Rapid Selection of Reliable Templates for Visual Tracking

Nicolas Alt[1], Stefan Hinterstoisser[2] and Nassir Navab[2]
[1]Institute for Media Technology [2]Chair for Computer Aided Medical Procedures
Technische Universität München, 80290 München, Germany
n.alt@tum.de, {hinterst, navab}@in.tum.de

Abstract

We propose a method that rates the suitability of given templates for template-based tracking in real-time. This is important for applications with online template selection, such as SLAM, where it is essential to track a low number of preferably reliable templates. Our approach is based on simple image features specifically designed to identify texture properties which are problematic for tracking. During a training step, a support vector regressor is learned. It uses a tracking quality measure which considers both convergence rate and speed obtained by simulation of many tracking attempts. Finally, a minimum set of image features is identified to speed up the online selection process. In experiments on real-world video sequences our method improved the detection rate of an existing tracking-by-detection system by 8% on average.

1. Introduction

Since the pioneering work of Lucas and Kanade [13], template-based tracking has been an important technique for many applications, such as augmented reality, visual servoing and industrial automation. Various algorithms [3, 4, 8, 11, 20] have been presented aiming to improve efficiency and robustness especially with respect to fast movements.

However, success rate and speed are not only dependent on the selected tracking algorithm but also on the texture of the template to be tracked: The texture may not provide enough information for a reliable estimation of the image warp. Several methods [2, 7, 18, 20] were suggested to overcome this problem and to select the template with a promising texture. While [7, 18] came up with an analytic measure of "texturedness" of a template, more recent approaches [2, 20] obtain better results using training steps. However, extensive training is not possible for online tracking systems which are working in a permanently changing environment and must constantly identify new templates from a large set of available "candidate" templates. Therefore, these applications demand a real-time method to find templates that promise most reliable and fastest tracking.

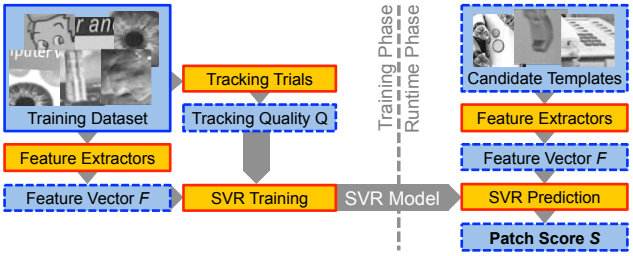

Fig. 1. Proposed template rating system. In a one-time offline training phase, a regressor is trained with a large dataset of arbitrary templates. During runtime, the regressor is used to reliably predict tracking behavior of any observed template in real-time.

In this paper, we propose a fast and reliable method which rates the tracking suitability of templates in order to identify the best templates on an object. Given an image template at run-time we predict the tracking behavior based on its texture. An appropriate model is hard to grasp analytically, and existing methods such as [20] are too slow for real-time applications. Our method is based on learning the relationship between simple and fast-to-evaluate template features and a tracking quality measure in an offline stage. Processing time in the online phase is then in the order of one millisecond, allowing real-time processing.

The rest of the paper is structured as follows: We recall existing work in the remainder of this section. Sec. 2.1 describes the quality measure used in our approach. Sec. 2.2 presents simple and fast features used for learning and prediction. Rapid template rating (or quality prediction) is discussed in Sec. 2.3. We give detailed evaluation on simulated tracking experiments as well as on real-world videos in Sec. 3 and conclude with Sec. 4.

Related Work Lucas and Kanade [13] introduced a popular method to register the deformed view of a template to a reference view by minimizing a linear approximation of intensity image differences with a steepest decent approach. Many other so called first order template-based tracking algorithms have been suggested meanwhile (refer to [1] for an overview), such as the inverse compositional (IC) approach.

Recently, [3] presented an efficient algorithm (ESM) that behaves superior to all analytic first order approaches in terms of convergence rate and convergence basin. Instead of using analytic methods, [11] presented a tracking algorithm (JD) that is based on learning, assuming a linear relationship between image difference and motion. The advantage compared to the analytic approaches lies in the larger convergence basin.

Shi and Tomasi [18] discussed the problem of identifying textures which are suitable for tracking. They came up with a "texturedness" measure that identifies pixels which can be located precisely and robustly by measuring the conditioning of their tracker. We pick up this idea in Sec. 2.2.2.

Carneiro [5] proposed a method to find robust and distinctive local image features, using a learning approach. They investigated a similar problem as we do in the context of feature descriptors. For template tracking we must define new features, while [5] relies on the existing feature descriptors. Also the target value (quality) must be defined in a specific way for trackers.

A method to select an active set of templates on an object was presented by [20], based on optimization of a coverage and a quality measure. However, the latter requires learning and evaluation of a large number of "candidate predictors" (or trackers) evenly distributed on the object. This must be done during a time-consuming offline phase, which is not suitable for online systems working with constantly changing objects. In contrast, our approach predicts tracking performance of templates in real-time, without the need to learn or evaluate a tracker for each new object.

Another contribution of [20] is to find template pixels which behave optimal with respect to the linearity assumption in the tracking equations. It uses a greedy search over random subsets of pixels which is very time consuming especially for larger templates. Benhimane [2] also presented a method that finds subsets of template pixels which verify the analytic approximation (linear or quadratic) of the tracking algorithm. Combined with a bottom up search strategy and a stable union operation it avoids the combinatorial explosion of [20] and is therefore also suitable for larger templates. Both approaches improved the convergence behavior significantly. However, the time to obtain these special pixel sets is in both cases far from real-time and makes the presented methods unsuited for online applications.

Mac Aodha [14] proposed a method using learning to find the best algorithm for optical flow estimation on a per-pixel basis. Some of their features are similar to ours, but they also use complex ones not suitable for an online method. Finally, several authors investigated metrics for tracking uncertainty or confidence during the tracking process [12, 15]. In this work, we seek a metric that does not rate individual observations, but is general for a given template and algorithm setup.

2. Proposed Approach for Template Rating

The presented method estimates the suitability of a given template for tracking in real-time and is partitioned into a training and a runtime phase, see Fig. 1: During the one-time training phase, tracking quality is measured by simulation for a large database of templates. At the same time, certain features are evaluated on all templates of said database. In a subsequent step, a regressor is trained using template features (Sec. 2.2) as input variables and the quality measures as target values. At run-time, tracking quality is predicted in real-time using the template features and the previously learned regressor.

2.1. Measuring Tracking Quality

In this section, we discuss how to measure the tracking behavior for a given template $\mathcal{T}$ using a given tracking algorithm. The time-consuming process is based on extensive testing by simulation, which considers most of the factors relevant for tracking and allows to realistically observe the algorithm's behavior. A *tracking quality measure* is obtained and later used as a reliable "ground truth".

Given a template $\mathcal{T}$ with surrounding texture seen under N_v randomly selected artificial views with noise added, N_v tracking trials are performed. The extent of the warp is chosen according to the abilities of the investigated tracker. The convergence likelihood for $\mathcal{T}$ is $p_c = \frac{N_c}{N_v}$ where N_c denotes the number of views for which the tracking algorithm converges correctly. Tracking converges if the spatial root mean square error is below a certain threshold τ_{rms} after at most N_{rms} trials. The convergence rate serves as a straightforward measure for the tracking quality of one template $\mathcal{T}$ using one tracking algorithm:

$$\mathcal{Q}^r = p_c \tag{1}$$

Additionally, for the converged trials $k = 1, \ldots, N_c$, the average convergence speed $\overline{s} = \frac{1}{N_c} \sum_k s_k$ is obtained by averaging over the number s_k of iterations until convergence. We define the combined tracking quality $\mathcal{Q}$, taking into account both tracking convergence rate and speed as follows:

$$\mathcal{Q} = p_c \left(1 - \frac{w\overline{s}}{N_{rms}}\right) \tag{2}$$

where weighting factor $w \in [0; 1]$ (here $w = 0.5$ is used) determines the importance of tracking speed with respect to convergence rate. N_{rms} is the maximum number of allowed iterations (such that $\overline{s} \leq N_{rms}$) and is chosen based on constraints of the available computational time. With the definition of Eq. (1) and (2), the values of the quality measures lie between 0 and 1, where 0 denotes a bad template for tracking. The measure $\mathcal{Q}$ showed to be a more discriminative metric, and it is also more relevant for for real-time system, where convergence speed is of great interest. We will mainly use $\mathcal{Q}$ in the rest of this document, but the de-

sign process is valid for $\mathcal{Q}^r$ as well. Fig. 2 shows some example templates along with their quality $\mathcal{Q}$ (red bar).

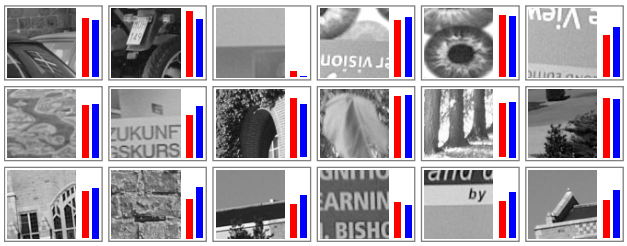

Fig. 2. Some templates from the testing dataset. Bars indicate the template's quality $\mathcal{Q}$ (left/red, see Eq. (2)), and its predicted score $\mathcal{S}$ (blue, Eq. (5)) with algorithm JD. Note that most of the good templates look salient to a human observer.

2.2. Template Features

This section discusses the template features we propose for our template rating method. They are derived from the grayscale reference template $\mathcal{T}$ and will be used in Sec. 2.3 for real-time prediction of $\mathcal{Q}$. All features (denoted $\Phi_\star$ with index $\star$) are collected in a feature vector $\mathcal{F}$.

2.2.1 Texture Features

The proposed texture features are designed to extract properties relevant for tracking success from the reference template $\mathcal{T}$. Specifically, we consider the following properties which are unfavorable for tracking:

1. Low spatial locality: Locality, i.e. similar pixel intensities in a local neighborhood, is one of the basic assumptions for tracking. The JD and IC algorithms use a linear approximation of the warp–error image relation, which becomes invalid in case of low locality.
2. Uniform texture: Large texture areas of the same color do not show an appearance change when an image warp is applied. The respective region cannot provide any information about the observed warp. Therefore, uniform areas are unfavorable for tracking, especially if they are located close to the template's boundaries.
3. Low contrast: A low contrast of the texture leads to a higher sensitivity to noise and lighting changes.
4. Directional texture: Uni-directional structures such as parallel lines look equal or similar under translations in a certain direction. This leads to a vanishing tracker error image for said translations, and the observed warp cannot be uniquely determined.
5. Repetitive texture: Regular patterns exhibit a similar problem as directional textures: They lead to a zero error image for periodic displacements and thus allow multiple solutions for the observed warp.

It is hard to derive a metric for any of those properties in a formal way, which is why we need to work with heuristic design techniques in the following. In order to verify the validity of these design steps, parameter optimization and evaluation steps are discussed in Sec. 2.2.3 & 2.2.4. All features are implemented using simple image processing techniques to facilitate real-time processing.

Subregion Uniformity This feature evaluates properties 2 and 3 from above in an efficient and fast way: First, template $\mathcal{T}$ is subdivided into N_{sub}^2 rectangular subregions of equal size. Next, the intensity variability δi of each subregion is found by calculating the difference between the brightest and darkest pixel. In case this local intensity difference is small, the respective subregion exhibits a uniform texture. Two scalar feature are derived from the intensity difference δi:

- Number of subregions for which $\delta i < \tau_{sub}$, Φ_{C_G}
- Sum of δi for all subregions, Φ_{C_Σ}

Both features are normalized by $\frac{1}{N_{sub}^2}$. Optimization (see Sec. 2.2.3) yields $N_{sub} = 5$ and $\tau_{sub} = 20$.

Frequency spectrum We tackle properties 1, 4 and 5 using a Discrete Fourier Transform (DFT), yielding $\mathfrak{T} \bullet\!\!-\!\!\circ \mathcal{T}$. Pixels in the 2D image $\mathfrak{T}$ correspond to orthogonal periodic base functions (spatial waves), which exhibit a spatial frequency f and a spatial direction θ. Thus, $\mathfrak{T}$ can be expressed in terms of variables f and θ. High-frequency spatial waves exhibit a low locality and may be caused by repetitive structures. Very low frequency components have a high locality, but exhibit the same problem as uniform texture areas (property 2). This is why a high share of mid-frequency components is favored. In addition, the directionality of a texture can be found based on the spatial direction θ of waves.

We integrate $|\mathfrak{T}|\,(f,\theta)$ respectively over θ and f, yielding two 1D-histograms: One for the spatial frequency distribution H_f, and another one for the spatial direction of waves H_θ. Histogram H_θ is normalized to $\sum_{\textbf{bins}} H_\theta = 1$ and only considers pixels for which $f \in [f_s, f_e]$. The number of bins is set to 18, providing a resolution of $10°$. Three scalar features are extracted from histograms H_f and H_θ:

- Share of mid-range frequency components for $f \in [f_s, f_e]$, $\Phi_{\mathrm{FT}_E} = \int_{f_s}^{f_e} H_f$
- Equipartition of direction histogram H_θ, expressed by the variance of fill-levels $\Phi_{\mathrm{FT}_{\mathrm{VAR}\,\theta}}$
- Histogram peak comparison, yielding $\Phi_{\mathrm{FT}_{\Delta\theta}}$

Optimal values for the parameters are found by optimization (see Sec. 2.2.3), which yielded for the JD algorithm:

$$f_s = 0.019s, f_e = 0.11s,$$

with template size $s = \frac{1}{2}\sqrt{w^2+h^2}$, where w, h are the width and height of the template. For "peak comparison", we compare the difference of the strongest and k-strongest

peaks in a histogram, normalized by the strongest peak. It serves as quick estimate of the histogram distribution. Typically, we use $k = 3$ for an 18 bin histogram.

Edge image An edge image as generated by the Sobel operator [19] shows edge strengths and directions in an image. Edges are borders between areas with a uniform intensity value. The edge strengths and directions are an indication for the uniformity property (2) or the directionality property (4), respectively. We generate a global edge direction histogram $H_{S\theta}$ from the Sobel-filtered image. Finally, two scalar features are calculated:

- Global sum of edge strengths on template, Φ_{S_Σ}
- Peak comparison in histogram $H_{S\theta}$, yielding $\Phi_{S_{\Delta\theta}}$

Intensity histogram Finally, we design a feature extractor evaluating properties 2 and 3 based on the image intensity histogram. It yields features Φ_{H_C}, $\Phi_{H_{\mathrm{VAR}}}$ and Φ_{H_Δ}. We do not discuss these features here in detail, as they do not belong to the top features identified in Sec. 2.3.4.

2.2.2 Analytic Features

Analytic features are derived from the tracking equations and may thus be justified by construction. However, before they can be calculated, a tracker needs to exist. In the case of the JD algorithm, this means that a time-consuming training step must be performed, rendering the corresponding features unfeasible for real-time applications.

Shi and Tomasi [18] discuss the problem of identifying textures which are suitable for tracking, and come up with an analytic method applied to their own tracking algorithm. Their basic tracking equation is

$$Z\mathbf{d} = \mathbf{e}, \tag{3}$$

with square tracking matrix Z, deformation parameters $\mathbf{d}$ and error image parameters $\mathbf{e}$. The authors argue that tracking only works well if Eq. (3) can be solved reliably and investigate the eigenvalues of Z. A vanishingly small eigenvalue means that Z is not of full rank and gets "blind" for pose changes along the corresponding eigenvector. Additionally, Eq. (3) is only well-conditioned if the eigenvalues do not differ by several orders of magnitude. In [18], a template is accepted if the smallest eigenvalue is above a pre-defined threshold. The present work uses the smallest eigenvalue directly as feature Φ_{SHI}. While the Shi-Tomasi feature is only analytically justified for the tracking algorithm presented in [18], it may also be applied to similar tracking algorithms using first order approximation.

We apply the idea of Shi and Tomasi to the IC and JD algorithms and investigate the conditioning of the respective tracking matrices using singular values. Matrix A (see [11, Eqn. 2]) is considered for JD, and for IC, we regard the steepest descent images (see [1, Fig. 2]). The following values are taken as template features:

- Smallest and largest singular values of steepest descent image, $\Phi_{SD\sigma_S}$ and $\Phi_{SD\sigma_L}$
- Ratio of these singular values, $\Phi_{SD\sigma_{SL}}$
- Largest singular value of first JD matrix, $\Phi_{JD\sigma_L}$
- Mean of largest singular values of JD tracking matrices (JD uses a set of tracking matrices), $\Phi_{JD\sigma_{\overline{L}}}$
- Ratio of smallest and largest singular value, $\Phi_{JD\sigma_{SL}}$

2.2.3 Parameter Optimization

In the above discussion about feature extractors (Sec. 2.2.1 & 2.2.2), several parameters or coefficients were introduced, whose values still have to be picked. Obviously they should be chosen such that the corresponding feature provides most information about tracking quality. Thus, we use a large training dataset of N templates $(\mathcal{T}_k)_{k=1}^N$ and choose parameters $\mathbf{p}$ such that the correlation of ρ of one feature's values $(\Phi_{\star,k}^{\mathbf{p}})_{k=1}^N$ with the quality measures $(\mathcal{Q}_k)_{k=1}^N$ is maximized:

$$\mathbf{p_{opt}} = \underset{\mathbf{p}}{\operatorname{argmax}} \left| \rho \left((\Phi_{\star,k}^{\mathbf{p}})_{k=1}^N, (\mathcal{Q}_k)_{k=1}^N \right) \right| \tag{4}$$

The qualities $(\mathcal{Q}_k)_{k=1}^N$ are obtained according to Eq. (2). Because of computational effort, only up to two parameters are optimized simultaneously and on a per-feature basis. A grid search is performed in order to solve Eq. (4).

2.2.4 Evaluation of Feature Performance

As stated earlier, each feature's significance must be checked by testing it on a large database of templates. Here, only the individual significance is considered, which means that no statement can be made about redundancies between features. We use a general dataset of $N = 2600$ templates, each 75×75 pixels in size. The dataset comprises arbitrary textures from indoor and outdoor scenes, some of which are shown in Fig. 2.

For each template in the dataset $(\mathcal{T}_k)_{k=1}^N$, the feature vector and quality measure are evaluated, yielding $(\mathcal{F}_k)_{k=1}^N$ and $(\mathcal{Q}_k)_{k=1}^N$ according to Eq. (2). The significance of a feature $\Phi_\star$ (i.e. an element of vector $\mathcal{F}$) is expressed by the correlation coefficient ρ between feature values $(\Phi_{\star,k})_{k=1}^N$ and quality measures $(\mathcal{Q}_k)_{k=1}^N$. A value of $|\rho| = 0$ indicates that $\Phi_\star$ is a useless feature, whereas for a linear dependence, $|\rho| = 1$. One weakness of the correlation coefficient is that it is suboptimal for uncovering nonlinear dependencies.

The absolute correlation coefficient is shown in Fig. 3, using the above dataset and two different tracking algorithms. For the JD algorithm, while most features exhibit a medium or low correlation, there are two features with $|\rho| > 0.6$, namely Φ_{C_G} and $\Phi_{H_{\mathrm{VAR}}}$. Low-correlated features might still be useful when combined with other fea-

tures (Sec. 2.3). For the IC algorithms, all correlation coefficients are rather low, with a maximum $|\rho| \approx 0.3$. While in that case individual features are almost useless, feature combination yields relevant results (see 2.3.2).

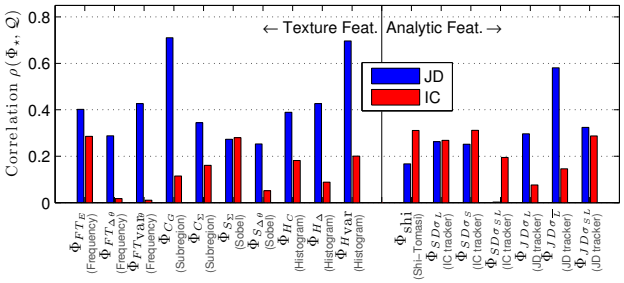

Fig. 3. Correlation coefficient between individual features $\Phi_\star$ and tracking quality $\mathcal{Q}$ for the JD and IC algorithms

2.3. Fast Prediction of Template Quality

Individual template features from Sec. 2.2.1 and 2.2.2 are not significant enough to predict tracking quality, see Fig. 3. They need to be combined in some way, using for instance a machine learning method. Features are used to calculate a template score $\mathcal{S}$, which serves as an estimate for the tracking quality $\mathcal{Q}$. This is based on the assumption that there exists a function f such that

$$\mathcal{Q} \approx \mathcal{S} = f(\mathcal{F}). \tag{5}$$

We use Support Vector Regression (SVR) to find function f. The calculation of the template score is faster by several orders of magnitude than measuring $\mathcal{Q}$ by extensive evaluation (see Sec. 2.1). Fig. 1 illustrates how the both phases of the template rating system work together.

2.3.1 Training Phase

During the training phase, which is performed only once, function f relating template score and features is found using supervised learning. A training set of templates is required, which is made up of 50% of all templates of the dataset mentioned in Sec. 2.2.4, randomly selected. For each template, the feature vector $\mathcal{F}$ is extracted and the quality $\mathcal{Q}$ is measured. They serve as the training data for an SVR regressor, with predictors $\mathcal{F}$ and target values $\mathcal{Q}$. The template score can therefore be at most as meaningful as $\mathcal{Q}$, underlining the importance of the measurement procedure in Sec. 2.1. Typically, SVR algorithms work with normalized data in the range $[-1; 1]$. Because of some scattered extreme values, we normalize predictors and target variables such that they exhibit a mean of 0 and a variance of 1.

SVR is a modification of Support Vector Machines for regression, and capable of dealing with nonlinearities. In recent years, support vector based machine learning methods showed great performance on many problems, which is why we chose it here. We use the LIBSVM [6] library, which implements, among others, ν-SVR [17]. Regressor training yields an SVR model, which defines function f. The regressor function f is solely derived from the training data, which should therefore be as representative as possible. Additionally, SVR is known to generalize well and does not tend to create overfitted models.

There are a few parameters for the SVR training, which must be specified *a priori*. Optimal values are found during supervised learning by maximizing the correlation between $\mathcal{S}$ and $\mathcal{Q}$. The most important parameters are the cost parameter for errors c and the coefficient for the radial kernel basis function γ. All other parameters are left at the default values. Optimization with built-in tools from LIBSVM yields: $c = 2^3, \gamma = 2^{-5}$.

2.3.2 Runtime Phase

This part of the proposed system runs online within a tracking system. It provides a lightweight and fast mean to estimate the tracking quality of templates. For systems such as SLAM (Simultaneous localization and mapping), new templates become available constantly, and their score (i.e. estimated tracking quality) must be determined in real-time. Template rating and selection is performed as follows:

- Define possible templates (candidate templates)
- Extract feature vector $\mathcal{F}$ for each candidate template
- Calculate score $\mathcal{S} = f(\mathcal{F})$ for each candidate template
- Select templates with highest scores for tracking

Template quality is estimated rapidly using $\mathcal{S}$, and the time-consuming measuring step for $\mathcal{Q}$ is not necessary.

2.3.3 Optimal Template Size

Besides the position of a template on an object, often there is also a free choice of template size g. Trackers can use sub-sampling, such that the template size does not influence the complexity. Of course, g must lie within a certain range, or even in a discrete set of possible sizes G. Depending on the size, obviously, different parts of an object's texture appear in the template, so in general, tracking quality $\mathcal{Q}$ changes for different values of g. The size selection is performed online, i.e. within the runtime phase, which means that only the template score $\mathcal{S}$ is available. The optimal size $g \in G$ for a size-dependent template $\mathcal{T}_g$ is chosen such that $\mathcal{S}$ becomes maximal:

$$g_{opt} = \underset{g \in G}{\operatorname{argmax}}\, \mathcal{S}_{\mathbf{size=g}} = \underset{g \in G}{\operatorname{argmax}}\, f(\mathcal{F}(\mathcal{T}_g)) \tag{6}$$

Features from Sec. 2.2 are designed such that if g is changed while the picture within the template region is scaled equally, feature vector $\mathcal{F}$ and consequently score $\mathcal{S}$ remain constant.

2.3.4 Feature Selection

Features from Sec. 2.2.1 exhibit redundancies, as they were designed with similar ideas in mind. In the following, we investigate if there is a small subset of features yielding similar prediction performance as the complete feature set. A smaller subset reduces the computational requirements for the online template rating process. Again, a large number of N templates is required for evaluation along with the measured tracking qualities $\mathcal{Q}$. The templates are assigned either to a training or a testing dataset.

The optimal subset of features with a given size can be found by training a regressor for each possible feature combination and selecting the best one. However, this is a very time-consuming process, which is why a greedy but much faster approach is presented here. Assume that an optimal subset of features $\mathcal{F}_i$ of size i is known. The optimal subset of features of size $i+1$ is found by training k regressors using features $\mathcal{F}_i$ plus one of the remaining k features on the training dataset. Evaluation of the regressor performance on the testing dataset yields the locally optimal subset of features $\mathcal{F}_{i+1}$ of size $i+1$. The process is repeated until there are no more features left. To start the iteration, $\mathcal{F}_1$ consists of the single feature with the best prediction performance.

We choose to evaluate regressor performance based on the prediction reliability p_{ok} with $\epsilon = 0.02$ (see Sec. 3.1). Fig. 4 shows p_{ok} over the number of features used for regression. The features based on the JD tracking matrix are not included, because they cannot be used for real-time processing anyway. Regressor performance with the described greedy feature selection is shown by the red curve, whereas the dashed blue curve depicts the performance when features are added based on the largest correlation coefficient $|\rho|$. One can see that 4 features, selected by the greedy method, are sufficient for prediction with a close-to-maximum reliability of $p_{ok} = 0.78$. The correlation-based selection method achieves this reliability only with 9 features. The four identified "top-features" are: Φ_{C_G}, Φ_{S_Σ}, Φ_{FT_E} and $\Phi_{S_{\Delta\theta}}$. As outlined in Sec. 2.2.1, except for Φ_{FT_E}, these features are constructed using very simple and fast operations. Note that $\Phi_{H_{\mathrm{VAR}}}$, which showed a high correlation in Fig. 3, does not offer a great gain in prediction reliability and is only chosen as the fifth feature.

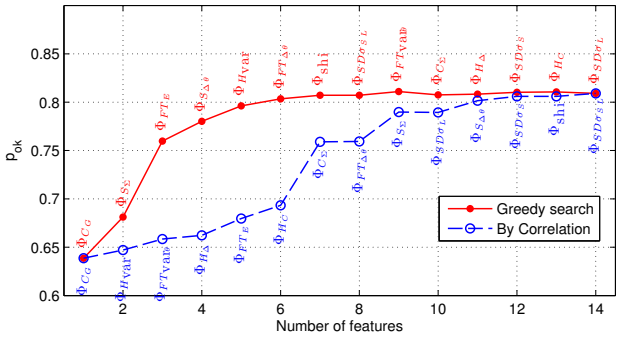

Fig. 4. Prediction reliability for JD over number of features. Two different methods for feature subset selection are used (see text). The feature added during each step is printed in the plot.

3. Results

3.1. Prediction Accuracy for Tracking

In a first experiment, we investigate the prediction accuracy of the proposed method for classic tracking with the JD and IC algorithms. The SVR predictor is applied to a testing set of ca. 1300 templates as outlined in Sec. 2.3.2, yielding a score $\mathcal{S}$ for each template. (The testing set is disjoint from the training set used in Sec. 2.3.1). Next, the tracking behavior of all templates is measured by running 200 tracking trials, as outlined in Sec. 2.1. The resulting quality $\mathcal{Q}$ is taken as the ground truth. Fig. 2 shows 18 templates, along with quality measures $\mathcal{Q}$ and scores $\mathcal{S}$ for JD.

The prediction error for a template k is the difference between prediction and ground truth: $E_k = \mathcal{S}_k - \mathcal{Q}_k$. Fig. 5 shows the probability density function (pdf) of $E_k \forall k$. Regressors are trained independently for the two shown quality measures $\mathcal{Q}$ and $\mathcal{Q}^r$. Note that this plot is based on a large dataset (1300 patches) and a total of $1300 \cdot 200 = 260000$ tracking experiments. Integration of the pdf in the interval $[-\epsilon, +\epsilon]$ yields the probability p_{ok} of prediction error $|E| \leq \epsilon$. Tab. 1 lists some values for p_{ok} and shows that the predictor for the tracking quality $\mathcal{Q}$ works accurately for the JD algorithm: An error value of less than ± 0.05 (or 5%, as $\mathcal{Q} \in [0, 1]$) is observed for 91% of all tested templates. For IC, a larger, but still acceptable prediction error is on hand.

3.2. Performance of GEPARD Tracking System

Second, we apply our novel method for template selection to the GEPARD tracking-by-detection system [9]. It was designed for fast perspective patch rectification and makes use of an online learning phase during which the system selects a low number of reference templates on the object. Afterwards, these templates are searched for in the cur-

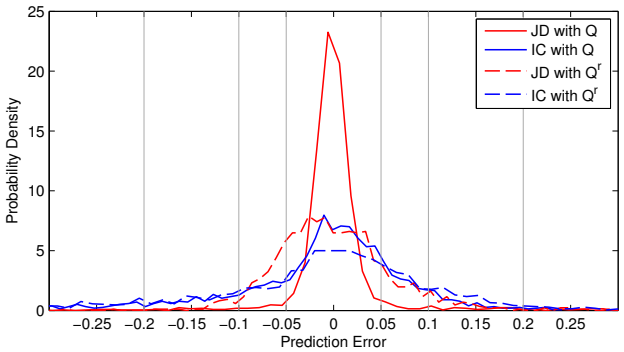

Fig. 5. Probability density function of the prediction error for tracking algorithms JD, IC and for quality measures $\mathcal{Q}$ & $\mathcal{Q}^r$

		Using $\mathcal{Q}$		Using $\mathcal{Q}^r$	
	Accuracy ϵ	JD	IC	JD	IC
p_{ok}	± 0.05	92%	55%	62%	43%
	± 0.10	95%	80%	90%	67%
	± 0.20	98%	94%	98%	87%

Table 1. Prediction reliability p_{ok} with regressors trained for two tracking algorithms and the two presented quality measures. p_{ok} denotes how often the prediction lies within the given accuracy range (or allowable error).

rent image, using the following steps: First, a very coarse initial pose is estimated around Harris interest points. Next, the initial pose is refined with the JD algorithm. Upon success, the exact pose and patch identity of reference templates in the current image are know, and the object pose can be derived. If the refinement step is unsuccessful or inaccurate, detection of the respective reference template fails. The performance of the system therefore extremely benefits from a template that supports the JD algorithm.

In our experiments, the system is tested on real-world video sequences, showing objects with a planar textured surface, see Fig. 6. The object pose is varied within a wide range and even driven beyond the known limitations of GEPARD. Ground truth data is available for the object pose in 3D world coordinates, using a commercial tracker system [16]. The pose is converted into 2D image coordinates, which are used to verify the tracking results. The LEDs installed on the object for the commercial tracker system are ignored by GEPARD.

We compare four different methods for the selection of reference templates: First, two naive selection schemes are considered. N reference templates are selected around the N strongest Harris interest points on the object ("strong, no rating"), or the most robust ones ("robust, no rating"). The latter method was used previously in GEPARD, see below. Second, we present results for template selection with our new proposed approach. Templates still need to be selected around Harris points for the initial pose estimation step, so we generate a set of candidate templates around interest points with a high strength or robustness. Out of this set, we select N templates with the highest score $\mathcal{S}$ ("strong, rating" and "robust, rating"). "Robust" interest points are found by synthetically warping the object texture and identifying points which appear in the majority of views. The method ensures a high rediscovery rate of Harris points, but may be too slow for real-time systems.

Tab. 2 shows detection results for six sequences of an average length of 400 frames each. Two relevant performance measures are given: The average number of successfully detected templates which were confirmed by the ground truth T_{good}, and the detection rate F_{good}, i.e. the share of frames for which at least one template was successfully found. The tested setup uses $N = 5$ reference patches, so $T_{good} \leq 5$. A typical size for the set of candidate templates is 15. Com-

Detection rate (F_{good}) in %	IP selection		Robust		Strong	
	Template Rating		No	Yes	No	Yes
	Sequence	mv	83	85	75	84
		sf_1	87	88	86	89
		sf_2	95	96	91	97
		mat_1	77	82	75	79
		mat_2	78	78	53	78
		pci	74	70	59	57
		Average	82	83	73	81

Templates per frame (T_{good})	IP selection		Robust		Strong	
	Template Rating		No	Yes	No	Yes
	Sequence	mv	2.49	2.81	2.17	2.35
		sf_1	2.60	3.39	3.38	3.65
		sf_2	3.25	3.50	3.45	3.60
		mat_1	2.63	2.93	2.30	2.34
		mat_2	2.13	2.37	1.35	1.85
		pci	1.75	1.81	1.64	1.43
		Average	2.48	2.80	2.38	2.54

Table 2. Comparison of GEPARD [9] performance with and without the proposed method for "Template Rating". Interest points ("IP") are selected using Harris response ("Strong") or using the slower robustness evaluation ("Robust", see text). The number of reference templates is set to 5, so $T_{good} \leq 5$.

paring "robust" interest point selection without and with rating, a clear increase of T_{good} of up to 0.8 is observed, showing the benefits of template rating. This translates to faster or more reliable object pose detection. For the "strong" interest point selection method, which is better suited for real-time applications, template rating increases the detection rate F_{good} by 8% on average. In some sequences naive template selection works considerably worse, and a larger performance gain of up to 25% is achieved by template rating. Only in one case (sequence "pci") a small performance drop is observed. This can be explained by the low variety of candidate templates, which are mainly located in the text block (see Fig. 6 bottom right). The Wilcoxon rank-sum test [10] applied to the number of detected templates per frame for all sequences combined confirms an improvement with our method at 2% significance level.

Some templates from the tested sequences are depicted in Fig. 7, along with their predicted score. Note how the system rejects templates with large uniform areas and structures which allow no reliable estimation of the image warp.

3.3. Runtime

Feature extractors were implemented in non-optimized C++ using the OpenCV library. Runtime of the four "top features" discussed in Sec. 2.3.4 is measured using a 2.60 GHz Intel Core 2 Duo system with 3 GB RAM. Rating one template takes 0.9 ms for feature extraction and score prediction. The individual extractors require between 0.03 ms and 0.6 ms each, whereat the DFT is the slowest.

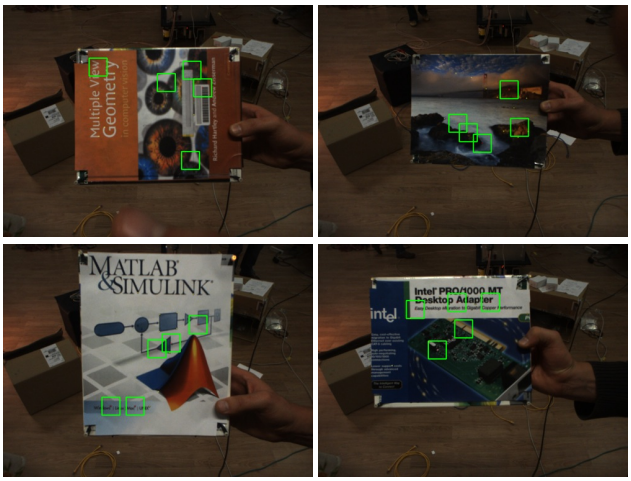

Fig. 6. Reference frames for sequences mv, sf_1, mat_2 and pci. Videos sf_2 and mat_1 have their own reference frames, but use the same textures as sf_1 and mat_2. Templates selected by the "strong, rating" experiment (Tab. 2) are indicated by a green box.

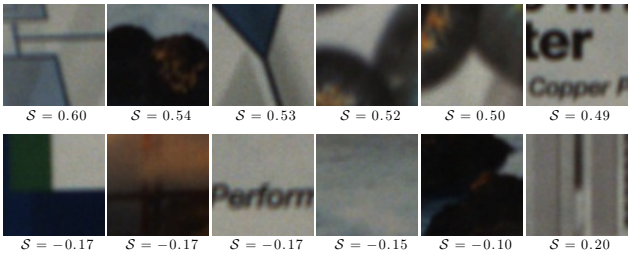

Fig. 7. Templates from tested sequences along with scores $\mathcal{S}$ (normalized to $[-1; 1]$). The first row shows good templates, whereas the second one shows bad ones rejected for tracking.

4. Conclusion

We proposed a predictor for tracking quality on the basis of specifically designed fast features. It allows to find good templates on a given texture for reliable tracking in real-time, which is especially important for applications with online template selection. A single template can be rated in less than 1 ms. During training, the approach uses a quality measure obtained from experiments which considers both convergence rate and and speed. A small set of four "top-features" is found to be sufficient for reliable prediction. These features are based on a uniformity measure of template subregions (Φ_{C_G}, Φ_{C_Σ}), the Sobel-filtered template ($\Phi_{S_{\Delta\theta}}$) and its spatial frequency spectrum (Φ_{FT_E}). The prediction error for Jurie and Dhome's tracker [11] is small (less than 10% for 95% of templates) and still acceptable when using the inverse compositional [1] tracking algorithm. We demonstrated our method on simulated tracking experiments as well as on real-world videos and showed its superiority to naive template selection approaches.

Acknowledgement This work was supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys* and by the Bayrische Forschungsstiftung. The authors would like to thank Ezio Malis (INRIA) and Vincent Lepetit (EPFL) for their advice and feedback.

References

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3), Feb. 2004.

[2] S. Benhimane, A. Ladikos, V. Lepetit, and N. Navab. Linear and quadratic subsets for template-based tracking. In *CVPR*, 2007.

[3] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IROS*, 2004.

[4] J. Buenaposada and L. Baumela. Real-time tracking and estimation of plane pose. In *ICPR*, 2002.

[5] G. Carneiro and A. Jepson. The distinctiveness, detectability, and robustness of local image features. In *CVPR*, 2005.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[7] F. Dellaert and R. Collins. Fast image-based tracking by selective pixel integration. In *ICCV Workshop of Frame-Rate Vision*, 1999.

[8] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10), 1998.

[9] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit. Real-time learning of accurate patch rectification. In *CVPR*, 2009.

[10] M. Hollander and D. Wolfe. *Nonparametric statistical methods*. Wiley, 2nd edition, 1999.

[11] F. Jurie and M. Dhome. A simple and efficient template matching algorithm. In *ICCV*, 2001.

[12] E. Loutas, N. Nikolaidis, and I. Pitas. Evaluation of tracking reliability metrics based on information theory and normalized correlation. In *ICPR*, 2004.

[13] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, 1981.

[14] O. Mac Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *CVPR*, 2010.

[15] K. Nickels and S. Hutchinson. Estimating uncertainty in ssd-based feature tracking. *Image and vision computing*, 20(1), 2002.

[16] PTI Inc. Visualeyez vz 4000. http://www.ptiphoenix.com/.

[17] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5), 2000.

[18] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[19] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. Stanford Artificial Project (talk), 1968.

[20] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *PAMI*, 31(4), 2009.