Adaptive Linear Predictors for Real-Time Tracking

Stefan Holzer, Slobodan Ilic, Nassir Navab

Department of Computer Science, Technical University of Munich (TUM)
Boltzmannstrasse 3, 85748 Garching, Germany
{holzers,slobodan.ilic,navab}@in.tum.de

Abstract

Enlarging or reducing the template size by adding new parts, or removing parts of the template, according to their suitability for tracking, requires the ability to deal with the variation of the template size. For instance, real-time template tracking using linear predictors, although fast and reliable, requires using templates of fixed size and does not allow on-line modification of the predictor. To solve this problem we propose the Adaptive Linear Predictors (ALPs) which enable fast online modifications of pre-learned linear predictors. Instead of applying a full matrix inversion for every modification of the template shape as standard approaches to learning linear predictors do, we just perform a fast update of this inverse. This allows us to learn the ALPs in a much shorter time than standard learning approaches while performing equally well.

We performed exhaustive evaluation of our approach and compared it to standard linear predictors and other state of the art approaches.

1. Introduction

Template tracking has been extensively studied and used in many computer vision applications such as vision-based control, human-computer interfaces, surveillance, medical imaging and visual reconstruction.

While there are many template tracking approaches based on the analytical derivation of the Jacobian [14, 19, 9, 5, 6, 1, 2, 15, 3, 4], learning-based methods [12, 13, 8, 18, 16, 17, 20] have proved to allow faster tracking and are generally more robust with respect to large perspective changes.

A very successful learning based template tracker was proposed by Jurie and Dhome [12]. It is based on learning linear predictors to efficiently compute template parameter updates. The costly off-line learning phase, however, prohibits this method from computing templates of varying size online.

Figure 1. An initial small template is enlarged according to a tracking quality measure. The template is tracked over time and reduced if parts of it go out of the image. The removed parts are reinserted as soon as they become visible.

Yet, the ability to dynamically change the template size is necessary in applications such as indoor SLAM. The fact that the 3D geometry of the scene is a priori unknown makes it necessary to initially rely on planar structures. In this case it is preferable to start from small-sized templates, in order to reduce the risk of loosing track due to non-planar structures, and to grow or shrink them online. Thus, the learning of large templates can be distributed over multiple frames while keeping the failure rate low. In combination with a planarity check this strategy enables online segmentation of planar structures and the reliable maintenance of large templates. As a result, the set of initially tracked templates evolves towards a relatively small number of comparably large, optimally shaped templates, yielding increased robustness.

Current learning based tracking approaches, like [12], use templates of fixed size, because the computation of the linear predictors requires the costly inversion of a large, template specific matrix. Since this is the computationally most expensive part of the learning process, the ef-

fort for changing the template size is nearly equivalent to that of learning a new template from scratch. Therefore, to overcome the limitations of fixed size template approaches, while maintaining their robustness to large perspective changes, we propose an extension to linear predictors which allows efficient online modification of the template size. Instead of computing the inversion of the whole matrix every time the template shape changes, we introduce a way to update the inverse computationally efficient which dramatically reduces the time needed for learning. We start with a small initial template and grow it by small extension templates as defined in Fig. 3 according to their suitability for tracking. As long as the object to track is planar, our approach can grow the template in any direction which can result in an arbitrarily shaped template, as shown in Fig. 1. This breaks the standard, rectangular shape assumption widely used in current template tracking approaches and can be seen as a first step towards a dense SLAM system.

We perform extensive quantitative testing and compare our approach to the standard approach of Jurie and Dhome [12] under different transformations and noise levels, and to other state-of-the-art approaches in template tracking. We demonstrate that our approach performs equally well while requiring much shorter learning time. In the remainder of the paper we will discuss related work on template tracking, give a detailed description of our approach, present our results and show examples on real world sequences.

2. Related Work

A lot of effort has been made in the field of template tracking and image alignment since the work of Lucas and Kanade [14]. Most of the presented approaches can be put into one of the two categories: template tracking based on the analytical derivation of the Jacobian [14, 19, 9, 5, 6, 1, 2, 15, 3, 4] or based on learning [12, 13, 8, 18, 16, 17, 20]. While the analytical approaches generally are more flexible with respect to the template shape modification at run-time, learning approaches enable higher tracking speed and are more robust with respect to large perspective changes.

Since the seminal work of Lucas and Kanade [14] a large variety of analytical tracking approaches has been presented. Amongst others, these variations include different update rules of the warp [14, 9, 5, 19, 6, 1], different orders of approximations of the error function [15, 3, 4], occlusion and illumination change handling [9]. Basically, there are four different types of update rules, the additive approach [14], the compositional approach [19], the inverse additive approach [9, 5], and the inverse compositional approach [6, 1]. In the latter two, the roles of the reference and current image are switched, which makes it possible to move some of the computations into an initialization phase, that makes the tracking computationally very efficient. Faster convergence rate for a larger convergence area can be additionally obtained by using a second-order instead of a first order approximation of the error function [15, 3, 4]. Furthermore, Hager and Belhumeur [9] show how illumination changes and occlusions can be efficiently handled. For a more detailed overview over analytical tracking methods refer to Baker and Matthews [2].

In contrast to analytical tracking methods, Jurie and Dhome [12] propose an approach that learns linear predictors using randomly warped samples of the initial template. The linear predictors are then used to predict the parameter updates during tracking. This allows a very fast tracking, since the "Jacobians" are initially computed once and for all and the update parameters can be obtained by simple matrix vector multiplications. In [13] the authors also extend the approach in order to handle occlusion. Gräßl *et al*. [7] additionally shows how the robustness of the linear predictor based approach can be further increased with regard to illumination changes. They [8] also present an intelligent way how to select the points for sampling the image data, such that the accuracy of the tracking is increased. Another linear predictor approach [20] describes a template using many small templates and tracks these small templates independently. Based on the local movements of these small templates they estimate the movement of the large template. Instead of using linear predictors, Mayol and Murray [17] present an approach that fits the sampling region to pretrained samples using general regression.

All the proposed learning approaches, however, do not deal with templates of variable size. To overcome this limitation we developed a method that extends the approach of Jurie and Dhome [12] to allow online template size adaptation.

3. Background and Terminology

In this section we introduce notation and, for the sake of completeness, review the original template tracking approach proposed by Jurie and Dhome [12].

3.1. Template and Parameter Description

A template consists of a set of n_p sample points, which are distributed within the template region and are used to sample image data. The template parameters $\boldsymbol{\mu}$ describe the current deformation of the template within an image. Within this paper we use a homography to represent the current perspective distortion of a planar template and parameterize it using four points as shown in Fig. 2. Note that our approach can also be easily adapted to any other parameterizable template deformation.

The sample points are arranged in a regular grid and grouped together into subsets of four points as shown in

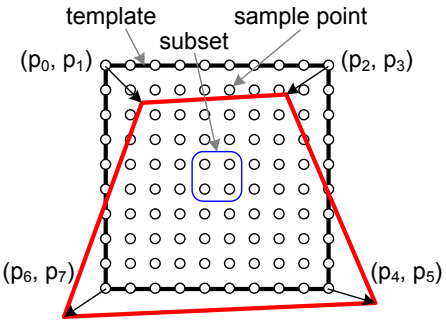


Figure 2. A template is represented by a set of regularly placed sample points, which are grouped into subsets of four points. The pose of a template is parameterized using four corner points.

Fig. 2. The usefulness of this grouping will be justified later in Section 4.1, when we describe our approach for template extension. However, neither the approach of Jurie and Dhome [12] nor our approach are restricted to this special kind of sample point arrangement. The image values obtained from the sample points, warped according to the current template parameters $\boldsymbol{\mu}$, are arranged in a vector $\mathbf{i} = \left(i_1, i_2, \ldots, i_{n_p}\right)^T$.

3.2. Template Tracking based on Linear Predictors

The goal of template tracking is to follow a reference template, defined by a vector $\mathbf{i}_R$ of reference image values and an initial parameter vector $\boldsymbol{\mu}_R$, over a sequence of images. The basic approach for this is to compute a vector $\delta\mathbf{i} = \mathbf{i}_C - \mathbf{i}_R$ of image differences, where the vector $\mathbf{i}_C$ stores the image values extracted from the current image. This vector is then used to estimate a vector of parameter differences $\delta\boldsymbol{\mu}$ used to update the current template parameters $\boldsymbol{\mu}$ such that the position of the template within the current image is optimized.

Instead of explicitly minimizing an error function, *e.g.* by iteratively solving a first- or second-order approximation of it, Jurie and Dhome [12] use a learned matrix $\mathbf{A}$ to compute $\delta\boldsymbol{\mu}$ based on the vector $\delta\mathbf{i}$ as:

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

Here, the matrix $\mathbf{A}$ can be seen as a linear predictor. In order to learn $\mathbf{A}$ we apply a set of n_t random transformations to the initial template. This is done by applying small disturbances $\delta\boldsymbol{\mu}_i, i = 1, \ldots, n_t$, to the reference parameter vector $\boldsymbol{\mu}_R$. Then, each of these transformations is used to warp the sample points in order to obtain the corresponding vectors $\mathbf{i}_i$ of image values. The image value vector $\mathbf{i}_R$, obtained using the reference parameters $\boldsymbol{\mu}_R$, is used to compute the image difference vectors $\delta\mathbf{i}_i = \mathbf{i}_i - \mathbf{i}_R$ for each of the random transformations. These vectors of parameters and image differences are combined in the matrices $\mathbf{Y} = \left(\delta\boldsymbol{\mu}_1, \ldots, \delta\boldsymbol{\mu}_{n_t}\right)$ and $\mathbf{H} = \left(\delta\mathbf{i}_1, \ldots, \delta\mathbf{i}_{n_t}\right)$. In general, n_t is chosen such that it is much bigger than n_p. Using these matrices Equ. 1 can be written as $\mathbf{Y} = \mathbf{AH}$. Finally, the matrix $\mathbf{A}$ is learned using

$$\mathbf{A} = \mathbf{Y}\mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T\right)^{-1}. \tag{2}$$

In practice, we normalize the extracted image data with zero mean and unit standard deviation, which increases the robustness against illumination changes. In order to prevent $\mathbf{H}\mathbf{H}^T$ from being rank deficient we add random noise to the obtained image value difference vectors. Additionally, we apply a multi-predictor approach, where multiple linear predictors $\mathbf{A}_1, \ldots, \mathbf{A}_{n_l}$ are learned for one template, with n_l being the number of predictors per template. Thereby, the first linear predictor $\mathbf{A}_1$ is learned for large motions $\delta\boldsymbol{\mu}_i$ and the following predictors are learned for subsequently smaller motions. During tracking we iteratively apply the linear predictors. Additionally, every predictor is used multiple times. Within this paper we use five different predictors per template and three iterations for each of the predictors.

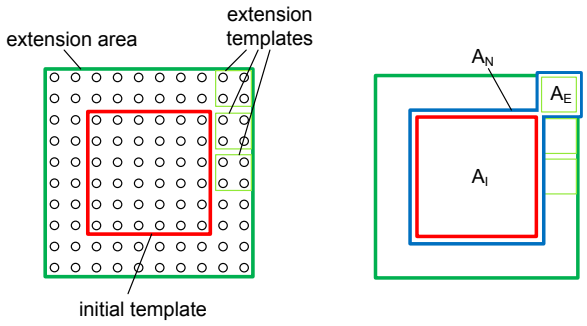


Figure 3. **(a)** The initial template together with possible extension templates defined by the corresponding extension area. **(b)** Different template areas and their corresponding linear predictors. The red border defines the initial template with its predictor $\mathbf{A}_I$, the light green border defines an extension template with its predictor $\mathbf{A}_E$ and the blue border defines the new extended template with its predictor $\mathbf{A}_N$.

4. Template Adaption

In this section we describe our approach for adapting the template by extending or reducing its size. This enables to start tracking with a small-sized template and grow or shrink it over time, automatically adapting its size and corresponding linear predictor according to the tracked scene.

4.1. Template Extension

In the following we denote the linear predictor of an initial template with $\mathbf{A}_I$, and the linear predictor of an extension template with $\mathbf{A}_E$ as depicted in Fig. 3. Using the standard approach of Sec. 3.2, the separate predictors would be

learned as:

$$\mathbf{A}_I = \mathbf{Y}\mathbf{H}_I^T \left(\mathbf{H}_I\mathbf{H}_I^T\right)^{-1} \text{ and} \quad (3)$$

$$\mathbf{A}_E = \mathbf{Y}\mathbf{H}_E^T \left(\mathbf{H}_E\mathbf{H}_E^T\right)^{-1}, \quad (4)$$

where $\mathbf{Y}$ stores the same random transformations for both linear predictors. The standard approach for learning a combined predictor $\mathbf{A}_N$ for the entire template leads to:

$$\mathbf{A}_N = \mathbf{Y}\mathbf{H}_N^T \left(\mathbf{H}_N\mathbf{H}_N^T\right)^{-1} \quad (5)$$

$$= \mathbf{Y}\begin{bmatrix} \mathbf{H}_I \\ \mathbf{H}_E \end{bmatrix}^T \left(\begin{bmatrix} \mathbf{H}_I \\ \mathbf{H}_E \end{bmatrix}\begin{bmatrix} \mathbf{H}_I \\ \mathbf{H}_E \end{bmatrix}^T\right)^{-1} \quad (6)$$

$$= \mathbf{Y}\begin{bmatrix} \mathbf{H}_I \\ \mathbf{H}_E \end{bmatrix}^T \left(\begin{bmatrix} \mathbf{H}_I\mathbf{H}_I^T & \mathbf{H}_I\mathbf{H}_E^T \\ \mathbf{H}_E\mathbf{H}_I^T & \mathbf{H}_E\mathbf{H}_E^T \end{bmatrix}\right)^{-1}. \quad (7)$$

Now, instead of directly updating the old linear predictor $\mathbf{A}_I$ we will update the matrix $\mathbf{S}_I = \left(\mathbf{H}_I\mathbf{H}_I^T\right)^{-1}$ using the formulas presented by Henderson and Searle [10], such that we obtain the matrix $\mathbf{S}_N = \left(\mathbf{H}_N\mathbf{H}_N^T\right)^{-1}$. Let $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$ and $\mathbf{S}_{22}$ be the four sub-matrices of $\mathbf{S}_N$:

$$\mathbf{S}_N = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{H}_I\mathbf{H}_I^T & \mathbf{H}_I\mathbf{H}_E^T \\ \mathbf{H}_E\mathbf{H}_I^T & \mathbf{H}_E\mathbf{H}_E^T \end{bmatrix}\right)^{-1}. \quad (8)$$

Then, we can update $\mathbf{S}_I$ to $\mathbf{S}_N$ using

$$\mathbf{S}_{11} = (\mathbf{H}_I\mathbf{H}_I^T)^{-1} + (\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\mathbf{S}_{22}\mathbf{H}_E\mathbf{H}_I^T(\mathbf{H}_I\mathbf{H}_I^T)^{-T} \quad (9)$$

$$\mathbf{S}_{12} = -(\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\mathbf{S}_{22}, \quad (10)$$

$$\mathbf{S}_{21} = \mathbf{S}_{12}^T, \quad (11)$$

$$\mathbf{S}_{22} = \left(\mathbf{H}_E\mathbf{H}_E^T - \mathbf{H}_E\mathbf{H}_I^T(\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\right)^{-1}, \quad (12)$$

where $(\mathbf{H}_I\mathbf{H}_I)^{-1}$ is known from the learning of the initial predictor. Therefore, the only inversion that has to be applied is for the computation of $\mathbf{S}_{22}$. However, this inversion is not a problem since the extension templates are always of smaller size than the entire extended template and therefore $\mathbf{S}_{22}$ is small, as well.

The approach as presented up to now is limited by the number of random transformations n_t, used for learning. Since n_t has to be the same for all extension templates as well as for the initial template, and since the number of random transformations has to be greater or at least equal to the number of used sample points, $n_t \geq n_p$, the maximum number of random transformations has to be known a priori. In order to remove this restriction we use the approach presented by Hinterstoisser *et al*. [11], which allows to update the matrix $\mathbf{S}_I$ in a way such that we can increase the number of random transformations n_t without the necessity to recompute the updated $\hat{\mathbf{S}}_I$ from scratch. This is done by using the Sherman-Morrison formula:

$$\hat{\mathbf{S}}_I = \left(\mathbf{S}_I^{-1} + \delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^T\right)^{-1} \quad (13)$$

$$= \mathbf{S}_I - \frac{\mathbf{S}_I\delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^T\mathbf{S}_I}{1 + \delta\mathbf{i}_{n_t+1}^T\mathbf{S}_I\delta\mathbf{i}_{n_t+1}}, \quad (14)$$

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. In practice, the number of random transformations is increased each time before a new extension template is added.

4.2. Template Reduction

In case that already learned templates have to be reduced, *e.g*. due to the presence of non-planarity or shortcoming for tracking, the corresponding linear predictors can be computed by updating the linear predictor of the larger template. For this, we denote the linear predictor of the large template with $\mathbf{A}_L$, the predictor of the new reduction template with $\mathbf{A}_R$ and the predictor of the reduced template with $\mathbf{A}_N$.

In order to reduce the matrix $\mathbf{S}_L$, it has to be rearranged first, such that the data corresponding to the reduction template is positioned in the last rows and columns of $\mathbf{S}_L$. After the rearrangement, the reduction template can be removed using the following approach. First, let us consider the sub-matrices of the matrix $\mathbf{S}_L$:

$$\mathbf{S}_L = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{H}_N\mathbf{H}_N^T & \mathbf{H}_N\mathbf{H}_R^T \\ \mathbf{H}_R\mathbf{H}_N^T & \mathbf{H}_R\mathbf{H}_R^T \end{bmatrix}\right)^{-1}, \quad (15)$$

where all the sub-matrices $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$, $\mathbf{S}_{22}$, $\mathbf{H}_N\mathbf{H}_N^T$, $\mathbf{H}_N\mathbf{H}_R^T$, $\mathbf{H}_R\mathbf{H}_N^T$ and $\mathbf{H}_R\mathbf{H}_R^T$ are available from the large template. The goal is to compute

$$\mathbf{A}_N = \mathbf{Y}\mathbf{H}_N^T \left(\mathbf{H}_N\mathbf{H}_N^T\right)^{-1} \quad (16)$$

without the need of inverting $\mathbf{H}_N\mathbf{H}_N^T$, since this is a large matrix in general. Similar to the Equations 9-12 Henderson and Searle [10] also presents the formula

$$\mathbf{S}_{11} = \left(\mathbf{H}_N\mathbf{H}_N^T - \mathbf{H}_N\mathbf{H}_R^T(\mathbf{H}_R\mathbf{H}_R^T)^{-1}\mathbf{H}_R\mathbf{H}_N^T\right)^{-1}, \quad (17)$$

which can be reformulated as

$$\mathbf{H}_N\mathbf{H}_N^T = \mathbf{S}_{11}^{-1} + \mathbf{H}_N\mathbf{H}_R^T(\mathbf{H}_R\mathbf{H}_R^T)^{-1}\mathbf{H}_R\mathbf{H}_N^T. \quad (18)$$

Taking the inverse leads to the desired result:

$$(\mathbf{H}_N\mathbf{H}_N^T)^{-1} = \left(\mathbf{S}_{11}^{-1} + \mathbf{H}_N\mathbf{H}_R^T(\mathbf{H}_R\mathbf{H}_R^T)^{-1}\mathbf{H}_R\mathbf{H}_N^T\right)^{-1}. \quad (19)$$

Since we, however, have to invert a big matrix in this case, namely $\mathbf{S}_{11}$, this is not suitable for online computation. Therefore, we use the following formula presented in [10]:

$$(\mathbf{X} + \mathbf{U}\mathbf{Y}\mathbf{U}^T)^{-1} = \mathbf{X}^{-1} - \mathbf{X}^{-1}\mathbf{U}\mathbf{Z}\mathbf{U}^T\mathbf{X}^{-1}, \quad (20)$$

$$\mathbf{Z} = \left(\mathbf{Y}^{-1} + \mathbf{U}^T\mathbf{X}^{-1}\mathbf{U}\right)^{-1}. \quad (21)$$

By setting $\mathbf{X} = \mathbf{S}_{11}^{-1}$, $\mathbf{Y} = (\mathbf{H}_R\mathbf{H}_R^T)^{-1}$ and $\mathbf{U} = \mathbf{H}_N\mathbf{H}_R^T$ we obtain our desired result:

$$(\mathbf{H}_N\mathbf{H}_N^T)^{-1} = \mathbf{S}_{11} - \mathbf{S}_{11}\mathbf{H}_N\mathbf{H}_R^T\mathbf{D}\mathbf{H}_R\mathbf{H}_N^T\mathbf{S}_{11}, \quad (22)$$

$$\mathbf{D} = \left(\mathbf{H}_R\mathbf{H}_R^T + \mathbf{H}_R\mathbf{H}_N^T\mathbf{S}_{11}\mathbf{H}_N\mathbf{H}_R^T\right)^{-1} \quad (23)$$

Now, the necessary inversion is no longer a problem since the reduction template is chosen to be of small size and computing $\mathbf{D}$ is not expensive.

4.3. Practical Issues

In this section we discuss practical issues. These are the normalization of the image data and the estimation of the subset quality, which is used for the selection of the next extension template.

4.3.1 Normalization

As mentioned before, the image values are normalized to zero mean and unit standard deviation. However, instead of doing this globally by considering all image values of the template we apply a local normalization, where each subset is normalized by considering only its image values and the image values of its direct local neighboring subsets. This normalization is applied to the reference data, the learning data and the current image data during tracking. The local normalization is superior to the global normalization since in case of the global normalization the mean and standard deviation of the whole image data change if new parts are added to the template or some parts are removed.

4.3.2 Suitability Criterion for Subset Selection

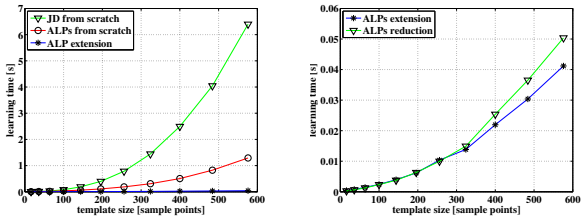
In order to decide which subset should be chosen for extending the current template we compute a quality measure for each of the potential extension templates in the local neighborhood of the current template. This is done by learning a local predictor $\mathbf{A}_\mathrm{S} = \mathbf{Y}_\mathrm{S}\mathbf{H}_\mathrm{S}^T(\mathbf{H}_\mathrm{S}\mathbf{H}_\mathrm{S}^T)^{-1}$ for this subset at first, where the image data $\mathbf{H}_S$ is collected using the set of random transformations represented by $\mathbf{Y}$. Then, using this predictor together with the collected image data we compute a prediction $\hat{\mathbf{Y}}_\mathrm{S}$ of $\mathbf{Y}$ as

$$\hat{\mathbf{Y}}_\mathrm{S} = \mathbf{A}_\mathrm{S}\mathbf{H}_\mathrm{S}. \quad (24)$$

Finally, we compute a similarity measurement, which defines the quality q_S of the corresponding subset as

$$q_\mathrm{s} = \frac{1}{n_t}\sum_{i=1}^{n_t}\frac{\hat{\mathbf{y}}_{\mathrm{s}i}\mathbf{y}_i^T}{|\hat{\mathbf{y}}_{\mathrm{s}i}||\mathbf{y}_i|}, \quad (25)$$

where $\mathbf{y}_i$ and $\hat{\mathbf{y}}_{\mathrm{s}i}$ are the i-th column vector of $\mathbf{Y}$ respectively $\hat{\mathbf{Y}}_\mathrm{S}$. The current template will then be extended using the subset with the highest quality measure.



(a) (b)

Figure 4. Comparison of the computation time necessary for learning a linear predictor using Jurie-Dhome approach (green) and using ALPs (red and blue). **(a)** For the later case we distinguish between learning the predictor from scratch (red) and adding only one extension subset (blue) at a time. Learning from scratch means that we consider the whole time necessary to build up the template of the specified size. **(b)** Computation times for template extension and reduction, when one extension subset is added at a time.

5. Experimental Results

In this section we perform extensive comparison of our approach with several state of the art approaches on template tracking. This includes comparisons with the standard learning approach of Jurie and Dhome [12], the analytical approach of Benhimane and Malis [4] and a recent approach called NoSLLip of Zimmermann *et al*. [20]. The comparisons are done in terms of tracking precision and computational efficiency. In the end we show several qualitative results from real video sequences showing tracking results with one and several templates. All experiments are performed on a 2.66 GHz Intel(R) Core(TM)2 Quad CPU with 8 GB of RAM, where only one core is used for the computations.

In all experiments the maximum random perturbation applied for learning the linear predictors is set to 21 pixels except for the comparison with NoSLLip, where we slightly increased the perturbation by 10% to make the tracking more robust against large motions.

5.1. Comparison with Jurie-Dhome Approach

Computational Complexity of Learning In Fig. 4 we show computation times for learning the linear predictors with respect to different template sizes. We compare our ALPs method, shown in red and blue, with the standard approach of Jurie and Dhome [12], depicted as green curve in Fig. 4(a). For our approach we distinguish between two cases. In the first case, shown as a red curve, the computation of the linear predictor is done iteratively from scratch. In that case we start with a small initial template, whose size is equal to the size of an extension template of Fig. 3. Such a small template is then grown until the specified size is reached. The obtained results reveal clearly that the adaptive learning of the linear predictor, which starts with the small sized template, is much more efficient than learning a linear predictor for the fixed size template. This proves

that our approach can also be used to efficiently learn linear predictors for templates of fixed size, starting from small templates and adapting their linear predictors until the desired template size is reached. In the second case, shown as a blue curve, we show the time necessary to add one extension template. This is a typical case during online tracking, where the template is grown step by step. As to be expected, adding the extension template does not significantly increase computation time, when changing the template size. In Fig. 4 (b) we show computation times for extension and reduction of templates. Note that the necessary time to grow or reduce the template by an extension template consisting of four sample points is around 0.05s for initial templates of sizes around 600 sample points, whereas the computation from scratch would need over 1s using ALPs and more than 6s when using the approach of Jurie and Dhome [12].

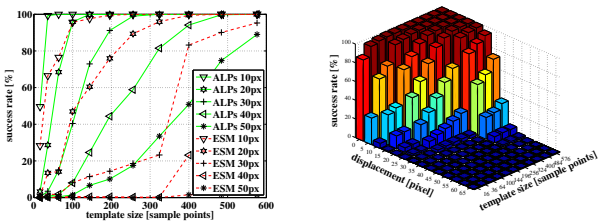
Robustness To evaluate the robustness of our approach we compare the tracking success rate of our approach with that of the standard approach proposed by Jurie and Dhome [12] for different template sizes and with respect to changes in translation (Fig. 7 (a)), in-plane rotation (Fig. 7 (b)), viewing angle (Fig. 7 (c)), and scale (Fig. 7 (d)). In addition we compare ALPs to Jurie and Dhome in respect to noise and different number of random transformations used for learning. The results are shown in Fig. 6.

For all experiments, we use synthetic images, corrupted by noise and warped according to the specific experiments. Noise is added according to $I_n(\mathbf{x}) = I(\mathbf{x}) + \epsilon$, with $\epsilon \in [-\alpha I_{\text{range}}/100, \alpha I_{\text{range}}/100]$, and $\alpha = 5$ for all experiments. An exception is the noise experiment, where different levels of noise were applied. I_{range} specifies the possible range of image values, *e.g.* $I_{\text{range}} = 255$ holds for image values between 0 and 255. In all experiments we also add a random displacement in the range of $[-5, 5]$ pixels, with the exception of the displacement experiments, and a random change in the view-point angle ranging between $[-5, 5]$ degree, again with the exception of the view-point angle experiments.

The results show that both approaches, the standard Jurie-Dhome approach as well as ALPs, yield similar success rates. The only exception is the sensitivity to noise, where the Jurie-Dhome approach performs better than ALPs. This performance difference, however, can be reduced by increasing the number of random warpings used for learning the linear predictors, as demonstrated in Fig. 6 (b).

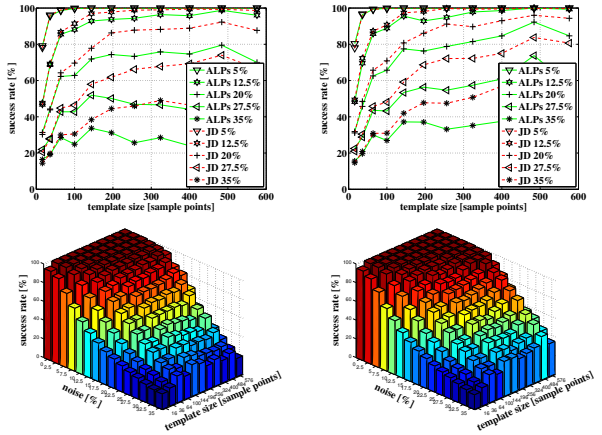
5.2. Comparison with ESM

To demonstrate the usefulness of learning-based approaches we compare our approach with the analytical method of Benhimane and Malis [4]. For this, we have



(a) (b)

Figure 5. Comparison of success rates with respect to different displacements and template sizes. **(a)** Performance of ESM vs. ALPs. **(b)** Performance of ESM for further displacements.



(a) (b)

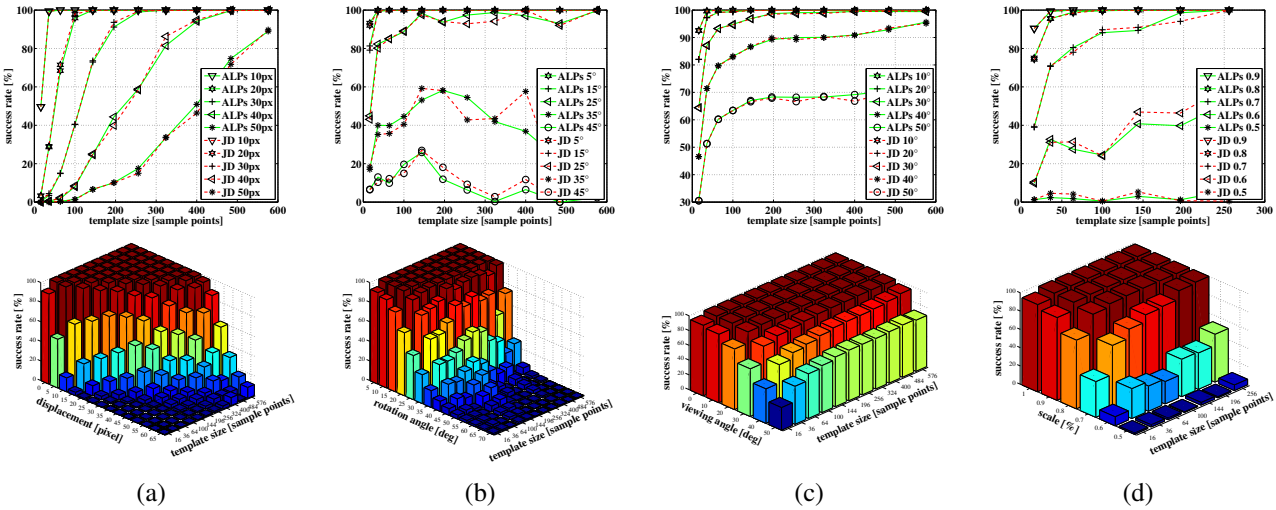
Figure 6. Comparison of success rates of ALPs and linear predictors of Jurie and Dhome(JD) with respect to different levels of noise and different template sizes. **(a)** Success rates of ALPs using 1000 random warpings and **(b)** using 2000 random warpings.

chosen ESM, a state of the art approach that minimizes the energy function using a second order approximation. In Fig. 5 we compare the success rate of the ESM tracking to that of ALPs regarding different magnitudes of displacements and different template sizes. Our learning-based approach clearly outperforms ESM, especially for larger template sizes.

5.3. Comparison with NoSLLiP

We also compare ALPs to the approach of Zimmermann *et al.* [20] using the phone sequence provided by the authors[1]. Example images of the tracking are shown in Fig. 8. The comparison between the tracking results of [20] and those obtained using ALPs are shown in Table 1. Although the number of provided images is larger than the number of images used by Zimmermann *et al.* [20], we still obtain a better loss-of-locks count. The given error values are rela-

[1]Zimmermann *et al.* [20] provide three different video sequences. One of them, however, includes occlusion, which can not yet be handled by our approach, and for another one the supplied ground truth data is erroneous. Therefore, we compare only to one of the provided sequences.



(a) (b) (c) (d)

Figure 7. Comparison of success rate of ALPs and JD linear predictors with respect to changes in translation **(a)**, in-plane rotation **(b)**, viewing angle **(c)**, and scale **(d)**. In all four cases the results of both approaches are approximately equal. The lower row shows more detailed results of ALPs.

Method	Frame-rate [fps]	Loss-of-locks [-/-]	Error [%]
NoSLLiP	16.8	20/1799	1.8
ALPs	96.7	10/2299	1.2

Table 1. Comparison between the tracking results of NoSLLiP (Matlab implementation) given in [20] and results obtained using ALPs (C++ implementation).

tive to the upper edge of the template. A frame is counted as loss-of-lock if one of the template corners has an error larger than 25%. Note that the template is reduced when it partially goes out of sight and enlarged again as it becomes visible again (see Fig. 8).

5.4. Usefulness of larger templates

As shown in Figures 5, 6 and 7, the success rates increase with increasing template sizes. The only exception are changes in the in-plane rotation angle, where the success rate reaches a maximum for templates with a size of approximately 100 to 200 sample points.

5.5. Qualitative Evaluation

In Fig. 1, 8 and 9 we show different image sequences, which demonstrate the processing of the proposed approach. In Fig. 1 and 9 we start with templates of size 10 by 10 sample points and iteratively grow them by adding the neighboring extension template with the highest quality. In Fig. 9 we demonstrate the use of multiple templates. In Figures 1 and 8 we track the templates, reduce them if they partially go out of sight and grow them back to the original size when their hidden parts become visible again.

6. Conclusion and Future Work

We introduced an efficient method for adapting linear predictors used in real-time template tracking to dynamically change the template shape. Our method allows both, enlargement and reduction of the template size. We demonstrated that our ALPs approach can also be used to efficiently learn linear predictors for templates of fixed size. In that case we start from small templates and adapt their linear predictors until the desired template size is reached. This resulted in much shorter learning time compared to the standard approach of Jurie and Dhome [12]. The efficiency of the presented approach derives from the special computation of the matrix inverse. In the standard approach the inverse has to be recomputed from scratch after each change of the template size. In contrast, our approach updates the matrix according to the change in the template shape.

We demonstrated that our ALPs yield tracking results comparable to those of the standard approaches, while learning is much faster. The current approach, however, lacks robustness against occlusion. Therefore, the next step will be to provide some means of occlusion handling for large templates.

Acknowledgments

We want to thank Stefan Hinterstoisser and Jürgen Sotke for prove-reading the paper. The project was partially funded by the Bayerische Forschungsstiftung.

References

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Conference on Computer*

Figure 8. Result images of the phone sequence, which is provided by Zimmermann *et al*. [20]. Note that the template is reduced if it goes out of the image and grown again of it gets visible again.

Figure 9. Iterative growing of two independent templates.

Vision and Pattern Recognition, volume 1, page 1090, Los Alamitos, CA, USA, 2001.

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, March 2004.

[3] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *Conference on Intelligent Robots and Systems*, volume 1, pages 943–948 vol.1, Sept.-2 Oct. 2004.

[4] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *International Journal of Robotics Research*, 26(7):661–676, July 2007.

[5] M. Cascia, S. Sclaroff, and V. Athitsos. 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*, 22(4), April 2000.

[6] F. Dellaert and R. Collins. Fast image-based tracking by selective pixel integration. In *ICCV Workshop of Frame-Rate Vision*, pages 1–22, September 1999.

[7] C. Gräßl, T. Zinßer, and H. Niemann. Illumination insensitive template matching with hyperplanes. In *Proceedings of Pattern recognition: 25th DAGM Symposium*, pages 273–280, Magdeburg, Germany, September 2003.

[8] C. Gräßl, T. Zinßer, and H. Niemann. Efficient hyperplane tracking by intelligent region selection. In *Image Analysis and Interpretation*, pages 51–55, March 2004.

[9] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[10] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981.

[11] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online learning of patch perspective rectification for efficient object detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, Alaska, 2008.

[12] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–100, 2002.

[13] F. Jurie and M. Dhome. Real time robust template matching. In *British Machine Vision Conference*, pages 123–131, 2002.

[14] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[15] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1843–1848 Vol.2, 26-May 1, 2004.

[16] J. Matas, K. Zimmermann, T. Svoboda, and A. Hilton. Learning efficient linear predictors for motion estimation. In *Computer Vision, Graphics and Image Processing*, pages 445–456, 2006.

[17] W. W. Mayol and D. W. Murray. Tracking with general regression. *Journal of Machine Vision and Applications*, 19(1):65–72, 2008.

[18] P. Parisot, B. Thiesse, and V. Charvillat. Selection of reliable features subsets for appearance-based tracking. *Signal-Image Technologies and Internet-Based System*, 0:891–898, 2007.

[19] H.-Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, pages 101–130, 2000.

[20] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):677–692, 2009.