	90.6%	70.6%	51.8%	12.9%	82.2%
DCTHP49	90.3%	56.8%	21.9%	13.2%	77.4%
DCTHP25	74.0%	51.7%	21.1%	11.5%	75.1%

Low-2	Angle	Range	Fast Far	Fast Close	Illum.
JD	100%	73.6%	33.5%	31.6%	79.2%
HP	98.6%	75.3%	21.2%	27.6%	79.6%
DCT81	96.0%	73.7%	28.4%	27.8%	76.2%
DCT49	92.8%	72.2%	26.9%	17.4%	74.2%
DCT25	28.0%	8.2%	2.7%	4.3%	19.2%
DCTHP81	99.6%	73.7%	21.1%	23.4%	75.7%
DCTHP49	89.3%	78.8%	20.3%	16.6%	73.8%
DCTHP25	6.8%	6.2%	4.8%	2.4%	7.5%

Repetitive-1	Angle	Range	Fast Far	Fast Close	Illum.
JD	70.1%	69.1%	33.4%	20.6%	78.0%
HP	18.2%	9.4%	7.8%	4.2%	18.8%
DCT81	69.8%	62.8%	28.3%	15.2%	74.2%
DCT49	68.4%	54.2%	35.2%	14.0%	67.5%
DCT25	28.0%	8.2%	2.7%	4.3%	19.2%
DCTHP81	14.2%	5.0%	3.8%	2.5%	9.2%
DCTHP49	3.0%	1.6%	4.2%	0.1%	2.1%
DCTHP25	1.7%	6.6%	3.3%	1.8%	2.5%

Repetitive-2	Angle	Range	Fast Far	Fast Close	Illum.
JD	84.1%	47.8%	9.4%	34.8%	92.5%
HP	6.1%	1.5%	2.3%	0.6%	10.2%
DCT81	76.9%	35.6%	10.1%	19.9%	79.6%
DCT49	76.6%	21.8%	7.8%	13.2%	71.8%
DCT25	18.7%	6.2%	1.2%	2.2%	22.5%
DCTHP81	2.4%	0.3%	0.2%	0.9%	0.4%
DCTHP49	0.4%	0.2%	0.2%	0.3%	0.8%
DCTHP25	0.3%	1.0%	0.6%	0.6%	0.3%

Normal-1	Angle	Range	Fast Far	Fast Close	Illum.
JD	96.8%	60.4%	23.6%	13.7%	72.8%
HP	99.4%	51.2%	21.7%	15.8%	75.5%
DCT81	94.0%	46.2%	22.8%	12.4%	71.8%
DCT49	63.3%	15.8%	6.1%	11.6%	65.1%
DCT25	37.1%	4.8%	4.7%	6.9%	19.7%
DCTHP81	96.8%	54.2%	8.8%	12.8%	69.8%
DCTHP49	55.9%	33.1%	7.9%	12.2%	51.0%
DCTHP25	27.5%	5.0%	4.6%	7.0%	15.9%

Normal-2	Angle	Range	Fast Far	Fast Close	Illum.
JD	97.8%	55.2%	17.2%	12.3%	82.2%
HP	75.3%	55.2%	15.5%	11.3%	96.9%
DCT81	96.5%	54.6%	20.4%	12.0%	77.7%
DCT49	54.1%	36.6%	12.7%	11.2%	72.2%
DCT25	32.2%	4.9%	4.0%	6.3%	18.8%
DCTHP81	88.5%	54.9%	15.0%	10.9%	74.0%
DCTHP49	75.9%	51.2%	14.5%	10.8%	82.0%
DCTHP25	9.8%	0.8%	0.4%	2.8%	2.5%

High-1	Angle	Range	Fast Far	Fast Close	Illum.
JD	44.2%	8.2%	5.8%	5.1%	67.4%
HP	17.2%	4.0%	4.3%	3.3%	33.4%
DCT81	9.4%	4.7%	2.2%	3.2%	36.1%
DCT49	1.7%	1.8%	1.8%	2.3%	15.2%
DCT25	0.5%	0.4%	0.3%	0.7%	2.7%
DCTHP81	3.5%	4.5%	2.5%	2.4%	48.7%
DCTHP49	0.8%	1.8%	0.4%	0.7%	6.1%
DCTHP25	0.1%	0.2%	0.1%	0.5%	0.4%

High-2	Angle	Range	Fast Far	Fast Close	Illum.
JD	29.6%	14.8%	8.1%	19.2%	50.8%
HP	6.8%	6.2%	4.8%	2.4%	7.5%
DCT81	26.8%	12.2%	7.8%	13.5%	50.9%
DCT49	29.5%	12.2%	7.2%	9.6%	49.1%
DCT25	10.4%	4.0%	6.0%	4.8%	22.7%
DCTHP81	20.3%	6.3%	6.0%	9.8%	21.1%
DCTHP49	13.4%	3.7%	4.3%	3.6%	0.5%
DCTHP25	1.8%	0.4%	0.9%	2.4%	1.8%

Table 2 Comparison of the approach of Jurie and Dhome [22] (JD) with our proposed approaches (HP, DCT, DCTHP) on the datasets of Lieberknecht *et al.* [27]. The results show the tracking success rate under different texture, motion, and illumination conditions. Results for other approaches can be found in [27].

leads to a slightly decreased success rate in tracking, especially if the object of interest shows a repetitive texture. Lastly, the combined approach in Sec. 6 is a compromise between the two other approaches and has the advantage of making a trade-off between learning speed, robustness to noise and robustness to large motions.

11 Acknowledgements

This work was partly funded by Willow Garage, Inc., California, USA.

References

1. Baker, S., Matthews, I.: Equivalence and efficiency of image alignment algorithms. In: Conference on Computer Vision and Pattern Recognition (2001)
2. Baker, S., Matthews, I.: Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision (2004)
3. Ben-Israel, A., Greville, T.N.: Generalized Inverses. Springer-Verlag (2003)
4. Benhimane, S., Malis, E.: Homography-based 2d visual tracking and servoing. International Journal of Robotics Research (2007)
5. 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)

Method	Object	Frame-rate [fps]	Loss-of-locks [-/-]	Error [%]
NoSLLiP	Towel	21.8	5/3229	2.1
JD	Towel	1887.8	74/3230	1.8
HP	Towel	1785.2	731/3230	3.2
DCT-81	Towel	1888.5	62/3230	2.2
DCT-49	Towel	1875.9	75/3230	2.9
DCT-25	Towel	1871.2	474/3230	5.5
DCTHP-81	Towel	1879.9	558/3230	2.6
DCTHP-49	Towel	1849.5	1628/3230	4.7
DCTHP-25	Towel	1839.8	2819/3230	8.8
TLD	Towel	11.0	2967/3230	12.3
NoSLLiP	Phone	16.8	20/1799	1.8
ALPs	Phone	96.7	10/2299	1.2
JD	Phone	1861.7	99/2299	2.0
HP	Phone	1758.8	1765/2299	4.0
DCT-81	Phone	1862.4	214/2299	3.8
DCT-49	Phone	1883.7	2018/2299	5.1
DCT-25	Phone	1886.1	1951/2299	14.5
DCTHP-81	Phone	1827.7	2271/2299	2.0
DCTHP-49	Phone	1869.8	2203/2299	11.7
DCTHP-25	Phone	1871.6	2299/2299	16.0
TLD	Phone	7.7	1623/2299	11.7
NoSLLiP	MP	18.9	13/6935	1.5
ML-ALPs	MP	17.2	10/6945	2.1
SIFT	MP	0.5	281/6935	1.4
LK (IC)	MP	2.6 (25)	398/6935	2.4
LLiP LS	MP	24.4	1083/6935	6.3
LLiP LS 1/2	MP	24.2	93/6935	3.0
JD	MP	1773.2	249/6945	2.0
HP	MP	1666.9	524/6945	2.2
DCT-81	MP	1779	260/6945	2.5
DCT-49	MP	1799.6	254/6945	2.9
DCT-25	MP	1877.6	1579/6945	6.0
DCTHP-81	MP	1776.3	619/6945	2.6
DCTHP-49	MP	1794	466/6945	3.4
DCTHP-25	MP	1868.9	1378/6945	4.3
TLD	MP	8.5	6150/6945	12.5

Table 1 Comparison between the tracking results of NoSLLiP, SIFT, an extension of the original Linear Predictor approach that is able to handle occlusions [23] (LLiP LS, LLiP LS 1/2), Lukas-Kanade template tracking [29] (LK), all as given in [38], as well as results for ALPs, Multi-layered ALPs (ML-ALPs), the approach of Jurie and Dhome (JD), TLD [24], and our proposed approaches (HP, DCT, DCTHP). ALPs, ML-ALPs, JD, and our approaches are implemented in C++. For the evaluation of SIFT, Zimmermann *et al.* [38] used a publicly available implementation. All other methods were implemented in Matlab.

6. Dame, A., Marchand, E.: Accurate real-time tracking using mutual information. In: Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on (2010)
7. Dellaert, F., Collins, R.: Fast image-based tracking by selective pixel integration. In: ICCV Workshop of Frame-Rate Vision (1999)
8. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: Computer Vision ECCV 2008, *Lecture Notes in Computer Science*, vol. 5302, pp. 234–247. Springer Berlin Heidelberg (2008)
9. Gräßl, C., Zinßer, T., Niemann, H.: Illumination insensitive template matching with hyperplanes. In: Proceedings of Pattern recognition: 25th DAGM Symposium (2003)
10. Gräßl, C., Zinßer, T., Niemann, H.: Efficient hyperplane tracking by intelligent region selection. In: Image Analysis and Interpretation (2004)
11. Hager, G., Belhumeur, P.: Efficient region tracking with parametric models of geometry and illumination. IEEE Transactions on Pattern Analysis and Machine Intelligence (1998)
12. 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 (2008)
13. Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., Lepetit, V.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: IEEE International Conference on Computer Vision (ICCV) (2011)
14. Hinterstoisser, S., Kutter, O., Navab, N., Fua, P., Lepetit, V.: Real-time learning of accurate patch rectification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2009)
15. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of texture-less objects. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010)
16. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., , Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: Asian Conference on Computer Vision (2012)
17. Holzer, S., Hinterstoisser, S., Ilic, S., Navab, N.: Distance transform templates for object detection and pose estimation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2009)
18. Holzer, S., Ilic, S., Navab, N.: Adaptive linear predictors for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010)
19. Holzer, S., Ilic, S., Navab, N.: Multi-layer adaptive linear predictors for real-time tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence (2013)
20. Holzer, S., Ilic, S., Tan, D., Navab, N.: Efficient learning of linear predictors using dimensionality reduction. In: Asian Conference on Computer Vision (2012)
21. Holzer, S., Pollefeys, M., Ilic, S., Tan, D.J., Navab, N.: Online Learning of Linear Predictors for Real-Time Tracking. In: 12th European Conference on Computer Vision (ECCV) (2012)
22. Jurie, F., Dhome, M.: Hyperplane approximation for template matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (2002)
23. Jurie, F., Dhome, M.: Real time robust template matching. In: British Machine Vision Conference (2002)
24. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1409–1422 (2012)
25. Klein, G., Murray, D.: Parallel tracking and mapping for small ar workspaces. In: Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, pp. 225–234 (2007)
26. Klein, G., Murray, D.: Improving the agility of keyframe-based slam. In: Computer Vision ECCV 2008, *Lecture Notes in Computer Science*, vol. 5303, pp. 802–815. Springer Berlin Heidelberg (2008)

27. Lieberknecht, S., Benhimane, S., Meier, P., Navab, N.: A dataset and evaluation methodology for template-based tracking algorithms. In: ISMAR, pp. 145–151 (2009)
28. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
29. Lucas, B., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: International Joint Conference on Artificial Intelligence (1981)
30. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: IEEE International Conference on Robotics and Automation (2004)
31. Matas, J., Zimmermann, K., Svoboda, T., Hilton, A.: Learning efficient linear predictors for motion estimation. In: Computer Vision, Graphics and Image Processing (2006)
32. Mayol, W.W., Murray, D.W.: Tracking with general regression. Journal of Machine Vision and Applications (2008)
33. Özuysal, M., Fua, P., Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. In: Conference on Computer Vision and Pattern Recognition (2007)
34. Parisot, P., Thiesse, B., Charvillat, V.: Selection of reliable features subsets for appearance-based tracking. Signal-Image Technologies and Internet-Based System (2007)
35. 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)
36. Shum, H.Y., Szeliski, R.: Construction of panoramic image mosaics with global and local alignment. International Journal of Computer Vision (2000)
37. Vacchetti, L., Lepetit, V., Fua, P.: Stable real-time 3d tracking using online and offline information. Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(10), 1385–1391 (2004)
38. Zimmermann, K., Matas, J., Svoboda, T.: Tracking by an optimal sequence of linear predictors. IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)