Detecting tracking errors via forecasting

ObaidUllah Khalid[1,2]
o.khalid@qmul.ac.uk
Andrea Cavallaro[1]
a.cavallaro@qmul.ac.uk
Bernhard Rinner[2]
bernhard.rinner@aau.at

[1] Centre for Intelligent Sensing
Queen Mary University of London
London, UK

[2] Inst. of Networked and Embedded Sys.
Alpen-Adria-Universitat Klagenfurt
Klagenfurt, Austria

Abstract

We propose a tracker-independent framework to determine time instants when a video tracker fails. The framework is divided into two steps. First, we determine tracking quality by comparing the distributions of the tracker state and a region around the state. We generate the distributions using Distribution Fields and compute a tracking quality score by comparing the distributions using the L_1 distance. Then, we model this score as a time series and employ the Auto Regressive Moving Average method to forecast future values of the quality score. A difference between the original and forecast returns an error signal that we use to detect a tracker failure. We validate the proposed approach over different datasets and demonstrate its flexibility with tracking results and sequences from the Visual Object Tracking (VOT) challenge.

1 Introduction

Detecting tracking errors is important to support self-aware systems and to correct [2] or to remove [17] failing trackers in a fusion framework. Validating the quality of a tracker over time can help determine time instants when these errors occur. The ability of the tracker to stay on target can be quantified by a tracking quality measure, which can be estimated using features, trajectories or both (hybrid approach). *Feature*-based track quality estimators (TQEs) exploit tracker-independent covariance descriptors [23], internal properties of the trackers such as the observation likelihood [19] or the spatial uncertainty of particle filters (PF) [2, 25]. *Trajectory*-based TQEs may use tracker correlation [11], target velocity [28] or a time-reversed approach [17, 33]. *Hybrid* approaches estimate tracking quality by combining multiple TQEs with a naive Bayes classifier [29] or by computing a weighted average of the quality scores [5]. These approaches are tuned to specific data [28], evaluate quality using heuristically determined thresholds [5, 23], depend upon specific trackers [2, 24, 25] or have a high computational cost [33]. Moreover, challenges such as motion blur, sudden changes in target motion and low resolution further make the task of detecting time instants of tracking failure difficult.

A TQE can be employed to detect tracking errors by analysing the temporal changes of the filter uncertainty [25] or by modelling these changes via a mixture of Gamma distributions [24]. However, these approaches are limited to the use of PFs. An example of a

Pages 140.1-140.12
DOI: https://dx.doi.org/10.5244/C.30.140

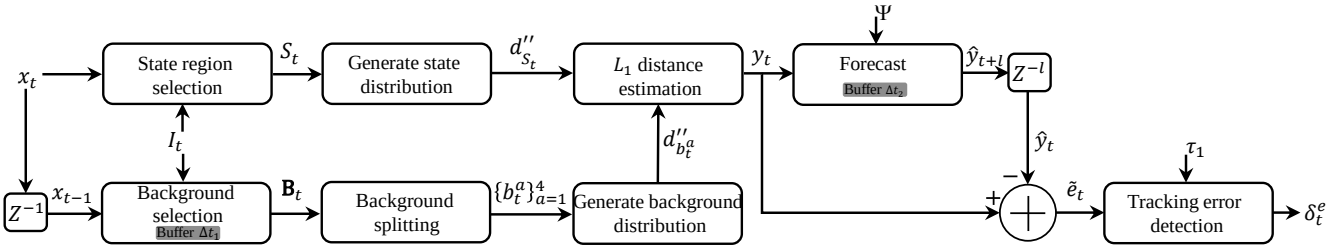

Figure 1: Block diagram of the proposed framework.

tracker-independent approach for the validation of tracking quality is comparing covariance descriptors (using colour features) over consecutive frames [23]. Nonetheless, this approach employs a heuristically set threshold for detecting tracking errors and is therefore limited to specific data. Using the reversibility property of Markov chains, tracking errors can be detected by comparing results of the tracker to those obtained when running the same tracker in the reverse temporal direction [33]. However, using a reverse tracker is computationally expensive and is affected by a decision latency.

In this paper we propose a failure detection framework that uses only the output of a tracker and a state-background discrimination approach to estimate tracking quality. The background region around the state is split into smaller regions of the same size as the state. We then determine the distributions of the state and of each background region using distribution fields (DF) [27], where DF represents a smoothed histogram of the image region composed of several layers (bins). We compare the state and background distributions to quantify the similarity between the two regions, thus producing a track quality score. Raw (noisy) values of the track quality score can have variable ranges for different sequences and trackers, thereby limiting the state-of-the-art (SOA) methods to specific sequences or trackers [23, 24, 25]. To address this problem and detect tracking errors, we model the score as a time series using the Auto Regressive Moving Average (ARMA) model and forecast future values of the time series. The difference between the original and the forecast generates a forecast error signal, which has a uniform range of values for any video data. We then detect significant changes (tracking errors) within the forecast error signal using an experimentally derived threshold. Figure 1 shows the block diagram of the proposed approach.

2 Track quality estimation

We perform background analysis to estimate the tracking quality. Background analysis is generally used for *foreground detection* (FD) [3] and *tracking-by-detection* (TD) [6, 13]. FD approaches generate over time a background model to separate moving objects from the scene [3], whereas TD approaches track the target by employing local search regions around the estimated state from the current time instant [9, 13].

Let $\mathbf{I} = \{I_t\}_{t=1}^{T}$ be an image sequence and $x_t = [u_t, v_t, w_t, h_t]$ be the tracker state at time $t = 1, \ldots, T$, where $[u_t, v_t]$ is the position, and w_t and h_t are the width and height of the bounding box of the target, respectively. Using past motion and position information of the state over a sliding temporal window, we select in I_t the background region $\mathbf{B}_t$ around the state region S_t defined by x_t. $\mathbf{B}_t$ encloses both the state and its surrounding background.

To select $\mathbf{B}_t$, we first predict the target position in the next frame. We use past information over a short sliding temporal window, Δt_1, to determine the average displacement and the direction of movement of the target (Figure 2(a)). Let $\vec{v}_{\Delta t_1}$ be the directional feature of a

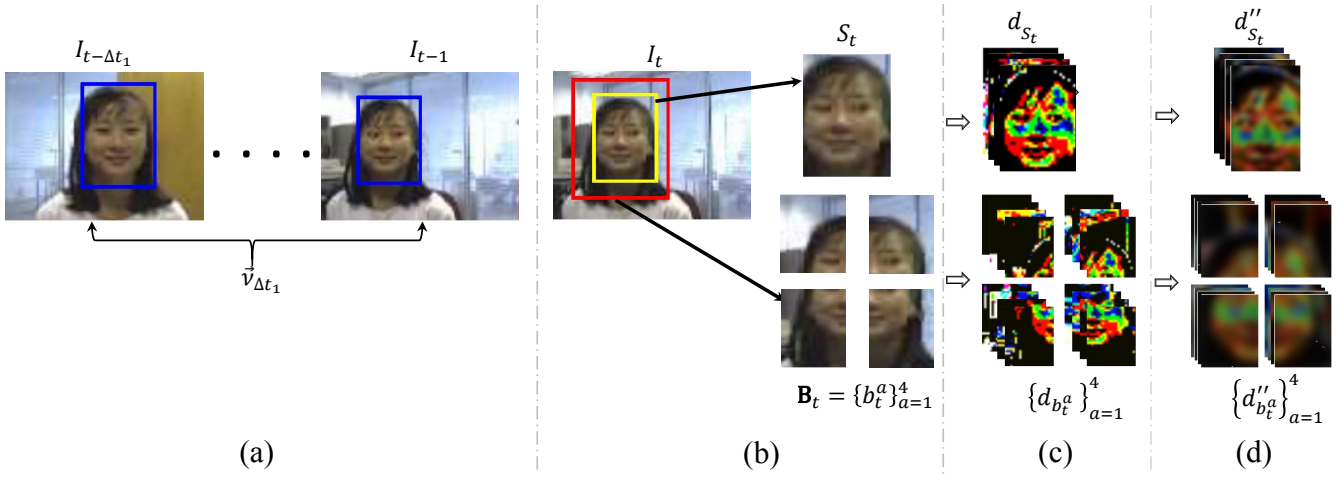

Figure 2: Background and state region selection. (a) $x_{t-\Delta t}$, ..., x_{t-1} (enclosed in the blue bounding boxes) and motion information $\vec{v}_{\Delta t_1}$ over a past temporal window Δt_1; (b) background region $\mathbf{B}_t$ (enclosed in the red bounding box) and state region S_t (enclosed in the yellow bounding box) selected at frame I_t; (c)-(d) distributions of $\mathbf{B}_t$ and S_t represented with colour DF [32].

tracker [1], which represents both the average displacement and direction of the target over Δt_1 and is computed as:

$$\vec{v}_{\Delta t_1} = \frac{1}{\Delta t_1} \sum_{t'=t-\Delta t_1}^{t-1} [u_{t'+1} - u_{t'}, v_{t'+1} - v_{t'}], \tag{1}$$

where the position of the target in I_t is predicted as $[\hat{u}_t, \hat{v}_t] = [u_{t-1}, v_{t-1}] + \vec{v}_{\Delta t_1}$. The background region $\mathbf{B}_t$, centred at $[\hat{u}_t, \hat{v}_t]$ with width and height of $\hat{w}_t = w_{t-1} + \lceil (w_{t-1} + h_{t-1})/4 \rceil$ and $\hat{h}_t = w_{t-1} + \lceil (w_{t-1} + h_{t-1})/4 \rceil$, respectively [9], where $\lceil . \rceil$ defines the ceil operation, is then selected from frame I_t.

We determine the distributions of the two regions using colour DF [32]. DFs have been employed for background subtraction [8, 30] as well as for tracking [10, 22, 27, 32] and combine the power of histograms and intensity gradients to preserve both visual information and the spatial structure of the image region [32]. A DF is a collection of probability distributions where each distribution defines the probability of a pixel taking the feature value (e.g. colour intensity).

Let $d_{S_t}(i, j, c, m)$ be the DF, where $i = 1, ..., h_t$ and $j = 1, ..., w_t$ define the pixel location in S_t, $c \in \{\text{R, G, B}\}$ is the feature channel and $m = 1, ..., M$ is the index of the layer. A DF is generated in three steps. The first step explodes the image into multiple layers (Figure 2(c)) resulting in a Kronecker delta function at each pixel location:

$$d_{S_t}(i, j, c, m) = \begin{cases} 1 & \text{if} \quad \lceil \frac{S_t(i,j,c)}{\lambda} \rceil = M, \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where λ, the size of each layer, is the ratio between the maximum feature value (e.g. 255 for an RGB channel) and M. The second step spatially spreads the information in $d_{S_t}(i, j, c, m)$ by convolving $d_{S_t}(i, j, c, m)$ with a 2-D Gaussian kernel h_{σ_1} over m as:

$$d'_{S_t}(m) = d_{S_t}(m) * h_{\sigma_1}, \tag{3}$$

where σ_1 is the standard deviation of h_{σ_1} and $*$ is the convolution operation. Finally, to better handle small variations in brightness and subpixel motion [27] $d'_{S_t}(m)$ is convolved with a 1-D Gaussian kernel h_{σ_2} (with standard deviation σ_2) over (i,j,c) (Figure 2(d)) as:

$$d''_{S_t}(i,j,c) = d'_{S_t}(i,j,c) * h_{\sigma_2}. \tag{4}$$

To compare the DF of $\mathbf{B}_t$ with $d''_{S_t}(i,j,c,m)$, we divide $\mathbf{B}_t$ into four smaller and equally sized regions b_t^a, where $a = 1,\ldots,4$, having width w_t and height h_t (Figure 2(b)). Then, using the same feature space, a distribution for each b_t^a ($d''_{b_t^a}(i,j,c,m)$) is computed using Equations (2) - (4), where S_t is replaced by b_t^a. We use the L_1 distance to compare $d''_{S_t}(i,j,c,m)$ and $d''_{b_t^a}(i,j,c,m)$ on each channel as:

$$\tilde{y}_t^a = \frac{1}{w_t\ h_t\ M} \sum_{c\in\{\mathrm{R,G,B}\}} \left(\omega_c \sum_{i=1}^{h_t} \sum_{j=1}^{w_t} \sum_{m=1}^{M} \left| d''_{S_t}(i,j,m) - d''_{b_t^a}(i,j,m) \right| \right), \tag{5}$$

where $|.|$ denotes the absolute value. We normalise the distance by the height and width of the state and the number of layers. The weight ω_c assigned to each channel is computed as:

$$\omega_c = \frac{\left| \mu_{S_t}^c - \mu_{b_t^a}^c \right|}{\sum_{c\in\{\mathrm{R,G,B}\}} \left| \mu_{S_t}^c - \mu_{b_t^a}^c \right|}, \tag{6}$$

where $\mu_{S_t}^c$ and $\mu_{b_t^a}^c$ are the mean R, G, B values for S_t and b_t^a, respectively. Weighting the colour channels allows us to exploit the most discriminative one(s) when comparing the two distributions.

The overall tracking quality score y_t is determined by quantifying the similarity between $\mathbf{B}_t$ and S_t as:

$$y_t = \frac{1}{4} \sum_{a=1}^{4} \tilde{y}_t^a, \tag{7}$$

where low (high) values of y_t indicate similarity (dissimilarity) between $\mathbf{B}_t$ and S_t.

3 Detecting tracking errors

We detect tracking errors by employing time series analysis to model $\mathbf{Y} = \{y_t\}_{t=1}^{T}$, a univariate discrete time series, for forecasting. Forecasting methods such as moving average models [16] have flat forecast functions and generally do not take past information into account. ARMA models [4] are built using past data and forecast using both past and present data. State-space models, such as the Kalman Filter, require the model of the time series to be known beforehand for forecasting [12]. Support Vector Machines [26] and neural network models [12] are more complex than ARMA for forecasting. A comprehensive survey of time series forecasting is presented in [12].

We employ ARMA to model $\mathbf{Y}$, where the difference between the forecast and the original returns a re-scaled signal, highlighting only the significant changes (tracking errors). ARMA(p,q) models are defined by their autoregressive (AR) and moving average (MA) orders $p = 0,\ldots,P$ and $q = 0,\ldots,Q$, respectively. Determining the right model requires identification of P and Q by a visual inspection of the auto-correlation function (ACF) and partial

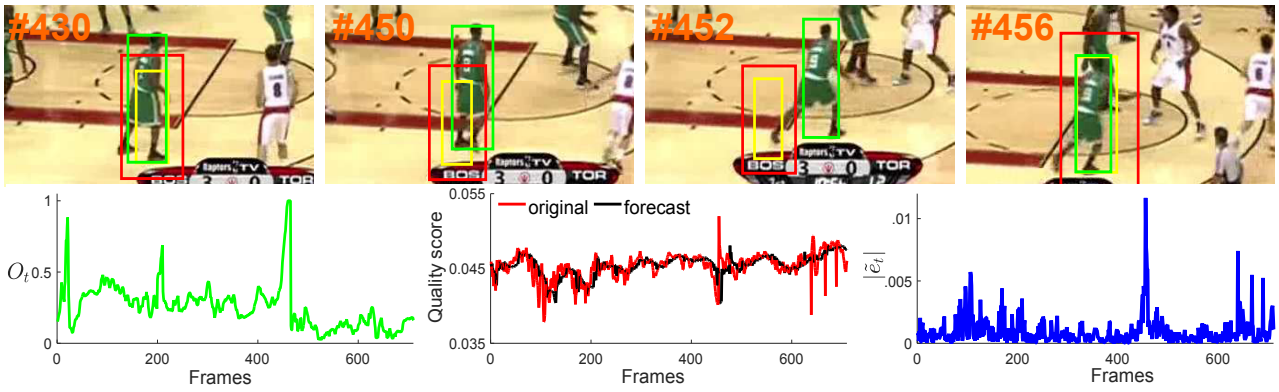

Figure 3: Example for track quality estimation with forecasting using the results produced by tracker DSST [7] on the VOT2014 sequence *basketball*. The tracker is re-initialised after an error at frame 452. Top row: Tracking results. Red, yellow and green represent $\mathbf{B}_t$, S_t and the ground-truth state, respectively. Bottom row: (a) Tracker performance measured as O_t (Section 4.1); (b) original y_t and its forecast $\hat{y}_{t+l}$; (c) forecast error signal $|\tilde{e}_t|$. The tracking error at frame 452 is reflected as a significant change within $|\tilde{e}_t|$.

ACF (PACF) of $\mathbf{Y}$ or by employing statistical tests such as the Akaike and Bayesian Information Criteria [12]. Using P, Q and the past data within a sliding temporal window Δt_2, the AR polynomial $\hat{\phi}(p)$, the MA polynomial $\hat{\theta}(q)$ and the constant $\hat{\beta}$ can be recursively estimated using techniques such as the conditional least squares or the maximum likelihood methods [4]. Using the parameters $\Psi = \left\{P, Q, \hat{\phi}(p), \hat{\theta}(q), \hat{\beta}\right\}$ forecasts are recursively computed over the forecast length $l \geq 1$ at time t as:

$$\hat{y}_{t+l} = \begin{cases} \sum_{p=0}^{P} \hat{\phi}(p)\ \hat{y}_{t+l-p} + \sum_{q=0}^{Q} \hat{\theta}(q)\ \varepsilon_{t+l-q} + \hat{\beta} & \text{for} \quad l < Q, \\ \sum_{p=0}^{P} \hat{\phi}(p)\ \hat{y}_{t+l-p} + \hat{\beta} & \text{otherwise}, \end{cases} \tag{8}$$

where $\varepsilon_t = y_t - \hat{y}_t$ is the estimation error (which is replaced by zero for $l > Q$, because it has not occurred yet) [4]. The forecasting error $\tilde{e}_{t+l} = y_{t+l} - \hat{y}_{t+l}$ determines the accuracy of a forecasting approach [12]: low (high) values indicate good (bad) forecasts.

Since the values of $\hat{y}_{t+l}$ are dependent on past values of y_t between $t - \Delta t_2$ and t, $|\tilde{e}_{t+l}|$ temporally smooths y_t. A significant change in the value of y_t is a tracking error, e_{t+l}, and computed as:

$$e_{t+l} = \begin{cases} 1 & \text{if} \, |\tilde{e}_{t+l}| \geq \tau_1, \\ 0 & \text{otherwise}, \end{cases} \tag{9}$$

where τ_1 is determined experimentally. Finally, if $\delta_t^e = e_{t+l} - e_{t+l-1}$, then $\delta_t^e = 1$ indicates when a tracking error first occurs.

4 Experimental evaluation

We compare the proposed approach with the SOA for detecting tracking errors and then test its flexibility with results and sequences from the VOT2014 challenge [18].

4.1 Experimental setup

We validate the proposed approach, Detect Tracking Errors using Forecasting (DTEF), using a PF-based tracker that employs sparse (intensity) features to generate the target appearance model and *maximum a posteriori* to estimate the target state and use the code publicly available from the author's web page[1] [31].

Dataset. For training we use 20 sequences from dataset D1 [17]. For testing we use 20 sequences from the Object Tracking Benchmark (OTB) dataset [34], namely: *CarDark*, *CarScale*, *Couple*, *Crossing*, *David* (300:500), *David3*, *Doll* (1:500), *FaceOcc1*, *Girl* (1:210), *Jogging*, *Liquor* (1:750), *MotorRolling*, *MountainBike*, *Singer1*, *Singer2*, *Subway*, *Tiger1*, *Walking*, *Walking2* and *Woman* (1:150). Sequences (first frame:last frame), are used with a reduced number of frames since the tracker does not recover the target after a failure in this point. The sequences cover indoor and outdoor scenarios containing tracking challenges such as pose, motion and illumination changes, occlusions, background clutter, motion blur and contain three target types, namely cars, people and faces. Sequences with less than 100 frames or where a tracking error occurs within the initial 15 frames are not used. Target initialisation for D1 and OTB can be found in http://www.eecs.qmul.ac.uk/~andrea/dtef.html.

Forecasting model. We achieve model building and parameter estimation using the MATLAB built-in *arima* and *estimate* functions, respectively. As finding the best fit for $\mathbf{Y}$ is out of the scope of this paper, we determine the values of $P = Q = 1$ by visual inspection of the ACF and PACF plots of $\mathbf{Y}$.

Experimental parameters. For both training and testing purposes, $\Delta t_1 = 10$ provides an optimal value to encode the average target displacement and direction. The standard deviation of the 2-D Gaussian kernel is set to $\sigma_1 = 1$ and 2 for the u and v directions, respectively, and $\sigma_2 = 0.625$ for the 1-D Gaussian kernel as in [27], while the number of layers, $M = 32$ provides a better discrimination between background and target distributions. For training over D1, we determine the optimal amount of past data required to build the forecast model using different values of Δt_2 as 10 and 20. The forecast length is varied as $l = 5$, 10, 25, 50 to determine the performance for both short-term and long-term forecasts. Finally, the threshold τ_1 was varied between 0.003 and 0.009 with a step size 0.001. For testing the approach over OTB, we use $\Delta t_2 = 20$, $l = 5$ and $\tau_1 = 0.004$ based on the training results.

Methods under comparison. We compare DTEF with two variations of the proposed approach: RAW and NAIVE; one SOA method for tracker error detection: Covariance Features (CovF) [23] and two SOA feature descriptors employed for video tracking: RGB Histograms (RgbHist) and RGB+LBP Histograms (RLHist) [21]. NAIVE detects tracking errors by forecasting y_t using the Naive forecasting model [12] that forecasts values equal to the last observed value ($\hat{y}_{t+l} = y_t$). Based on the training over D1, threshold for NAIVE = 0.004. RAW detects tracking errors using raw y_t values, where threshold = 0.039 is based on training over D1. CovF [23] employs a 5-dimensional target descriptor based on the colour and position values and compares them within consecutive frames to determine tracking quality. For error detection, the threshold for CovF is set to 2.3 as in [23]. RgbHist and RLHist are employed for tracking failure detection and trained over D1 to select threshold = 0.88 for both RgbHist and RLHist. For DTEF, NAIVE and CovF the tracking error is detected for values above their respective thresholds, while for RAW, RgbHist and RLHist for values below their respective thresholds.

[1] http://faculty.ucmerced.edu/mhyang/project/tip13_prototype/TIP12-SP.htm

Evaluation measures. We measure the tracking error by comparing the tracker output x_t with the ground-truth (GT) data as [24]:

$$O_t = 1 - \frac{2|A_t^x \bigcap A_t^{\mathrm{GT}}|}{|A_t^x| + |A_t^{\mathrm{GT}}|}, \tag{10}$$

where A_t^x and A_t^{GT} represent the area in pixels of the estimated, x_t, and GT target locations, respectively; $|A_t^x \bigcap A_t^{GT}|$ is their spatial overlap in pixels. $O_t \in [0,1]$: values close to 0 (1) indicate high (low) tracking performance. Time instants when O_t changes from success ($O_t < 1$) to failure ($O_t = 1$) [24] are determined by the GT transitions, ($\delta_t^O = O'_t - O'_{t-1}$, if $O'_{t-1} = 0$ and $O'_t = 1$, otherwise $\delta_t^O = 0$), where O'_t is determined as:

$$O'_t = \begin{cases} 1 & \text{if} \quad O_t = 1, \\ \\ 0 & \text{if} \quad O_t < 1, \end{cases} \tag{11}$$

For comparing the performance in detecting tracking errors, we use the number of true positives (n_{TP}), false positives (n_{FP}), false negatives (n_{FN}) and true negatives (n_{TN}). We compute the false positive rate $FPR = \frac{n_{FP}}{n_{FP}+n_{TN}}$, precision ($P$), recall ($R$) and the F-score [17]. n_{TP} (n_{FN}) indicate whether the decisions, δ_t^e, of the proposed method correspond correctly (incorrectly) to the failure decisions generated by δ_t^O. Similarly, a correct (incorrect) match of the successful decisions between δ_t^e and δ_t^O is determined by n_{TN} (n_{FP}). A tolerance window of ± 5 frames is used to match δ_t^e with each δ_t^O.

Let a generic z_t represent $|\tilde{e}_t|$, y_t or the tracking performance scores generated by the SOA methods, and $\mathbf{Z} = \{z_t\}_{t=1}^{T}$ be the corresponding time-series. We normalise each z_t as:

$$z'_t = \frac{z_t - min(\mathbf{Z})}{max(\mathbf{Z}) - min(\mathbf{Z})}, \tag{12}$$

to enable the comparison of variations of the corresponding values over the whole dataset.

4.2 Tracking error detection

We first compare DTEF with NAIVE and RAW, and then with CovF, RgbHist and RLHist on the OTB dataset.

While the values of y_t vary across sequences (see Figure 4(a)-(d)), forecasting enables us to generate a signal with the same range of values for the whole dataset. In *CarDark* an illumination variation and background clutter cause a tracking error between frames 270 and 280, while in *Crossing* a tracking error occurs due to scale variations. RAW achieves lower P than DTEF and NAIVE (Table 1), which detect these errors with their respective forecasting approaches. RAW achieves a lower FPR and hence a better P than DTEF and NAIVE, because for sequences where the tracker fails to re-acquire the target after an error, the values of y_t fall below the threshold. However, since the tracker is not stopped (or re-initialised), y_t generates false significant changes, which are recorded as tracking errors by DTEF and NAIVE (see Figure 4(b) and (d)). Since NAIVE forecasts values at time t for the complete forecast length (l), it may suppress some of the false significant changes (false positives) of y_t resulting in a better P than DTEF. However, this behaviour also results in a lower R for NAIVE. The R of DTEF outperforms that of RAW and NAIVE by 76% and 7%, respectively, hence DTEF achieves a better F-score.

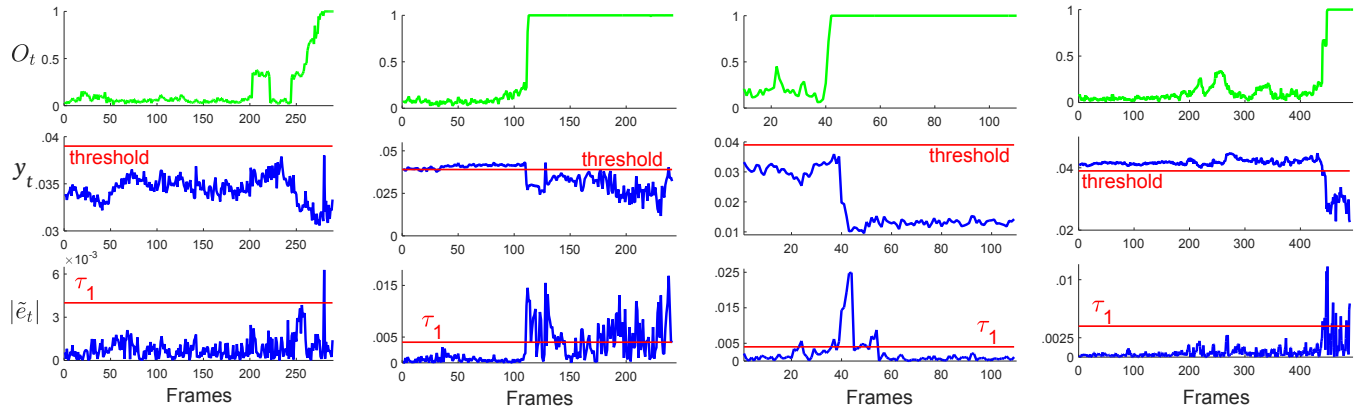

Figure 4: Example of variation of the track quality score, y_t, for (a) *CarDark*, (b) *CarScale*, (c) *Crossing*, (d) *Doll*. Top row: tracker performance measured as O_t. Middle row: track quality score y_t (blue line). Bottom row: forecast error $|\tilde{e}_t|$ (blue line). τ_1 and 'threshold' are used to detect time instants when the tracker fails for DTEF and RAW, respectively.

DTEF outperforms CovF, RgbHist and RLHist in terms of F-score, giving an overall improvement of 23%, 31% and 36% compared to CovF, RgbHist and RLHist, respectively. CovF, RgbHist and RLHist have a variable range of values over the dataset that leads to false positive and false negative tracking errors. Although CovF detects the same number of tracking errors as DTEF, a lower P results in lower F-score. RgbHist and RLHist are affected by these variations in terms of R. Furthermore, CovF, RgbHist and RLHist have similar failure modes to DTEF since their descriptors use colour as their primary feature. CovF achieves better results than RgbHist and RLHist, possibly because it uses position information as well, while RLHist improves over RgbHist, possibly due to the additional LBP features. Sample results for *CarDark* are shown in Figure 5.

DTEF generates false positive tracking errors when RAW generates significant changes after a tracker has failed. Furthermore, DTEF fails to detect tracking errors when RAW does not generate a change in y_t due to background clutter (in *Liquor* and *MountianBike*), occlusions (in *Jogging*), sudden background lighting changes (in *Singer1*) and fast target rotation and background clutter (in *MotorRolling*).

	DTEF	NAIVE	RAW	CovF	RgbHist	RLHist
P	.110	.111	**.122**	.087	.083	.078
R	**.714**	.667	.405	**.714**	.595	.667
F	**.191**	.190	.188	.155	.146	.140
FPR	.037	.035	**.019**	.048	.042	.051
$\mu \pm \sigma$	.17±.18	.15±.17	.60±.23	.30±.17	.67±.16	.61±.16

Table 1: Comparison of tracking error detection performance in terms of precision (P), recall (R), F-score (F) and false positive rate (FPR). The results are presented as total values over the OTB dataset. The best results are indicated by bold font. The last row shows the mean±standard deviation of z'_t. Key — DTEF: Detect Tracking Errors using Forecasting; NAIVE: error detection by forecasting y_t via the Naive forecasting model [12]; RAW: error detection using raw y_t values; CovF: Covariance Features [23]; RgbHist: RGB Histogram [21]; RLHist: RGB+LBP Histogram [21].

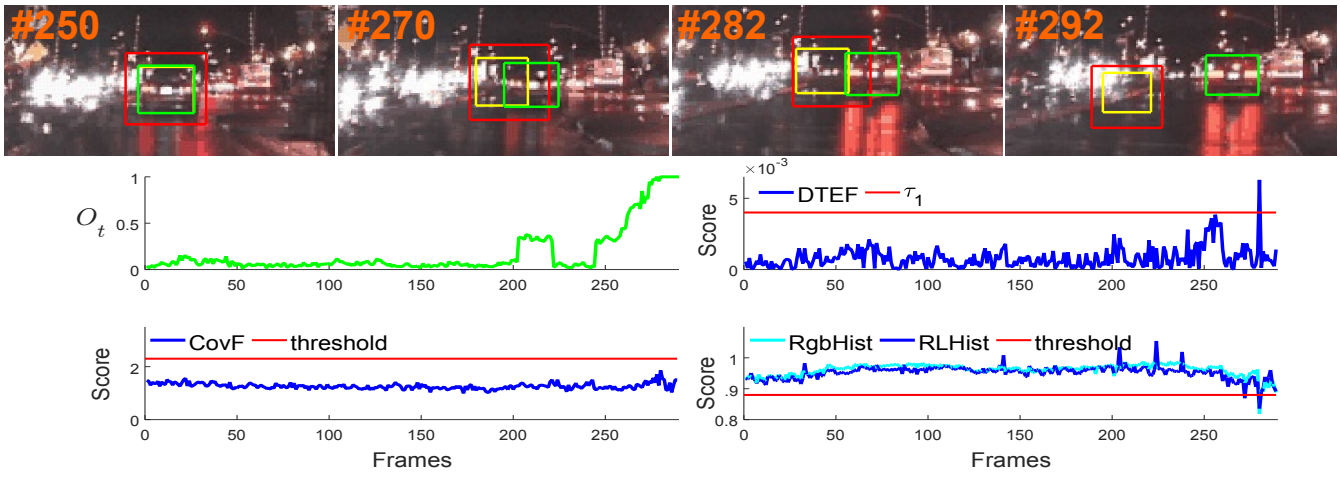

Figure 5: Example of track quality scores for *CarDark*. First row: Sample tracking results: the red, yellow and green bounding boxes represent $\mathbf{B}_t$, S_t and the ground-truth state, respectively. Second row: tracker performance measured as O_t (left) and $|\tilde{e}_{t+l}|$ score measured with DTEF (right). Third row: track quality score measured with CovF (left); and track quality score measured with RgbHist and RLHist (right).

	DSST						SAMF					
	DTEF	NAIVE	RAW	CovF	RgbHist	RLHist	DTEF	NAIVE	RAW	CovF	RgbHist	RLHist
P	**.154**	.124	.080	.077	.136	.135	.118	.078	.053	.059	.106	**.126**
R	**.700**	.326	.395	.465	.488	.651	**.488**	.220	.244	.342	.415	**.488**
F	**.252**	.180	.135	.132	.213	.224	.190	.115	.087	.100	.168	**.200**
FPR	.016	**.010**	.020	.024	.013	.018	.015	**.011**	.018	.022	.014	.014
$\mu \pm \sigma$	.18±.17	.16±.16	.54±.19	.35±.17	.66±.16	.62±.14	.18±.17	.16±.16	.54±.19	.34±.17	.64±.16	.63±.13
	KCF						PLT_14					
	DTEF	NAIVE	RAW	CovF	RgbHist	RLHist	DTEF	NAIVE	RAW	CovF	RgbHist	RLHist
P	.158	.095	.070	.093	**.183**	.145	**.065**	.057	.041	.029	.016	.045
R	**.535**	.256	.302	.419	.488	.395	.435	.304	**.478**	.217	.130	.261
F	.243	.138	.114	.152	**.266**	.213	**.113**	.096	.076	.051	.028	.076
FPR	.012	.010	.017	.017	**.009**	.010	.014	**.012**	.025	.017	.019	.015
$\mu \pm \sigma$	.16±.16	.16±.16	.55±.18	.36±.17	.68±.15	.64±.13	.19±.17	.17±.16	.57±.19	.41±.17	.61±.17	.61±.15

Table 2: Comparison of tracking error detection performance in terms of precision (P), recall (R), F-score (F) and false positive rate (FPR). The results are presented as total values over the whole VOT2014 dataset. The best results are indicated by bold font. The last row for each tracker shows the mean±standard deviation of z'_t. Key — DSST: Discriminative Scale Space Tracker [7]; KCF: Kernelized Correlation Filter [15]; SAMF: Scale Adaptive KCF tracker [20]; PLT_14: Pixel based LUT Tracker [14]; DTEF: Detect Tracking Errors using Forecasting; NAIVE: error detection by forecasting y_t via the Naive forecasting model [12]; RAW: error detection using raw y_t values; CovF: Covariance Features [23]; RgbHist: RGB Histogram [21]; RLHist: RGB+LBP Histogram [21].

4.3 VOT Results

Finally, we analyse the flexibility of DTEF via an experimental comparison with other methods using results from four trackers (DSST [7], SAMF [20], KCF [15], PLT_14 [14]) and sequences from the VOT2014 challenge [18] (see Table 2). Note that VOT re-initialises trackers after failure (O_t=1): the tracker is stopped for the subsequent five frames and then is re-initialised with the ground truth. In order to compensate for the missing tracking results, we keep for these five frames the same tracking result obtained when the tracker fails.

The re-initialisation of the trackers allows DTEF to reduce its FPR and to achive a better F-score than RAW. Overall, DTEF improves by 51% and 94% in terms of F-score over both RAW and NAIVE, respectively. Using the forecast error signal allows DTEF to detect tracking errors that are not detected by RAW. DTEF detects more tracking errors than CovF, RgbHist and RLHist, and achieve the best R values for all the four trackers and a better F-score for trackers DSST and PLT_14, indicating that the trained threshold, τ_1, is applicabile to various datasets. However, RgbHist and RLHist achieve a better F-score for KCF and SAMF, respectively, due to a smaller FPR. All the approaches achieve their best results for DSST followed by KCF, SAMF and PLT_14.

5 Conclusion

We presented a tracker-independent framework to estimate tracking quality using a tracker state-background discrimination approach. We used time series forecasting to model the track quality score while minimising noise. The difference between the original and forecast values returns a forecast error signal that we use to detect tracking errors. The use of the forecast error signal improves both the precision and recall of the proposed approach compared to using the raw values of the track quality score. We validated the proposed approach using publicly available datasets and demonstrated its flexibility using selected sequences from the OTB benchmark dataset and tracking results and sequences from the VOT challenge.

As future work, we aim to use the background analysis technique to explore upcoming positions of the target in the sequence and to help predict tracking errors. Furthermore, in order to remove the dependence of the proposed approach on thresholds, we will model the probability distribution of the forecasting error signal.

Acknowledgements

O. Khalid was supported by the EACEA Agency of the European Commission under the Erasmus Mundus Joint Doctorate ICE, FPA n. 2010-0012.

References

[1] N. Anjum and A Cavallaro. Multifeature object trajectory clustering for video analysis. *IEEE Trans. Circuits Syst. Video Technol.*, 18(11):1555–1564, Nov 2008.

[2] T.A. Biresaw, A. Cavallaro, and C.S. Regazzoni. Tracker-level fusion for robust bayesian visual tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 25(5):776–789, May 2015.

[3] T. Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Comput. Science Review*, 11-12:31 – 66, May 2014.

[4] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis: Forecasting and control*. Wiley, 2008. ISBN 9780470272848.

[5] D.P. Chau, F. Bremond, and M. Thonnat. Online evaluation of tracking algorithm performance. In *Int. Conf. Crime Detection and Prevention*, pages 1–6, Dec 2009.

[6] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *IEEE Int. Conf. on Comput. Vis.*, Dec 2015.

[7] M. Danelljan, G. Häger, F.S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *Proc. British Mach. Vis. Conf.*, Sep 2014.

[8] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *European Conf. Comput. Vis.*, pages 751–767, Apr 2000.

[9] J. Fan, X. Shen, and Y. Wu. What are we tracking: A unified approach of tracking and recognition. *IEEE Trans. Image Process.*, 22(2):549–560, Feb 2013.

[10] M. Felsberg. Enhanced distribution field tracking using channel representations. In *IEEE Int. Conf. Comput. Vis. Workshops*, pages 121–128, Dec 2013.

[11] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann. Symbiotic tracker ensemble towards a unified tracking framework. *IEEE Trans. Circuits Syst. Video Technol.*, 24(7):1122 – 1131, Jul 2014.

[12] J.G. Gooijer and R.J. Hyndman. 25 years of time series forecasting. *Int. Jour. Forecasting*, 22(3):443 – 473, Jul 2006.

[13] S. Hare, A. Saffari, and P.H.S. Torr. Struck: Structured output tracking with kernels. In *Int. Conf. Comput. Vis.*, pages 263–270, Nov 2011.

[14] C. K. Heng, S. Yokomitsu, Y. Matsumoto, and H. Tamura. Shrink boost for selecting multi-lbp histogram features in object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3250–3257, Jun 2012.

[15] J.F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):583–596, Mar 2015.

[16] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *Int. Jour. Forecasting*, 20(1):5 – 10, Jan 2004.

[17] O. Khalid, J. C. SanMiguel, and A. Cavallaro. Multi-tracker partition fusion. *IEEE Trans. Circuits Syst. Video Technol.*, 2016. doi: 10.1109/TCSVT.2016.2542699.

[18] M. Kristan, R. Pflugfelder, A. Leonardis, et al. The visual object tracking VOT2014 challenge results. In *Proc. European Conf. Comput. Vis.*, pages 191–217, Sep 2014.

[19] J. Kwon and K. Lee. Tracking by sampling and integrating multiple trackers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1428–1441, Jul 2014.

[20] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *Proc. of European Conf. Comput. Vis.*, pages 254–265, Sep 2014.

[21] J. Ning, L. Zhang, D. Zhang, and C. Wu. Robust object tracking using joint color-texture histogram. *Int. Journal Pattern Recog. Artificial Intell.*, 23(07):1245–1263, Feb 2009.

[22] J. Ning, W. Shi, S. Yang, and P. Yanne. Visual tracking based on distribution fields and online weighted multiple instance learning. *Image Vis. Computing*, 31(11):853 – 863, Nov 2013.

[23] J.C. SanMiguel and A. Calvo. Covariance-based online validation of video tracking. *IEEE Elec. Lett.*, 51(3):226–228, Feb 2015.

[24] J.C. SanMiguel and A. Cavallaro. Temporal validation of particle filters for video tracking. *Comput. Vis. Image Understanding*, 131(0):42 – 55, Feb 2015.

[25] J.C. SanMiguel, A. Cavallaro, and J.M. Martinez. Adaptive online performance evaluation of video trackers. *IEEE Trans. Image Process.*, 21(5):2812–2823, May 2012.

[26] N.I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intell. Magazine*, 4(2):24–38, May 2009.

[27] L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1910–1917, Jun 2012.

[28] K. Shearer, K.D. Wong, and S. Venkatesh. Combining multiple tracking algorithms for improved general performance. *Pattern Recog.*, 34(6):1257–1269, Jun 2001.

[29] C. Spampinato, S. Palazzo, and D. Giordano. Evaluation of tracking algorithm performance without ground-truth data. In *IEEE Int. Conf. Image Process.*, pages 1345–1348, Sep 2012.

[30] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, Aug 2000.

[31] D. Wang, H. Lu, and M-H. Yang. Online object tracking with sparse prototypes. *IEEE Trans. Image Process.*, 22(1):314–325, Jan 2013.

[32] Y. Wang, H. Chen, S. Li, J. Zhang, and C. Gao. Object tracking by color distribution fields with adaptive hierarchical structure. *The Visual Comput.*, pages 1–13, Nov 2015.

[33] H. Wu, A.C. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1443–1458, Aug 2010.

[34] Y. Wu, J. Lim, and M-H. Yang. Online object tracking: A benchmark. In *IEEE Conf. Comput. Vis. Pattern Recogn.*, pages 2411–2418, Jun 2013.