Multi-Layer Adaptive Linear Predictors for Real-Time Tracking

Stefan Holzer, *Student Member, IEEE,* Slobodan Ilic, *Member, IEEE,* and Nassir Navab, *Member, IEEE*

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 a 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. Additionally, we propose a multi-layer approach to detect occlusions and use ALPs to effectively handle them. This allows us to track large templates and modify them according to the present occlusions.
We performed exhaustive evaluation of our approach and compared it to standard linear predictors and other state of the art approaches.

Index Terms—Template tracking, linear predictors.

1 Introduction

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

While there are many template tracking approaches based on the analytical derivation of the Jacobian [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], learning-based methods [11], [12], [13], [14], [15], [16], [17] have proved to be faster and generally more robust with respect to large perspective changes.

A very successful learning-based template tracker was proposed by Jurie and Dhome [11]. It is based on the learning of linear predictor to efficiently compute template parameter updates. The costly off-line learning phase, however, prohibits this method from computing templates of varying sizes online.

Yet, the ability to dynamically change the template size is necessary in many applications. In indoor SLAM, *e.g.*, the 3D geometry of the scene is a priori unknown making 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.

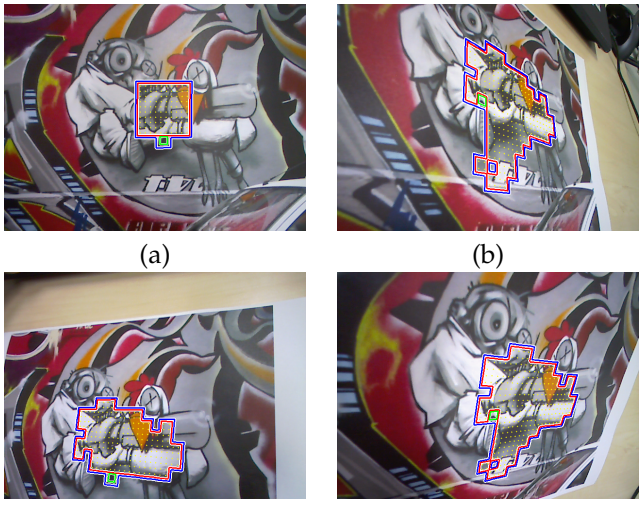


Fig. 1. (a) A small initial template is (b) enlarged according to a tracking quality measure. The template is tracked over time and (c) reduced if parts of it go out of sight. The removed parts are reinserted (d) as soon as they become visible again.

Current learning-based tracking approaches, like [11], use templates of a 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 effort for changing the template size is nearly equivalent to that of learning a new template from scratch. Hence, to overcome the limitations of fixed size template approaches, while maintaining their robustness to large

• *S. Holzer, S. Ilic and N. Navab are with the Department of Computer Science, Technical University of Munich (TUM), Boltzmannstrasse 3, 85748 Garching, Germany.*
E-mail: {holzers,slobodan.ilic,navab}@in.tum.de

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 present a computationally efficient way of updating the inverse which dramatically reduces the time needed for learning. We start with a small initial template and enlarge it by small extension templates, as shown in Fig. 4, according to their suitability for tracking. As long as the object to track is planar, our approach can expand the template in any direction, resulting 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.

Moreover, the ability to shrink and grow templates also enables us to handle occlusions. This, however, requires to detect occlusions in the first place, since it is usually barely possible to distuinguish between errors caused by occlusions and those caused by motions. We use an occlusion detection technique based on multiple layers of differently sized templates. In contrast to previous approaches [12], [17] which track all templates simultaneously, we track them sequentially. This means that the results of the tracking at higher layers are used to initialize the tracking of smaller templates at lower layers. If due to occlusion tracking at some layer fails, the smaller templates at the next lower layer are used for tracking. The tracking failure of some templates at the lowest layer indicates occlusion and corresponding regions are removed from the largest template of the top layer. This allows us to benefit from the stable tracking of a large template while using smaller templates for occlusion detection, which guide the change of the shape of the large template.

To further maximize the robustness of the template tracking with respect to large motion in case of occlusions, we introduce the concept of observed and insecure regions. These regions are used to detect incoming occlusions before they influence the tracking process. Finally, to achieve high tracking robustness we track a number of large, shifted templates at the top layer in parallel fashion.

We perform extensive quantitative evaluations and compare our approach to the standard approach of Jurie and Dhome [11] under different transformations and noise levels and with other state-of-the-art template tracking approaches [10], [17]. We demonstrate that our approach performs better or equally well with respect to the related approaches, while requiring much shorter learning times when templates are extended and shrinked. In case of occlusions, where we explicitly detect and handle them, it is superior to the other related works. In the remainder of the paper we will discuss related work on template tracking, give a detailed description of our approach on template extension and reduction, introduce our approach for occlusion detection, present our results and show examples on real world sequences.

2 Related Work

Since the seminal work of Lucas and Kanade [1], many efforts have been made in the field of template tracking and image alignment. Most of the presented approaches can be put into one of two categories: template tracking based on the analytical derivation of the Jacobian [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] or based on learning [11], [12], [13], [14], [15], [16], [17]. While analytical approaches are generally more flexible regarding template shape modifications at run-time, learning approaches enable higher tracking speed and are more robust in terms of large perspective changes.

A large variety of analytical tracking approaches have been presented since the work of Lucas and Kanade [1]. Amongst others, the variations in the presented analytical tracking approaches include different update rules of the warp [1], [3], [4], [2], [5], [6], different orders of approximations of the error function [8], [9], [10], occlusion [3], [12], [16], [17], [18], [19] and illumination change handling [3]. Basically, there are four different types of update rules, the additive approach [1], the compositional approach [2], the inverse additive approach [3], [4], and the inverse compositional approach [5], [6]. In the latter two, the roles of the reference and current image are switched, which allows to do some of the computations during an initialization phase, making the tracking computationally very efficient. Faster convergence rate can additionally be obtained by using a second-order instead of a first order approximation of the error function [8], [9], [10]. Furthermore, Hager and Belhumeur [3] showed how illumination changes and occlusions can be efficiently handled using iterativly reweighted least squares for occlusion handling. The weights are determined using a robust error function treating large image intensity differences as occlusions. Baker et. al [18] describe how such robust error functions can be incorporated into the inverse compositional template tracking framework. However, since large motion as well as occlusions generally result in large image intensity differences it is hardly possible to distinguish between them. For a more detailed overview of analytical tracking methods refer to Baker and Matthews [7].

In contrast to analytical tracking methods, Jurie and Dhome [11] proposed an approach for the learning of linear predictor using randomly warped samples of the initial template. The linear predictors are then used to predict the parameter updates during tracking. This allows very fast tracking, since the ”Jacobians” are initially computed once and for all and the parameter updates can be obtained by simple matrix vector multiplications. In [12] the authors also extend the approach in order to handle occlusion. This extension uses multiple layers of differently sized trackers, which are applied in parallel fashion. All available trackers then vote for their resulting pose change and the final pose change is found by iteratively subdividing the space of possible pose changes

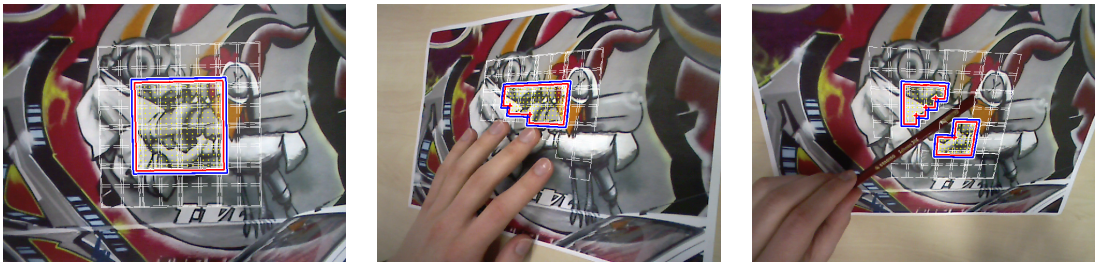


Fig. 2. Shows the tracking and adaption of a template according to present occlusions.

and selecting the N best sub-spaces. This is repeated until the resulting sub-spaces are sufficiently small so that they can be considered as a single pose change. However, as soon as occlusion occurs and tracking the largest template fails the robustness of the tracking is reduced. Gräßl *et al*. [20] show how the robustness of the linear-predictor based approach can be further increased with regard to illumination changes. In [13] they also present an intelligent way of selecting the points for sampling the image data, in order to increase the tracking accuracy. Another linear predictor approach [17] describes templates consisting of many small templates and tracks these small templates independently. The approach uses one layer of small-sized trackers, which allows for the computation of independent pose updates. A common pose is computed using RANSAC, which makes the approach robust against occlusions. However, due to the small size of the trackers this approach is less robust against large motion. Instead of using linear predictors, Mayol and Murray [16] present an approach that fits the sampling region to pre-trained samples using general regression. In order to increase the robustness against occlusions they use two filters. The first filter uses a weighted scheme in order to define whether the current scene fits the training samples or not. In case that it does not fit, it is considered as an occlusion. The second filter prohibits using the updated parameters from one iteration, if the obtained image value error increases. In both cases, the parameters from the previous frame iteration are kept. Therefore, this only works if the tracked object does not significantly move while occlusion is present. Patras and Hancock [19] use a particle filter approach that samples multiple observations and estimates the relevance of each of these observations. For each of the observations considered relevant they then use Bayesian Mixtures of Experts to compute a probabilistic prediction of the new pose. In that way, observations which include occlusions are avoided.

All of the proposed learning approaches, do not deal with templates of variable sizes. To overcome this limitation we developed a method that extends the approach of Jurie and Dhome [11] to allow online template size adaptation. We also propose a multi-layer approach for occlusion detection which, by contrast to other approaches based on linear predictors, is stable and precise. This is achieved by using relatively large templates for tracking, compared to other approaches which rely on tracking multiple small templates. Tracking small-sized templates is sensitive to large motion and such motion can be confused with occlusions. We benefit from the increased robustness against large motion and successfully handle detected occlusions by employing the proposed method for fast on-line adaption of the template size.

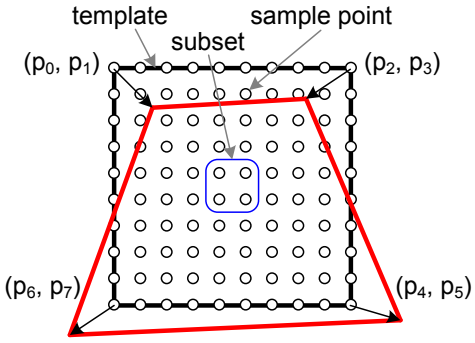


Fig. 3. A template is represented by a set of regularly placed sample points, which are grouped into subsets of four points. The pose of a template is parameterized using four corner points.

3 Background and Terminology

In this section we introduce our notation and, for the sake of completeness, review the original template tracking approach proposed by Jurie and Dhome [11].

3.1 Template and Parameter Description

A template consists of a set of n_p sample points, which are distributed over 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. 3. 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 Fig. 3. 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 [11] 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 $\mathbf{i} = (i_1, i_2, \ldots, i_{n_p})^T$.

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 $\boldsymbol{\mu}$ so 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 [11] 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 $\mathbf{Y} = (\delta\boldsymbol{\mu}_1, \delta\boldsymbol{\mu}_2, \ldots, \delta\boldsymbol{\mu}_{n_t})$ and $\mathbf{H} = (\delta\mathbf{i}_1, \delta\mathbf{i}_2, \ldots, \delta\mathbf{i}_{n_t})$. In general, n_t is chosen so that it is much bigger than n_p. Using these matrices Eq. 1 can be written as $\mathbf{Y} = \mathbf{AH}$. Finally, the matrix $\mathbf{A}$ is learned by minimizing:

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

which results in the closed-form solution:

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

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 levels of 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 and the following predictors are learned for subsequently smaller motions.

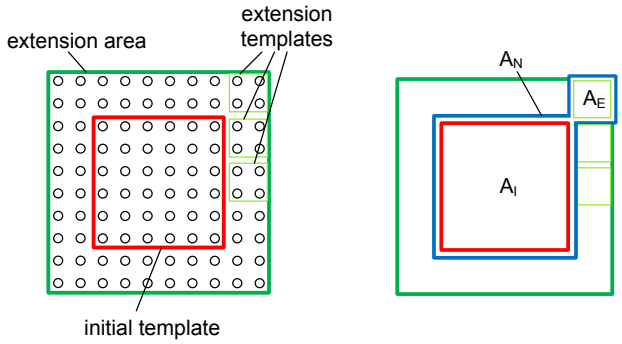


Fig. 4. **(a)** The initial template together with possible extension templates defined by the corresponding extension area. **(b)** Different template areas and their corresponding linear predictors. The red border defines the initial template with its predictor $\mathbf{A}_I$, the light green border defines an extension template with its predictor $\mathbf{A}_E$ and the blue border defines the new extended template with its predictor $\mathbf{A}_N$.

During tracking we sequentially apply the linear predictors. Additionally, every predictor is iteratively used n_i times. Within this paper we use five different levels of predictors per template and three iterations for each of the predictors. Alg. 1 formalizes the applied tracking approach.

Algorithm 1 Tracking without Occlusion Handling

```
function Track (in Image I,
    in/out TemplateParameters μ)
Compute homography T_μ from μ.
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 from T_μ.
```

4 Template Adaption

In this section we describe our approach for adapting the template by extending or reducing its size. This enables us 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. 4. Using the standard approach of Sec. 3.2, the separate predictors would be learned as:

$$\mathbf{A}_I = \mathbf{Y}\mathbf{H}_I^T \left(\mathbf{H}_I\mathbf{H}_I^T\right)^{-1} \text{ and} \quad (4)$$

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

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:

$$\mathbf{A}_N = \mathbf{Y}\mathbf{H}_N^T \left(\mathbf{H}_N\mathbf{H}_N^T\right)^{-1} \quad (6)$$

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

$$= \mathbf{Y}\begin{bmatrix} \mathbf{H}_I \\ \mathbf{H}_E \end{bmatrix}^T \left(\begin{bmatrix} \mathbf{H}_I\mathbf{H}_I^T & \mathbf{H}_I\mathbf{H}_E^T \\ \mathbf{H}_E\mathbf{H}_I^T & \mathbf{H}_E\mathbf{H}_E^T \end{bmatrix}\right)^{-1}. \quad (8)$$

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 [21], so 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 $\mathbf{S}_N$:

$$\mathbf{S}_N = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{H}_I\mathbf{H}_I^T & \mathbf{H}_I\mathbf{H}_E^T \\ \mathbf{H}_E\mathbf{H}_I^T & \mathbf{H}_E\mathbf{H}_E^T \end{bmatrix}\right)^{-1}. \quad (9)$$

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

$$\mathbf{S}_{11} = (\mathbf{H}_I\mathbf{H}_I^T)^{-1} + (\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\mathbf{S}_{22}\mathbf{H}_E\mathbf{H}_I^T(\mathbf{H}_I\mathbf{H}_I^T)^{-T} \quad (10)$$

$$\mathbf{S}_{12} = -(\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\mathbf{S}_{22}, \quad (11)$$

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

$$\mathbf{S}_{22} = \left(\mathbf{H}_E\mathbf{H}_E^T - \mathbf{H}_E\mathbf{H}_I^T(\mathbf{H}_I\mathbf{H}_I^T)^{-1}\mathbf{H}_I\mathbf{H}_E^T\right)^{-1}, \quad (13)$$

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 a smaller size than the entire extended template and therefore $\mathbf{S}_{22}$ is small as well. In case the template is already known before run-time, *i.e.* if template extension and reduction is only used to handle occlusions, the necessary inversions can even be precomputed in order to further speed up processing. In practice, we use templates for extension and reduction which consist of 4 sample points.

Note that for the computation of the image value differences $\mathbf{H}_E$ we have to use the same random transformations as used for computing $\mathbf{H}_I$. The same holds for $\mathbf{H}_R$ in case of template reduction (see Section 4.2).

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*. [22], which allows us to update the matrix $\mathbf{S}_I$ in such a way that we can increase the number of random transformations n_t without having to recompute the updated $\hat{\mathbf{S}}_I$ from scratch. This is done by using the Sherman-Morrison formula:

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

$$= \mathbf{S}_I - \frac{\mathbf{S}_I\delta\mathbf{i}_{n_t+1}\delta\mathbf{i}_{n_t+1}^T\mathbf{S}_I}{1 + \delta\mathbf{i}_{n_t+1}^T\mathbf{S}_I\delta\mathbf{i}_{n_t+1}}, \quad (15)$$

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, such that $n_t = 3n_p$.

4.2 Template Reduction

In the case when already learned templates have to be reduced, *e.g.* due to the presence of non-planarity or tracking failure, 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, so 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 submatrices of the matrix $\mathbf{S}_L$:

$$\mathbf{S}_L = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{H}_N\mathbf{H}_N^T & \mathbf{H}_N\mathbf{H}_R^T \\ \mathbf{H}_R\mathbf{H}_N^T & \mathbf{H}_R\mathbf{H}_R^T \end{bmatrix}\right)^{-1}, \quad (16)$$

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

$$\mathbf{A}_N = \mathbf{Y}\mathbf{H}_N^T \left(\mathbf{H}_N\mathbf{H}_N^T\right)^{-1} \quad (17)$$

without the need of inverting $\mathbf{H}_N\mathbf{H}_N^T$, since this is a large matrix in general. Similar to the Equations 10-13 Henderson and Searle [21] also present the formula

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

which can be reformulated as

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

Taking the inverse leads to the desired result:

$$(\mathbf{H}_N\mathbf{H}_N^T)^{-1} = \left(\mathbf{S}_{11}^{-1} + \mathbf{H}_N\mathbf{H}_R^T(\mathbf{H}_R\mathbf{H}_R^T)^{-1}\mathbf{H}_R\mathbf{H}_N^T\right)^{-1}. \quad (20)$$

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 [21]:

$$(\mathbf{X}+\mathbf{UYU}^T)^{-1} = \mathbf{X}^{-1}-\mathbf{X}^{-1}\mathbf{UZU}^T\mathbf{X}^{-1}, \quad (21)$$

$$\mathbf{Z} = \left(\mathbf{Y}^{-1}+\mathbf{U}^T\mathbf{X}^{-1}\mathbf{U}\right)^{-1}. \quad (22)$$

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:

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

$$\mathbf{D} = \left(\mathbf{H}_R\mathbf{H}_R^T+\mathbf{H}_R\mathbf{H}_N^T\mathbf{S}_{11}\mathbf{H}_N\mathbf{H}_R^T\right)^{-1} \quad (24)$$

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 of the 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 the 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 $\mathbf{A}_S=\mathbf{Y}_S\mathbf{H}_S^T(\mathbf{H}_S\mathbf{H}_S^T)^{-1}$ 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 $\hat{\mathbf{Y}}_S$ of $\mathbf{Y}$ as

$$\hat{\mathbf{Y}}_S=\mathbf{A}_S\mathbf{H}_S. \quad (25)$$

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

$$q_\mathrm{s}=\frac{1}{n_t}\sum_{i=1}^{n_t}\frac{\hat{\mathbf{y}}_{\mathrm{s}i}\mathbf{y}_i^T}{|\hat{\mathbf{y}}_{\mathrm{s}i}||\mathbf{y}_i|}, \quad (26)$$

where $\mathbf{y}_i$ and $\hat{\mathbf{y}}_{\mathrm{s}i}$ are the i-th column vector of $\mathbf{Y}$ and $\hat{\mathbf{Y}}_\mathrm{S}$ respectively. This measures the similarity of the prediction and the data used for learning by computing the mean angle between the corresponding parameter vectors. This way an extension template is chosen which ensures best that each tracking iteration brings the template parameters closer to the desired result. The current template will then be extended using the subset with the highest quality measure. The suitability criterion is only computed on a subset, which consists of 4 sample points, and not on the whole resulting template, since this would be computationally to expensive. In practice, we use the suitability criterion to select the best subtemplates within a user-defined area such that we obtain a template with a certain number of sample points, which is also defined by the user.

5 Occlusion-Aware Tracking

In order to make template tracking robust against occlusions, we detect occlusions and consider them during the computation of the template pose parameters. Since this computation depends on the image differences between the current image and the template warped according to the pose parameters of the previous frame, the difficulty is to distinguish whether these differences come from the camera/template motion or from real occlusions. To distinguish this, we propose a multi-layer approach where on the top level we use a relatively large template for tracking and on the lower levels smaller sized templates are used to detect occlusions. Tracking a large template is more stable compared to multiple smaller sized templates, thus we always track a large template while failures in tracking of small sized templates indicate presence of occlusions. We use the technique proposed in Sec. 4 to remove the occluded parts from the large top level template and add them back as soon as they are visible again. This allows us to handle complex situations, such as occlusions passing over the entire template or moving templates with a hole as demonstrated in the results section. In the remainder of this section we discuss the proposed approach. An overview over the proposed approach is given in Alg. 2.

5.1 Multi-Layer Approach

Similar to the approach of Jurie and Dhome [12] we propose to use multiple layers of linear predictor grids, where the sizes of the templates, which correspond to the linear predictors, vary over the different layers. Fig. 5(a) shows such a layered organization. The size of the templates decreases with the depth of the layer. The template of the top layer (Layer 1 in Fig. 5(a)) corresponds to the actual template size. The template is subdivided in the next level (Layer 2 in Fig. 5(a)) to four equally sized templates. Every template is further subdivided into four new templates at the next layer. In practice, we use only three layers. This is sufficient to handle occlusions. In contrast to Jurie and Dhome [12] who applied linear

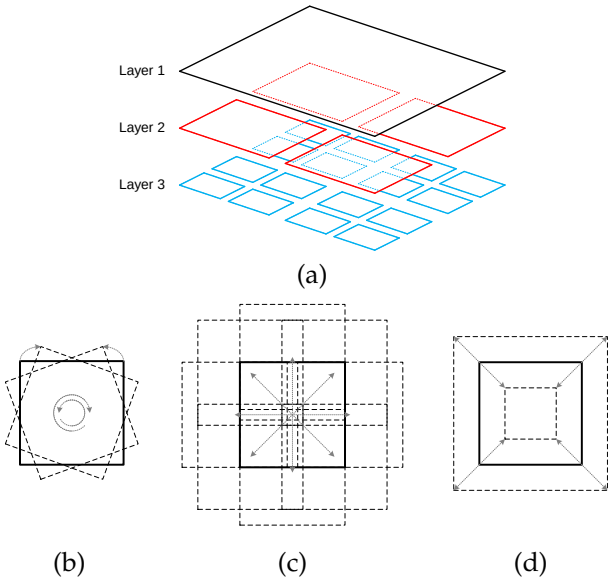


Fig. 5. **(a)** The organization of the multiple layers used for tracking. The sub-figures **(b)**, **(c)** and **(d)** show different transformed templates of the top layer used to increase the robustness against large motion. **(b)** shows differently rotated, **(c)** differently translated and **(d)** differently scaled templates.

predictors for each template at the same time, we do it sequentially. This means that in the first layer, we track a single large template. The resulting pose is then used as an initialization for the grid of trackers in the next layer. This is repeated for all available layers.

Occluded and non-occluded case. During the tracking, we can distinguish two possible scenarios. One is when there are no occlusions present and the other when occlusions intervene. If there are no occlusions, the successful tracking result at a higher level is forwarded to the next lower level. In general, the tracking of small templates can fail in the case of large motion. That is where the approach of Jurie and Dhome [12], in which all the templates at all the layers are tracked simultaneously, tends to fail. In our approach, the tracking of small templates at lower levels is preconditioned by the result of the preceding tracker layer. In the other scenario we consider that occlusions happen. In this case, depending on the size of the occluded area, the trackers at higher levels fail. Those on lower levels are then used to directly estimate the pose initialized by the resulting pose in the previous frame. The failure detection is done using a simple threshold applied on the mean image intensity differences of the normalized image data. In practice, we use different thresholds for each layer, for the first layer we use 0.03 as threshold on the mean image intensity differences of the normalized image data, 0.08 for the second layer and 0.15 for the third layer.

5.2 Combining multiple pose estimates.

Since the layers which consist of more than one template can lead to different pose parameters, an outlier rejection has to be applied, which removes erroneous results. For this, we consider the corner points of each template as single feature points and use RANSAC to robustly estimate a homography from them. Based on this, we only consider a template to be successfully tracked if its corner points are not rejected as outliers after the homography estimation. The final homography of each layer is then computed by considering the corner points of all the successfully tracked templates. In order not to replace a successful homography of a higher level by a failed tracking at a lower level, the homography is only updated if the change is not too big. The homography computed in the last layer is used for both pose estimation of the top layer template as well as for occlusion detection.

In order to increase robustness in the case of occlusions and large motion, we adapt the template size of the top layer using our approach discussed in the previous section. However, this has to be done before an occlusion occurs. Therefore, we introduce the concept of observed and insecure regions.

5.3 Observed and Insecure Regions

5.3.1 Observed Regions

The observed regions are placed within an area around the top layer template, as shown in Fig. 6, similar to the extension area in [23]. Here, these regions are used for early detection of incoming occlusions. In order to detect occlusions within the observed regions, we add corresponding trackers to the lower layers so that they cover these regions as depicted in Fig. 6. Then, these additional trackers are used together with the other trackers of the specific layer for multi-layer tracking. Successful trackers are considered in the RANSAC-process described in Sec. 5.1.

Detecting occlusions. After estimating the global pose using the multi-layer approach, we re-evaluate the image intensity differences of the lowest-layer templates as well as of the small subsets of the top layer template using the corresponding mean image intensity differences. These errors are then used to decide whether an occlusion is present or not. In practice, we use a value of 0.2 as threshold on the mean image intensity difference to decide whether a subset is occluded or not. In order to reduce the impact of noise we only consider subsets to be occluded if they are not covered by a successfully tracked template at one of the layers.

5.3.2 Insecure Regions

Knowing the positions of present occlusions, we define insecure regions around these occlusions as shown in Fig. 6. These insecure regions are considered as having a high chance of becoming occluded in the next frame. Therefore, the occlusions as well as the corresponding insecure regions are removed from the top layer template using techniques outlined in Sec. 4 as described below.

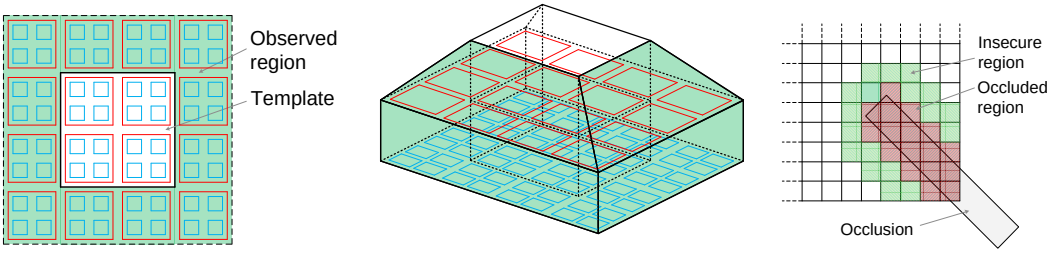


Fig. 6. The left figure shows the multi-layered template with its observation region depicted as green area. The middle figure shows the different layers and their contribution to the observed region from a side-view. The right figure illustrates insecure regions, which are areas around the detected occlusions.

5.4 Template Adaption

Once the occlusions and their corresponding insecure regions are detected, the shape of the top level template T_0 and its corresponding linear predictors are adapted. This means that as soon as a subset of T_0 is covered by an occluded part or a insecure region, it is removed from T_0. When the occlusion disappears and no longer covers the previously occluded parts of the template, they are added back to the template T_0. This enables us to continuously use a large template for tracking even in the case of occlusion and therefore, to maintain robustness against large motion.

5.5 Increased Robustness against Large Motions

To increase the tracking robustness in case of large motions we introduce multiple transformed versions of the template at the top layer (see Figures 5(b), (c) and (d)), which are tracked in parallel. For this, we use the same linear predictors for each of the transformed templates. However, since the multi-predictor approach as stated in Sec. 3.2 would be computationally expensive in this case we use only the first linear predictor, which is learned for the largest motions, for the transformed templates. Then, the parameters of the template that is best according to the resulting mean image intensity differences are selected and used to process the best template only. For this best template we continue with applying the remaining linear predictors of the multi-predictor approach.

Algorithm 2 Tracking with Occlusion Handling

function *TrackWithOcclusionHandling* (
in Image **I**, ***in/out*** TemplateParameters $\boldsymbol{\mu}$)
Compute homography $\mathbf{T}_{\boldsymbol{\mu}}$ from $\boldsymbol{\mu}$.
for $layer = 1 \rightarrow 3$ **do**
for each template i of this layer **do**
Compute $\boldsymbol{\mu}_i$ from homography $\mathbf{T}_{\boldsymbol{\mu}}$.
Track (**I**, $\boldsymbol{\mu}_i$) (see Alg.1).
Compute homography $\mathbf{T}_i$ from $\boldsymbol{\mu}_i$.
end for
Combine pose estimates $\mathbf{T}_i$ to $\mathbf{T}_{layer}$ (Sec. 5.2)
if $\mathbf{T}_{layer}$ is valid (Sec. 5.2) **then**
$\mathbf{T}_{\boldsymbol{\mu}} \leftarrow \mathbf{T}_{layer}$.
end if
end for
Detect occlusions (Sec. 5.3).
Adapt top-layer template (Sec. 5.4).

6 Experimental Results

6.1 Template Adaption

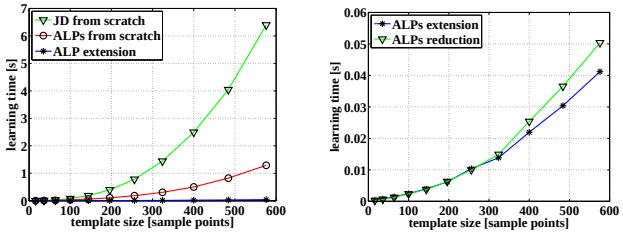
In this section we evaluate the template adaption proposed in Sec. 4 about template adaption. For this purpose, we perform an extensive comparison with several state-of-the-art approaches on template tracking. This includes comparisons with the standard learning approach of Jurie and Dhome [11], the analytical approach of Benhimane and Malis [10] and a recent approach called NoSLLip of Zimmermann *et al*. [17]. 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 of the 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 of the 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% since this sequence contains very large motions.

The probably most important results for this section are shown in Fig. 10, where we demonstrate that our ALPs method gives similar tracking results as the approach proposed by Jurie and Dhome [11].

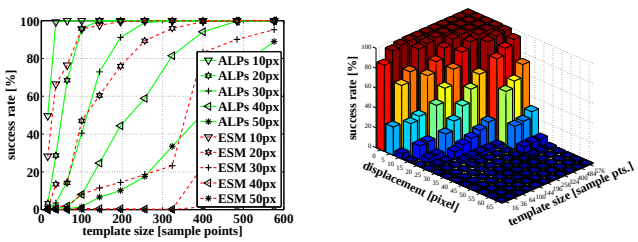
6.1.1 Comparison with Jurie-Dhome Approach

6.1.1.1 **Computational Complexity of Learning**: In Fig. 7 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 [11], depicted as a green curve in Fig. 7(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 this case we start with a small initial template, which size is equal to the size of an extension template of Fig. 4. Such a small template is then grown until the specified size is reached. The



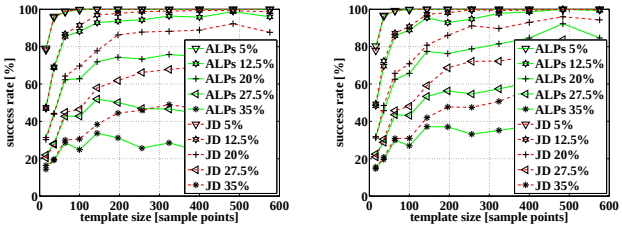
(a) (b)

Fig. 7. Comparison of the computation time necessary for learning a linear predictor using the Jurie-Dhome [11] (JD) approach (green) and using ALPs (red and blue). **(a)** For the latter 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 entire 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. The blue curve corresponds to the blue curve at (a).



(a) (b)

Fig. 8. Comparison of the 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)

Fig. 9. Comparison of the 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.

obtained results clearly reveal 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 for efficiently learning of a linear predictor for templates of a fixed size, starting from small templates and adapting their linear predictors until the desired template size is reached. In the following experiments for ALPs we always use this procedure for creating the linear predictors for a template. In the second case, shown as a blue curve and labeled as ALP extension, 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. 7 (b) we show computation times for the 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 computation from scratch would need over 1s using ALPs and more than 6s when using the approach of Jurie and Dhome [11].

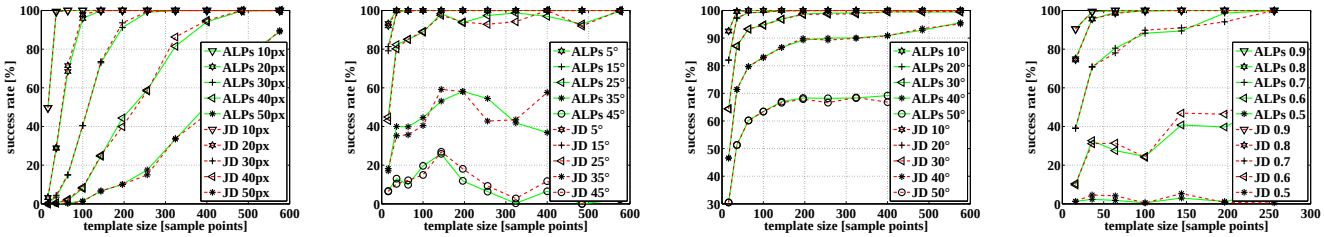
6.1.1.2 **Robustness**: To evaluate the robustness of our approach, we compare the tracking success rate of our approach with that of the standard approach proposed by Jurie and Dhome [11] for different template sizes and with respect to changes in translation (Fig. 10 (a)), in-plane rotation (Fig. 10 (b)), viewing angle (Fig. 10 (c)), and scale (Fig. 10 (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. 9. The robustness is measured using the tracking success rate. The accuracy is measured using the maximum mean template corner error. That is, after each tracking experiment the resulting template position is warped onto the original frame and the corner with the highest error is selected and used to compute the mean error. A template is considered as successfully tracked, if the maximum corner error is below 5 pixels.

For all of the 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 $\epsilon \in \left[-\alpha I_{\text{range}}/100, \alpha I_{\text{range}}/100\right]$, 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 of the experiments we also add a random displacement in the range of $[-5, 5]$ pixels, with the exception of displacement experiments, and a random change in the view-point angle ranging between $[-5°, 5°]$, 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 the learning of linear predictors, as demonstrated in Fig. 9 (b).

6.1.2 Comparison with ESM

To demonstrate the usefulness of learning-based approaches, we compare our approach with the analytical



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

Fig. 10. Comparison of the 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.

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 [17] and results obtained using ALPs (C++ implementation) using the PHONE sequence.

method called ESM of Benhimane and Malis [10]. This approach minimizes the energy function using a second order approximation. In Fig. 8 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.

6.1.3 Comparison with NoSLLiP

We also compare ALPs to the approach of Zimmermann *et al*. [17] using the PHONE sequence provided by the authors. Example images of the tracking are shown in Fig. 11. The comparison between the tracking results of [17] 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*. [17], we still obtain a better loss-of-locks count. The given error values are relative 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. 11).

6.1.4 Usefulness of larger templates

As shown in Figures 8, 9 and 10, 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.

6.1.5 Qualitative Evaluation

In Fig. 1, 11 and 12 we show different image sequences, which demonstrate the processing of the proposed approach. In Fig. 1 and 12 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. 12 we demonstrate the use of multiple templates. In Figures 1 and 11 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.

6.2 Occlusion-Aware Tracking

In this section we compare our occlusion-aware tracking presented in Sec. 5 with different state-of-the-art linear predictor based tracking approaches in terms of robustness and accuracy with respect to different types of motion using ground truth data. This is done both in the case of the presence of occlusions and without them. Additionally, we show several qualitative results from real video sequences. All of the experiments are again performed on a 2.26GHz Intel(R) Core(TM)2 Quad CPU with 4 GB of RAM, where only one core is used for the computation.

The robustness is again measured using tracking success rate and accuracy is computed using the maximum mean template corner error as explained in Subsec. 6.1.1.2 on robustness. For all ground truth experiments, we again use synthetic images, corrupted by noise and warped according to the specific experiments as already described in Subsec. 6.1.1.2. In this case the size of the templates used for all ground truth experiments is 16×16 sample points. The tracking speed for such a template is approximately 45 frames per second.

The probably most important results for this section are shown in Fig. 15, where we demonstrate that our multi-layered approach gives better tracking results in presence of occlusion compared to the approach of Jurie and Dhome [12].

6.2.1 Experiments without Occlusion

In Fig. 13 we compare our multi-layered ALP approach of Sec. 5 with the ALP approach proposed in Sec. 4. For the ML-ALPs approach we distinguish between the one as described in Sec. 5 ('Robustified ML-ALPs') and the one without the use of the transformed versions of the top layer template as described in Sec. 5.1 ('ML-ALPs'). The results show that in most cases both versions

Fig. 11. Result images of the phone sequence, which is provided by Zimmermann *et al*. [17]. Note that the template is reduced if it goes out of the image and grown again if it once again becomes visible.

Fig. 12. Iterative growing of two independent templates.

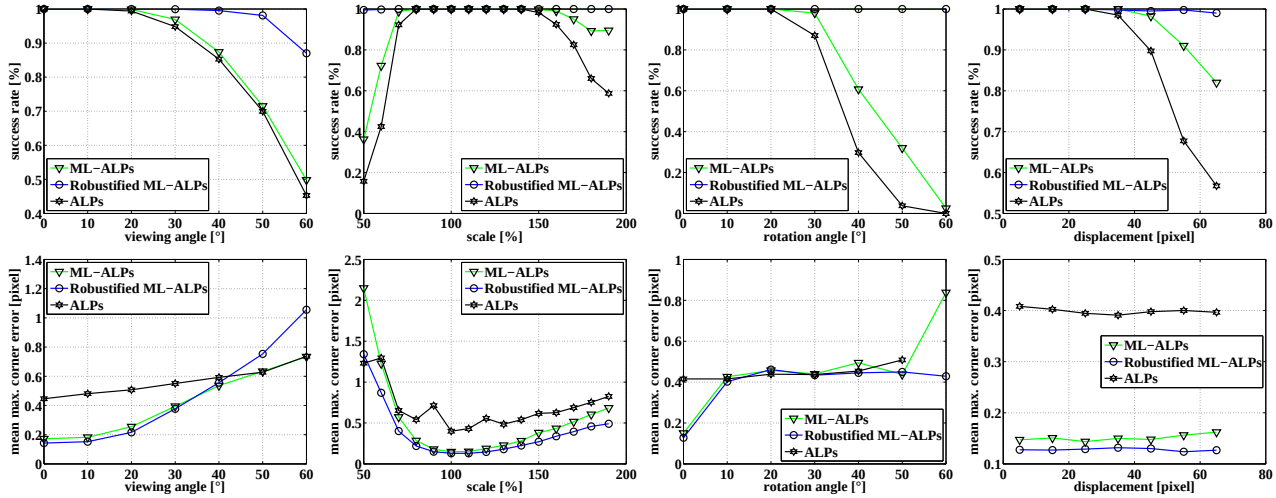


Fig. 13. Results of the comparison between the ML ALPs approach and the ALPs approach with respect to different types of motion without occlusion. The first row shows the tracking success rate and the second row the corresponding mean maximum corner errors.

perform better than ALPs. Only the accuracy of ALPs is sometimes better in case of large motion.

In Fig. 14 we compare ML-ALPs ('ML-ALPs') with the approach of Jurie and Dhome ('ML-JD') [12]. To evaluate our method we do not use the transformed versions of the top layer template as described in Sec. 5.1 in order to present a fair comparison with the multi-layered approach of Jurie and Dhome [12]. The results show that our approach performs better than the one of Jurie and Dhome [12] with respect to robustness as well as accuracy.

6.2.2 Experiments with Occlusion

In Fig. 15 we compare our ML ALPs approach with the one of Jurie and Dhome ('ML-JD') [12] in the presence of occlusions. Again, we do not use multiple transformed versions of the top layer template for the sake of fair comparison. The results show that our approach performs better than the approach of Jurie and Dhome [12] with respect to robustness as well as accuracy. The only exception is the view-point angle experiment, where ML-JD gives better accuracy for large occlusion and better robustness for large occlusion and small motion. Note also that the results of the approach of Jurie and Dhome [12] with respect to changes in the view point angle are stable for varying amounts of occlusion.

6.2.3 NoSLLiP with Occlusion

In this section we compare ML-ALPs to the approach of Zimmermann *et al*. [17] using the MOUSEPAD sequence, which contains occlusion and is provided by the authors. Example images of the tracking are shown in Fig. 16. The comparison between the tracking results of [17] and those obtained using ML-ALPs are shown in Table 2. The given error values are relative to the upper edge of the template. A frame is counted as loss-of-lock if one of the

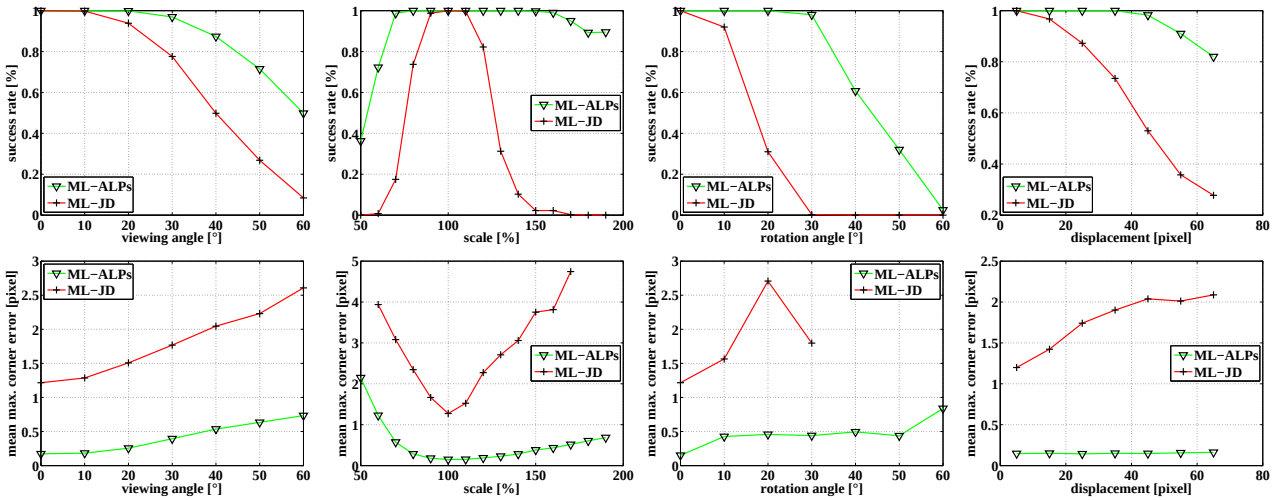


Fig. 14. Results of the comparison between the ML ALPs approach and the ML approach of Jurie and Dhome [12] with respect to different types of motion without occlusion. The first row shows the tracking success rate and the second row the corresponding mean maximum corner errors.

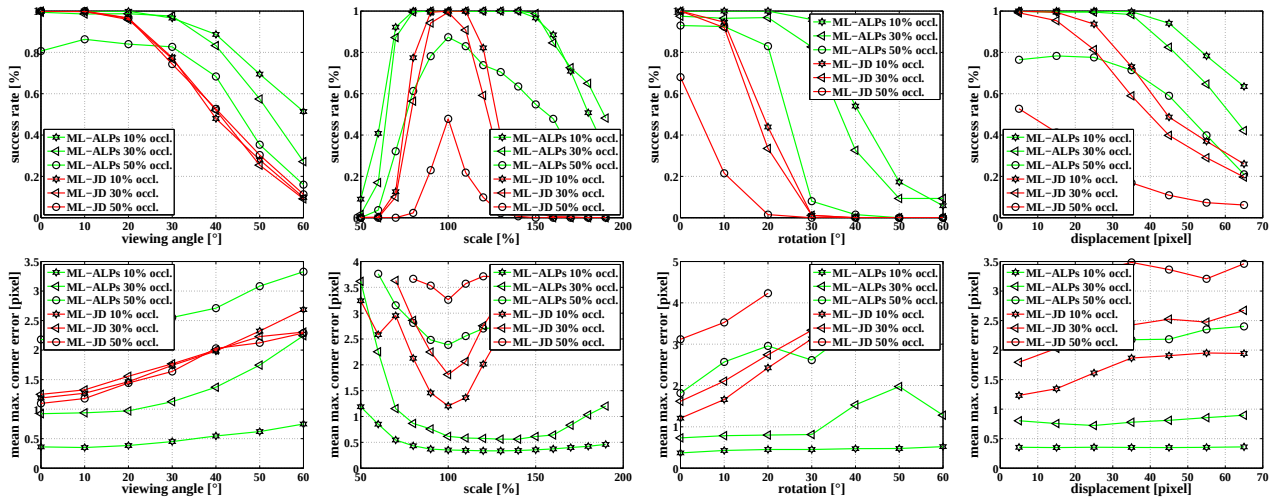


Fig. 15. Results of the comparison between the ML ALPs approach and the approach of Jurie and Dhome [12] with respect to different types of motion with occlusion. The first row shows the tracking success rate and the second row the corresponding mean maximum corner errors.

Method	Frame-rate [fps]	Loss-of-locks [-/-]	Error [%]
NoSLLiP	18.9	13/6935	1.5
ML-ALPs	17.2	1/6945	2.1

TABLE 2
Comparison between the tracking results of NoSLLiP (Matlab implementation) given in [17] and results obtained using ML-ALPs (C++ implementation) using the MOUSEPAD sequence.

template corners has an error larger than 25%.

6.2.4 *Qualitative Evaluation*

Fig. 16 and 17 demonstrate the performance of the proposed method on different image sequences. Fig. 16 shows several cases of partial occlusion, caused by a hand and a pen moving through the template from one side to the other. Fig. 17 shows a paper with a rectangular hole moving over a background surface. In this case the initially learned template, which includes the backround visible through the hole, is removed automatically when moving.

7 Conclusion

In order to support the dynamic change of the template shape, we introduce an efficient method for adapting linear predictors use for real-time template tracking . Our method allows both, the enlargement and reduction of the template size. We demonstrate that our ALPs approach can also be used to efficiently learn linear predictors for templates of a fixed size. In this case we start from small templates and adapt their linear predictors until the desired template size is reached. This results in much shorter learning time compared to the standard approach of Jurie and Dhome [11]. The efficiency of the presented approach derives from the special computation of the matrix inverse. In the

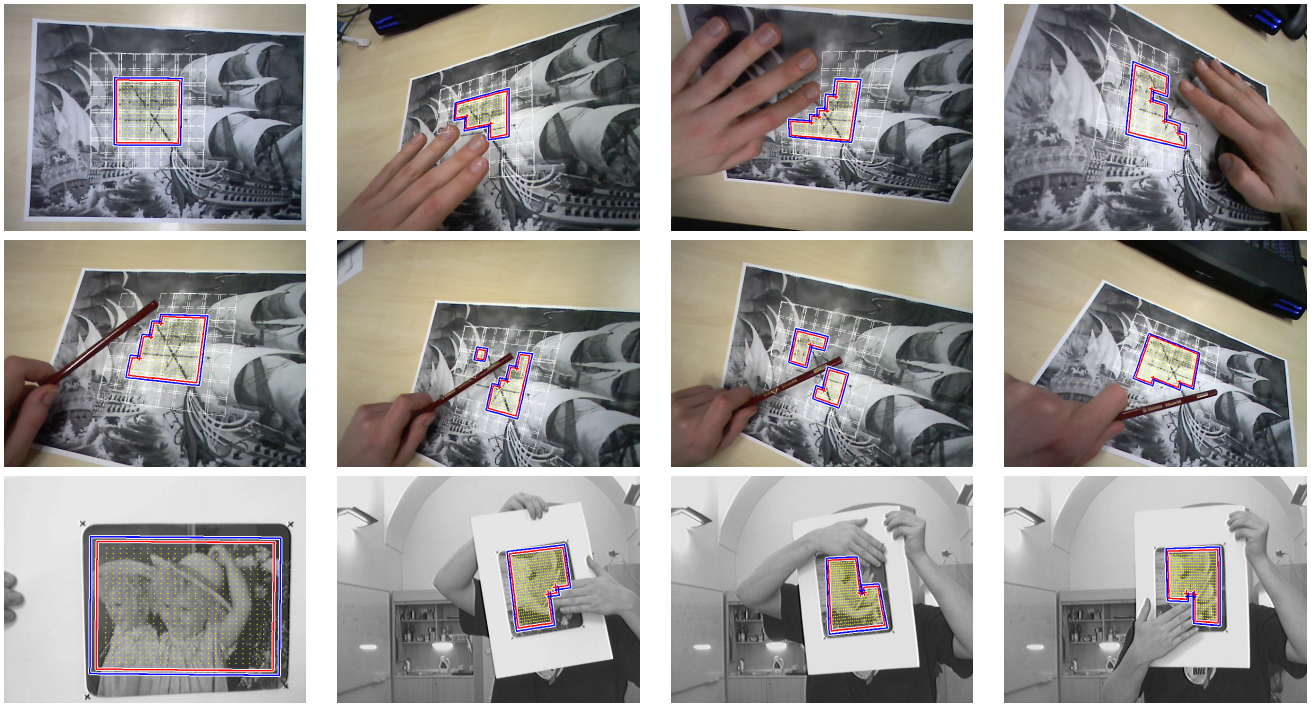


Fig. 16. Tracking with occlusions. **Upper two rows:** The upper left image shows the initially learned template. The other images show the template when occluded by a hand and when occluded by a pen which is moving through the template, both while the camera is moving. In both cases the shape of the template is automatically adapted according to the present occlusions. **Lower row:** The left image shows the initially learned template. The template is tracked over time, where occlusion occures at the end of the sequence.

Fig. 17. Tracking a template with a hole inside. The first image shows the template on a white background to show the hole in the middle. The second image shows the background. In the right image the template is learned with the background image. In the remaining images the template tracking results are shown. Note that the background behind the hole is changing, thus the hole is removed from the template and does not disturb the tracking.

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.

In this context we also introduce a robust method for detecting and handling occlusions. The multi-layer approach enables tracking of a template in the case of the abrupt occurence of occlusions. Early detection of incoming occlusions is necessary to adapt the top layer template before it is occluded, so that occlusion is prevented. The use of multiple transformed versions of the top layer template significantly increases the robustness with respect to large motions.

We demonstrate that our ALPs yield tracking results comparable to those of the standard approaches, while learning is much faster, and that the occlusion-aware tracking yields superior tracking results with respect to robustness and accuracy.

ACKNOWLEDGMENTS

This project was partially funded by the Bayerische Forschungsstiftung.

REFERENCES

[1] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada, August 1981, pp. 674–679.
[2] H.-Y. Shum and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, February 2000.
[3] G. Hager and P. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, October 1998.
[4] 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*, vol. 22, no. 4, April 2000.
[5] F. Dellaert and R. Collins, "Fast image-based tracking by selective pixel integration," in *ICCV Workshop of Frame-Rate Vision*, Kerkyra, Greece, September 1999, pp. 1–22.
[6] S. Baker and I. Matthews, "Equivalence and efficiency of image alignment algorithms," in *Conference on Computer Vision and Pattern Recognition*, vol. 1, Los Alamitos, CA, USA, December 2001, pp. 1090–1097.
[7] ——, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, pp. 221–255, March 2004.
[8] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *IEEE International Conference on Robotics and Automation*, vol. 2, New Orleans, LA, USA, May 2004, pp. 1843–1848.
[9] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Conference on Intelligent Robots and Systems*, vol. 1, New Orleans, LA, USA, Sept 2004, pp. 943–948.
[10] ——, "Homography-based 2d visual tracking and servoing," *International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, July 2007.
[11] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 996–100, July 2002.
[12] ——, "Real time robust template matching," in *British Machine Vision Conference*, Cardiff, UK, September 2002, pp. 123–131.
[13] C. Gräßl, T. Zinßer, and H. Niemann, "Efficient hyperplane tracking by intelligent region selection," in *Image Analysis and Interpretation*, Lake Tahoe, NV, USA, March 2004, pp. 51–55.
[14] P. Parisot, B. Thiesse, and V. Charvillat, "Selection of reliable features subsets for appearance-based tracking," in *Signal-Image Technologies and Internet-Based System*, Shanghai, China, December 2007, pp. 891–898.
[15] J. Matas, K. Zimmermann, T. Svoboda, and A. Hilton, "Learning efficient linear predictors for motion estimation," in *Proceedings of 5th Indian Conference on Computer Vision, Graphics and Image Processing*, Madurai, India, December 2006, pp. 445–456.
[16] W. W. Mayol and D. W. Murray, "Tracking with general regression," *Journal of Machine Vision and Applications*, vol. 19, no. 1, pp. 65–72, January 2008.
[17] K. Zimmermann, J. Matas, and T. Svoboda, "Tracking by an optimal sequence of linear predictors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. 1, pp. 677–692, April 2009.
[18] S. Baker, R. Gross, I. Matthews, and T. Ishikawa, "Lucas-kanade 20 years on: A unifying framework: Part 2," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-01, February 2003.
[19] I. Patras and E. Hancock, "Regression-Based Template Tracking in Presence of Occlusions," in *Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services*, Santorini, Greece, June 2007, p. 15.
[20] C. Gräßl, T. Zinßer, and H. Niemann, "Illumination insensitive template matching with hyperplanes," in *Proceedings of Pattern recognition: 25th DAGM Symposium*, Magdeburg, Germany, September 2003, pp. 273–280.
[21] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *SIAM Review*, vol. 23, no. 1, pp. 53–60, January 1981.
[22] 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*, Anchorage, AK, USA, June 2008, pp. 1–8.
[23] S. Holzer, S. Ilic, and N. Navab, "Adaptive linear predictors for real-time tracking," in *Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, June 2010, pp. 1807–1814.

Stefan Holzer is a PhD student at the institute for Computer Aided Medical Procedures (CAMP) at the Technische Universität München (TUM) where he is part of the Computer Vision group. He received his diploma in Computer Science from the University of Applied Sciences Landshut in 2006 and his M.Sc. in Computer Science from the TUM in 2009. He was affiliated with the group of Dr. Andrew Davison at the Department of Computing at Imperial College London, with Willow Garage where he worked on object detection in the context of robot based object grasping, and more recently with Microsoft Research Cambridge where he worked on learning-based approaches. Stefan Holzer's current research interests include reconstruction, object detection, tracking, and pose estimation for 2D/3D objects. You can find out more about his work by visiting http://campar.in.tum.de/Main/StefanHolzer.

Slobodan Ilic is senior research scientist working at TU Munich, Germany. Since February 2009 he is leading the Computer Vision Group of the CAMP Laboratory at TUM. Form June 2006 he was a senior researcher at Deutsche Telekom Laboratories in Berlin. Before that he was a postdoctoral fellow for one year at Computer Vision Laboratory, EPFL, Switzerland, where he received his PhD in 2005. His research interests include: deformable surface modeling and tracking, 3D reconstruction, real-time object detection and tracking, object detection and classification in 3D data. Slobodan Ilic serves as a regular program committee member for all major computer vision conferences, such as CVPR, ICCV and ECCV as well as journals, such as TPAMI and IJCV. Besides active academic involvement Slobodan has strong relations to industry and supervises a number of PhD students supported by industry.

Nassir Navab is a full professor and director of the Computer Aided Medical Procedures & Augmented Reality institute at Technische Universität München. He also has a secondary faculty appointment at TU München's Medical School. Nassir has a PhD in computer science from INRIA and the University of Paris XI. In 2006 he was elected as a member of board of director of the MICCAI society. He also served on steering committee of IEEE Symposium on Mixed and Augmented Reality 2000-2008. He acts as associated editor of IEEE Transactions on Medical Imaging and serves on the editorial board of Medical Image Analysis. Nassir is the author of more the 50 US and international patents. He received the Siemens Inventor of the Year Award in 2001 and the SMIT Society Technology Award in 2010. His PhD students have received many prestigious awards including four MICCAI young scientist awards, ISMAR, ISBI, DAGM, VOEC-ICCV and AMDO best paper awards. His research interests include computer vision, augmented reality, computer-aided surgery and medical image registration.