Long-Term Tracking Through Failure Cases

Karel Lebeda[1], Simon Hadfield[1], Jiri Matas[2], Richard Bowden[1]
[1]CVSSP, University of Surrey, Guildford, GU2 7XH, United Kingdom
[2]CMP, Czech Technical University, Karlovo namesti 13, Prague, 121 35, Czech Republic
{k.lebeda, s.hadfield, r.bowden}@surrey.ac.uk, matas@cmp.felk.cvut.cz

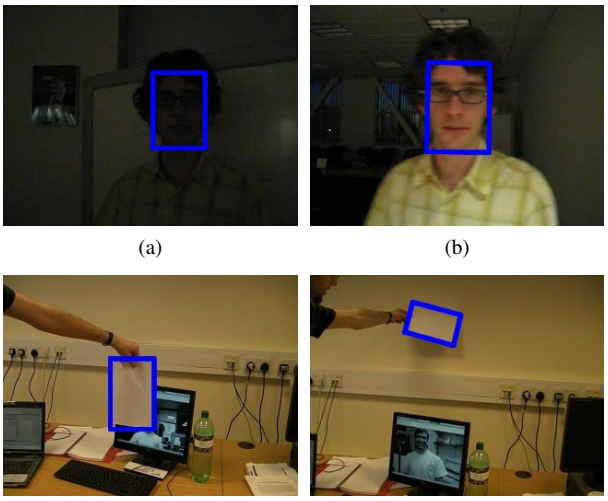
Abstract

Long term tracking of an object, given only a single instance in an initial frame, remains an open problem. We propose a visual tracking algorithm, robust to many of the difficulties which often occur in real-world scenes. Correspondences of edge-based features are used, to overcome the reliance on the texture of the tracked object and improve invariance to lighting. Furthermore we address long-term stability, enabling the tracker to recover from drift and to provide redetection following object disappearance or occlusion. The two-module principle is similar to the successful state-of-the-art long-term TLD tracker, however our approach extends to cases of low-textured objects.

Besides reporting our results on the VOT Challenge dataset, we perform two additional experiments. Firstly, results on short-term sequences show the performance of tracking challenging objects which represent failure cases for competing state-of-the-art approaches. Secondly, long sequences are tracked, including one of almost 30 000 frames which to our knowledge is the longest tracking sequence reported to date. This tests the re-detection and drift resistance properties of the tracker. All the results are comparable to the state-of-the-art on sequences with textured objects and superior on non-textured objects. The new annotated sequences are made publicly available.

1. Introduction

Visual tracking is a field of computer vision which has been thoroughly studied over the years. Many approaches have been proposed including tracking via local optimisation, regression, detection, segmentation, generative models and online learning. Long-term trackers attempt to model and adapt to changes in object appearance over time, using multiple observations to enrich their representations. This in turn leads to drift due to the difficulty of unsupervised learning. Recent approaches overcome this through a combination of detection combined with local search and intelligent online update strategies, which can compensate for drift via redetection. As such, consistency of appearance is extremely important, because radical changes can cause tracking failure or corruption of the model. The notion of appearance typically relies on texture or other strong visual attributes. However, there are a whole range of scenarios where sufficient texture is either absent or highly variable due to changes in pose or scene illumination.



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

Figure 1: Examples from challenging sequences.

Figure 1 shows example frames from two challenging sequences which typically lead to tracking failures. Fig. 1a and 1b show images from Ross *et al*. [15] where the initial target face is so dark that visual features are almost indistinguishable to the human eye. However, as shown by the bounding box, the proposed LT-FLO tracker is capable of tracking the entire sequence (see supplementary material for a results video). Even Ross *et al*. do not track this sequence from the start, only beginning at the 300th frame (shown in 1b) where the visual appearance from lighting is more consistent with the remainder of the video. Similarly Fig. 1c–1d show a texture-less object (a single piece of white paper on a light background). Trackers which rely on texture or appearance fail on this sequence. Again the bounding box shows LT-FLOs ability to successfully track

in this scenario. Furthermore, we demonstrate LT-FLO successfully tracking a sequence of over 29 000 frames which to our knowledge is the longest sequence reported to date.

Many recent state-of-the-art trackers are based either on an extension to the Lucas-Kanade tracker [12] (or connected group of trackers, *e.g.* [2]) or on learning texture-related classifiers for *tracking by detection* [3, 7, 5]. Almost all approaches rely upon strong appearance cues, leading to failure in scenes such as in Fig. 1c.

Cehovin *et al.* [2] use a *coupled-layer* approach. The local layer consists of independently tracked patches, constraining the appearance of object components. The global layer models object colour, shape and motion. Consistency of local trackers is enforced; however changes in shape are possible, allowing the LGT (Local-Global Tracker) to track highly non-rigid objects.

Kalal *et al.* [9] combine tracking and detection in their TLD (Tracking-Learning-Detection) algorithm. While the *tracker* estimates the frame-to-frame motion, the *detector* treats every frame as independent (as in a tracking-by-detection scenario). Positive and negative examples are *learned* according to the (dis-)agreement of these two components, improving further detection. Explicit modelling of failures for both components coupled with independent detection inherently makes this tracker suitable for *long-term tracking*, with necessary drift-resistance and redetection after full occlusions.

There have been previous attempts to decrease the reliance on object texture, including previous use of edge-based features. The aperture problem (see Fig. 2) renders these spatially unstable, as neighbouring pixels along the direction of the edge are indistinguishable. However, such features are still valuable: *e.g.* Smith *et al.* used lines in a visual SLAM framework [16]. For tracking, Tsin *et al.* [17] fit line segments modelling the object to detected edge-points. The same approach was used in [4, 6], when searching for a pose of a wireframe-represented, user-specified, 3D object. Our approach uses similar principles, but the object model is learned in a completely unsupervised manner.

Most recently, Lebeda *et al.* [10] proposed an approach that overcomes the aperture problem by estimating the geometry from edge-based line correspondences. To cope with situations involving insufficient point-to-point correspondences, line-to-line correspondences are used. These are robust to shifts along the edge. The tracker is dubbed FLOtrack – FeatureLess Object tracker. This article is partly inspired by their approach.

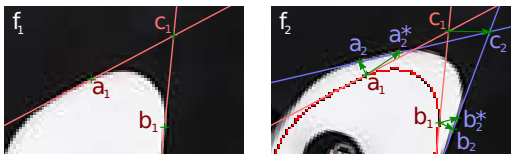
In this work we employ line correspondences within a robust motion estimation framework for frame-to-frame tracking. We also propose an approach to learning the manifold of observed object poses in a probabilistic manner. This probability is then directly used to reject "impossible" poses.

As a second major contribution we propose a redetection strategy that identifies when a failure or object disappearance occurs and uses the online model of appearance, in conjunction with the learnt pose manifold, to relocate the object. This gives the tracker long-term stability, which combined with its inherent drift resistance renders it suitable for tracking long video sequences with full occlusions. It is the first long-term edge-based tracker. We dub the tracker LT-FLOtrack (Long-Term FeatureLess Object tracker).

Further we introduce several improvements to the operational speed and robustness of the algorithm, giving a marked performance increase over FLOtrack. Finally, we make the new long-term sequences publicly available including manual ground-truth annotations.[1]

The LT-FLOtrack algorithm is introduced in section 2. It consists of two parts shown in sections 3 and 4. Section 5 describes the probability density estimation and section 6 the redetection scheme. Our claims are experimentally validated in section 7 and conclusions are drawn in section 8.

2. Texture-less Tracking



(a) (b)

Figure 2: Establishing line correspondences regardless of the aperture problem.

The task of the tracker is to find a pose P_t of the object in frame f_t. The tracked object is represented as a set of edge points (locations of locally maximal gradient). The fundamental elements for tracking are tentative correspondences of lines tangential to isocontours. Fig. 2a shows two points $\{\mathrm{a}_1, \mathrm{b}_1\}$ identified on the contour of an object and their tangent lines. Attempting to locate the motion of these points in the next consecutive frame is ill-defined. Fig. 2b shows that a local search normal to the edge direction incorrectly identifies correspondences $\{\mathrm{a}_2, \mathrm{b}_2\}$ which are shifted along the contour, instead of the true correspondence $\{\mathrm{a}_2^*, \mathrm{b}_2^*\}$. However, under the assumptions of a small shift between two consecutive frames and a local linearity of the edges, these points generate the same corresponding lines. By using the line intersection c_2 and its motion from the intersection in the previous frame (c_1), transformations can be calculated using edge features while overcoming the aperture problem.

[1] http://cvssp.org/data/YTLongTrack/

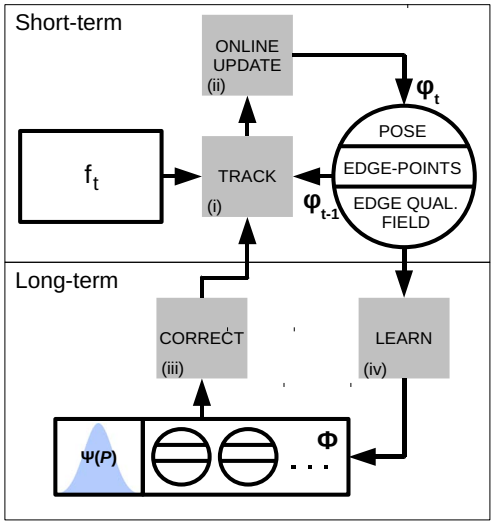


Figure 3: Overview of the LT-FLOtrack algorithm.

Fig. 3 shows an overview of the LT-FLOtrack. It consists of two modules performing different tasks. The first is a *short-term tracker*, which finds line correspondences and estimates frame-to-frame transformations (block (i)). It then *updates* knowledge about the object in each frame (block (ii)), including the positions of good edges to track and observed edge stability (*edge quality field*). The short-term tracker is formalised in Alg. 1.

The second module maintains *long-term relations*. In cases of low confidence within the short-term tracker, a procedure to *correct* the pose is performed (block (iii)). Alternatively, if short-term tracking leads to a correct pose estimate, its state is stored (block (iv)) for future corrections. Block (iv) also includes updates to the observed pose probability distribution in every frame.

3. Short-term Tracker

Algorithm 1 The Short-term Tracker

1: $Q_1 \leftarrow$ initialise point quality field
2: **for** $t = 2 \rightarrow F$ **do**
3: $p_t \leftarrow$ generate edge-points (f_{t-1})
4: $q'_t \leftarrow$ find tentative correspondences (f_t)
5: $S'' \leftarrow$ estimate transformation by RANSAC (p_t, q'_t)
6: $q_t \leftarrow$ find tentative correspondences $(f_t$, init by $S'')$
7: $S' \leftarrow$ estimate transformation by RANSAC (p_t, q_t)
8: $S \leftarrow$ refine transformation (S', Q_{t-1})
9: $Q_t \leftarrow$ update point quality field (p_t, S, Q_{t-1})
10: **end for**

To find the pose P_t, the short-term tracker estimates a (similarity) transformation S, such that $P_t = S(P_{t-1})$. Algorithm 1 describes this short-term part of LT-FLOtrack (F is the total number of frames processed). Lines 4 to 8 correspond to the block (i) and lines 3 and 9 refer to the block (ii) of Fig. 3. In each frame, a set of edge-points p_t is generated (line 3). Successfully matched correspondences (inliers to S) from the last frame (q_{t-1}) are retained and new edge-points are generated to keep a stable number of correspondences. As this number has a significant impact on execution times, it should be as low as possible, while keeping the results stable. To estimate a suitable number of correspondences for a given sequence, we propose a data-driven method, which is described and experimentally justified in section 3.1. The next step is to determine which edges are suitable for tracking. These should be invariant to the brightness changes and evenly distributed on the object, *e.g.* strong edges should be selected when possible and weaker edges only in regions of low contrast. We use a procedure, which iteratively searches for strong nearby edges, from randomly selected positions r along a line normal to the edge direction [6, 10]. As this uses only the image gradient, it is dubbed an *unguided edge search*. The selected edge-points are defined as

$$p_t = \underset{p'_t \in r + \lambda \vec{\nabla} f(r)}{\arg\max} |\nabla f_{t-1}(p'_t)| \cdot \exp(-\lambda^2/\sigma^2) \,, \qquad (1)$$

where λ is the distance along gradient direction. The maximum λ and the scale σ are based on the object size.

When establishing correspondences (line 4), another edge search is required, however the task is different. Instead of locally strong edges, we are looking for edges similar to those from the previous frame. As such, a *guided* (using information from the previous frame) *edge search* is used. This searches for positions with similar gradient angle and local appearance. The search starts at the locations of edge-points from the previous frame and multiple search lines are used (normal to the edge in the last frame and offset by $\pm\frac{\pi}{10}$). The tentative edge-point correspondences (p_t, q'_t) are then transformed to line-to-line correspondences using the gradient direction: $l_t = p_t + \lambda\vec{n}; \lambda \in (-\infty; \infty)$, where $\vec{n}$ is normal to $\vec{\nabla} f(p_t)$. The same process is done for q'_t.

LO-RANSAC [11] uses these to estimate a geometric transformation S'' between the two frames (line 5), maximising *image evidence* E_t [10]. This is defined as the average fit of edge-points from the previous frame to the edges in the current frame. The computation of image evidence is based on an *oriented Chamfer distance* [14] as

$$E_t(S') = \frac{1}{N_t} \sum_{p_t} e_{p_t} \cdot \Lambda \,, \qquad (2)$$

$$e_{p_t} = \frac{\cos(\Delta\alpha)/2 + 1/2}{1 + d(S''(p_t))} \,, \qquad (3)$$

where $d(\cdot)$ is Euclidean distance of a point to the nearest Canny's edge, $\Delta\alpha$ is difference between the gradient angle of a point and its nearest edge (taking rotation induced

Figure 4: Example of an image and its edge quality field.

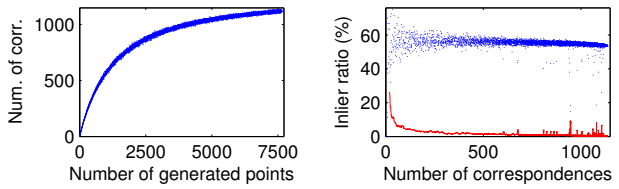


Figure 5: Dependence of the number of correspondences and inliers on number of generated points. The red line in the right image shows standard deviation in bins of size 20. In this particular case (beginning of the DUDEK sequence), we can expect stable behaviour with about 350 correspondences, thus it is enough to generate 500 new points.

by S'' into account), N_t is the number of correspondences and Λ is a regularisation term penalising large changes between P_{t-1} and P_t. The minimal sample for RANSAC is a triplet of lines, whose intersections are used to generate a transformation hypothesis. To get a more precise transformation with a higher number of inliers, the process is then repeated, using the transformation estimate as initialisation (lines 6 and 7). In this second iteration, the new locations of correspondences q_t are computed and converged to nearby edges. LO-RANSAC is then executed again using the new line-to-line correspondences.

The estimated transformation S' is usually more accurate, however it still may be noisy, which would ultimately result in tracker drift. It is therefore necessary to stabilise the estimation relative to previous frames. In LT-FLO this is done by learning the locations of edges, which have previously predicted a correct transformation. This knowledge is stored as an *edge quality field* Q_t, giving an estimate of object structure (stable edges, see Fig. 4 for an example). We expect the corresponding edges in the new frame to fit to this model of previously stable edges w.r.t. the estimated transformation. Therefore we maximise the fit (line 8 of the Alg. 1):

$$S = \arg\max_{s} \sum_{q_t} Q_{t-1}(s^{-1}(q_t)) \tag{4}$$

using Nelder-Mead iterative optimisation of the transformation parameters. In every frame, Q_t is updated as follows:

$$Q_t = \omega \cdot S(Q_{t-1}) + \bigcup_{p_t} e_{p_t} \ , \tag{5}$$

where ω is a forgetting factor (line 9). The field Q_1 is initialised taking all the edge-points from the first frame as reliable (line 1).

To allow us to refer to the short-term tracker in a compact form, we will adopt the following notation. The complete *state* of the tracker will be referred to as φ_{t-1} and includes information about the object pose P_{t-1}, edge-points p_t[2] and the *edge quality field* Q_{t-1}. The short-term tracker can then be seen as a series of recursive calls to a tracking function:

$$P_t = T_{\varphi_{t-1}}(f_t, P_{t-1}) \ . \tag{6}$$

3.1. On the Number of Generated Edge-points

To estimate a sufficient number of correspondences for a successful tracking of a given sequence, we employ an approach that accounts for the object's size and complexity. The number of final correspondences after the edge search is usually lower than the number of generated edge-points. Furthermore, this dependency is strongly non-linear and saturates (see Fig. 5). Thus we find the saturation level in the first frame by initialising with a high number of random points. Then we set the number of generated points proportional to the saturation level. This is adjusted according to the observed scale changes.

4. Long-term Module

Using line-correspondences for short-term tracking works well for short sequences. Nevertheless, for longer sequences it suffers from error accumulation (drift) and is not robust to severe occlusions.

The long-term module of LT-FLO continuously checks the *image evidence* score E_t, as this is a good indicator of the quality of the estimated transformation. When E_t decreases suddenly, this indicates a problem (the confidence in the current solution is low). On such an occasion, the short-term tracker may experience difficulties and may need *correction* (block (iii) of Fig. 3). The desired property of the (local) correction is that it can, given the last known pose of the tracked object, estimate its new pose regardless of any drift in the short-term tracker. Furthermore, it should identify a disappearance of the object (either because of a tracker failure or a full occlusion) and start a global correction, *i.e.* a *redetection*.

We propose a correction procedure, which fulfils these requirements and works both on a local level and in a redetection scenario. The long-term module has several *states* of the short-term tracker stored – a set Φ (initialised as $\Phi = \{\varphi_1\}$). When a tracking failure is detected, the short-

[2]Points p_t are computed from f_{t-1}.

term tracker is initialised using each of the stored states at the last known valid pose. A process analogous to the standard short-term tracking function $T_{\varphi_{t-1}}$ is performed, yielding several hypotheses – the *current* one and multiple *correcting* hypotheses. Each hypothesis is assigned a score:

$$\Gamma(T_{\varphi_u}) = \sqrt{\frac{|\mathcal{I}_t(S_{\varphi_u})|}{N_t}} \cdot E_t(S_{\varphi_u}) \cdot \sum_{q_t} Q_u(S_{\varphi_u}^{-1}(q_t)) \;, \quad (7)$$

where u is the frame number of φ_u. S_φ is the estimate of the transformation given by tracking from a state φ_u, and $\mathcal{I}(S_{\varphi_u})$ are the inliers: a subset of correspondences (p_t, q_t) consistent with S_{φ_u}. The hypothesis with the highest score is then selected and used for *correction*:

$$\varphi^* = \underset{\varphi \in \Phi \cup \varphi_t}{\arg\max}\, \Gamma\left(T_\varphi(f_t, P_{t-1})\right) \;, \quad (8)$$

$$S = S_{\varphi^*} \;, \qquad \varphi_{t-1} = \varphi^* \;. \quad (9)$$

If the best *correcting* hypothesis estimated is an object pose similar to the *current* one, it is a signal that the original estimated pose was correct and the current state is not replaced as it is expected to be better adapted to the current object appearance. In the extreme case, when the agreement of the estimated poses is (almost) exact, the current state is stored for use in future corrections (block (iv) of Fig. 3):

$$\Phi_{new} = \Phi \cup \varphi_{t-1} \;. \quad (10)$$

Thresholds for these decisions are calculated from the object size. At the start of the LT-FLO algorithm, the pool of correcting states is initialised with the state from the first frame.

The corrections may consume a significant portion of the execution time. As the time complexity is $O(F \cdot |\Phi|)$, it is not feasible to keep all observed states. Therefore we need a method to maximise the diversity of the learned states and cover as much variation in object appearance as possible in a fixed space and time (in our experiments, $|\Phi|_{\max} = 5$). We therefore place a limit on the cardinality of Φ and when we reach it, we replace the state used least often in recent corrections as this is a good indicator of state's usefulness in the future:

$$\Phi_{new} = \Phi \setminus \underset{\varphi \in \Phi}{\arg\min}\, \Upsilon(\varphi) \;, \quad (11)$$

where $\Upsilon(\varphi)$ is the number of occasions in the past when φ was selected according to the Eq. (8).

5. Probability of Object Pose

For many sequences, the object-and-camera system does not use the entire parameter space of object poses. Instead, only a significantly smaller subspace is occupied. An example would be the appearance of a car, followed from the rear.

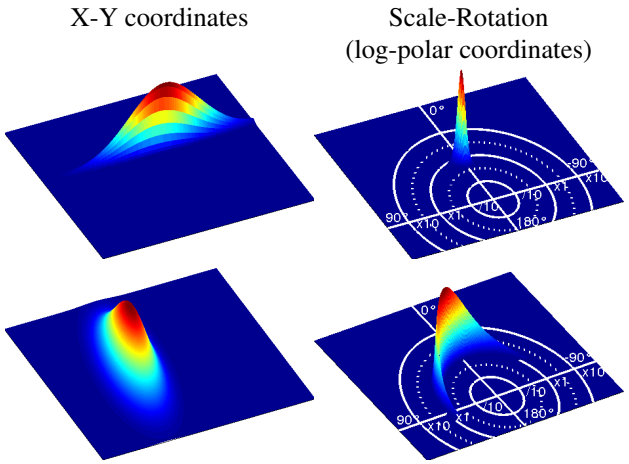


Figure 6: Examples of probability distribution Ψ (scale relative to the initial size). Top row: tracking a car (visible steps in the upper left image indicate how narrow Ψ is). Bottom row: tracking the PAGE sequence.

In such a scenario, there is almost no rotation and scale is correlated with y-coordinate (related to the actual distance between the cars). See Fig. 6.

This property of the sequences can be exploited to obtain a prior on the object pose P_t. The distribution Ψ of the object pose is modelled as a multivariate Gaussian distribution (log-normal for the scale component). This models the distribution well despite the fact that the true distribution is often not unimodal). We learn the distribution's parameters online, from observed object poses, according to [1]. In the t-th frame, the covariance is updated as:

$$\Sigma^t_{\Psi;i,j} = \frac{\Sigma^{t-1}_{\Psi;i,j}(t-1) + \left(x_i^t - \bar{x}_i^{t-1}\right)\left(x_j^t - \bar{x}_j^{t-1}\right)\frac{t-1}{t}}{t} \;, \quad (12)$$

where x_i^t denotes the i-th component of a variable x in the t-th frame and similarly $\Sigma^t_{\Psi;i,j}$ is the (i,j)-th element of its covariance matrix in the t-th frame; $\bar{x}$ is the mean (the update of which is trivial).

Given this probability distribution Ψ, we can make assumptions about the object's pose. For example, false "corrections" with extremely low probability can be rejected. This adds the following constraint to the optimisation in (8):

$$\text{s. t.} \quad \Psi(T_\varphi(f_t, P_{t-1})) > \theta_\Psi \;. \quad (13)$$

E.g. a rectangular object may look very similar when rotated, but low probability of the pose helps us identify such a situation as incorrect.

6. Failure and Redetection

The first step of the redetection procedure is to identify object disappearances and/or tracking failure. The proposed

approach to the failure discovery is based on the following observation. In the correction stage of LT-FLO (block (iii) of Fig. 3), the points from a stored state are applied to the current frame, to search for correspondences. When they are moved to an area of weak or noisy gradient, the direction of the gradient at a particular pixel is not in accordance with its neighbourhood, *i.e.* the following does not hold:

$$\cos\left[\operatorname{atan2}(g_x(p_t), g_y(p_t)) - \operatorname{atan2}(\bar{g}_x(p_t), \bar{g}_y(p_t))\right] > 0\,, \tag{14}$$

where g_x, g_y are components of the image gradient $\vec{\nabla} f$ and $\bar{g}$ indicates average over the 3×3 neighbourhood of p_t. We say the tracker failed, when there is an unusually high number of such points across all the corrections. The threshold is set here using the 99-th percentile of the fitted normal distribution for that sequence.

Object disappearance and tracking failures are also detected, based on geometric properties of the object pose. Specifically, when the tracked rectangle is too small (object considered too distant to track) or too large, when a large portion of tracked rectangle is outside the image (object out of scene) or when there are no inliers to the estimated transformation.

Once the tracker detects that the object is lost, a redetection occurs. Firstly, a local correction is performed (Section 4). This is followed by a global redetection, employing the stored states Φ. Instead of initialisation by P_{t-1}, the global redetection is initialised at several random poses P_{rand}, sampled from the observed object pose distribution Ψ:

$$\varphi^* = \underset{\varphi \in \Phi \cup \varphi_{t-1}}{\arg\max}\ \Gamma\left(T_\varphi(f_t, P_{rand})\right)\ ; \quad P_{rand} \sim \Psi\,. \tag{15}$$

After a failure, this global redetection is performed in every frame, until a good pose is found. Using the pose distribution to guide the search space is more efficient than a dense sampling of the whole parameter space. To improve the ability to generalise to possible, but previously unseen or rare poses, the search space is extended by multiplying the covariance matrix Σ_Ψ by a constant factor for each frame where redetection is performed.

7. Experimental Evaluation

7.1. VOT Challenge Results

Firstly, we report the performance of our proposed tracker on the standard VOT2013 dataset. The results are tabulated in Tab. 1 and 2. LT-FLOtrack performs equally well on both coloured and grayscale sequences, thus we report only results of experiments 2 and 3. The performance of LT-FLOtrack is very good on sequences where a rigid object is tracked (CAP, JUICE). On the other hand, sequences with highly non-rigid objects (DIVING, HAND) score lower, as well as cases of strong out-of-plane rotation (CAR).

	bicycle	bolt	car	cup	david	diving
accuracy	0.57	0.47	0.44	0.79	0.63	0.37
robustness	1.60	4.87	1.33	0.00	0.33	1.67
speed (fps)	4.10	2.99	3.43	9.09	5.70	2.72
	face	gymnastics	hand	iceskater	juice	jump
accuracy	0.72	0.50	0.45	0.43	0.78	0.52
robustness	0.07	1.20	4.67	1.80	0.07	0.20
speed (fps)	5.32	2.45	4.38	2.26	6.87	3.61
	singer	sunshade	torus	woman		
accuracy	0.60	0.63	0.55	0.49		
robustness	0.20	1.47	1.87	5.53		
speed (fps)	0.76	5.73	4.62	3.18		

Table 1: Results of the `region_noise` experiment.

	bicycle	bolt	car	cup	david	diving
accuracy	0.58	0.48	0.44	0.87	0.64	0.38
robustness	1.73	5.27	2.00	0.00	0.07	1.53
speed (fps)	4.52	3.16	2.37	8.65	6.54	3.06
	face	gymnastics	hand	iceskater	juice	jump
accuracy	0.82	0.53	0.45	0.39	0.89	0.52
robustness	0.00	1.33	4.93	2.47	0.00	0.27
speed (fps)	5.46	2.95	3.89	2.51	6.74	3.55
	singer	sunshade	torus	woman		
accuracy	0.60	0.68	0.58	0.51		
robustness	0.53	1.40	2.27	6.33		
speed (fps)	0.67	5.33	4.19	3.10		

Table 2: Results of the `grayscale` experiment.

7.2. Short-term Tracking

The performance of the LT-FLOtrack algorithm was next evaluated in a short-term tracking scenario, against a number of competitive SOTA approaches – FLO [10], TLD [9], LGT [2] and FoT [13] using the datasets from [10]. Properties of the sequences are summarised in Table 3 and the first frames are shown in Fig. 7 and 9.

The speeds are shown in Table 4. It should be noted that while the trackers are generally implemented as a compiled (C/C++) core with Matlab front-end, the FoT tracker is written completely in C++.

Figure 8 shows the results. Performance is measured using *position error* (distance of the tracked area centre from its ground truth position) and *scale error* (logarithm of ratio

Name	Resolution	Frames
DUDEK	720×480	1 145
DOG	320×240	1 353
MUG	640×480	737
PAGE	640×480	539

Name	Resolution	Frames
CARCHASE	290×217	9 928
PANDA	312×233	3 000
VOLKSW.	640×480	8 576
LIVERRUN^N	320×240	29 700
NISSAN^N	640×480	3 800

Table 3: Experimental video sequences (left: short-term tracking, right: long-term tracking). Sequences marked by ^{N} are new and the authors are not aware of their usage in any previous publication. These are available online with ground truth.

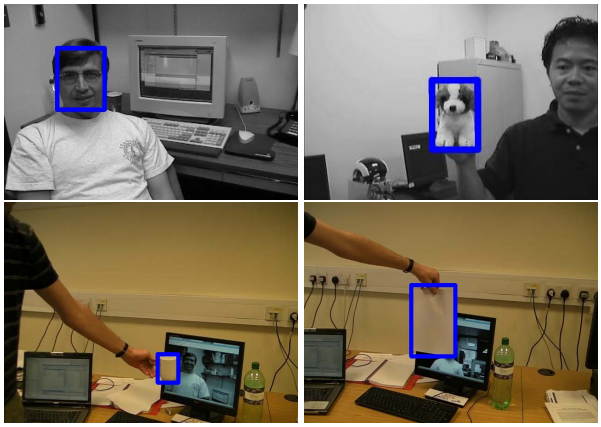


Figure 7: Short-term tracking dataset of [10]. From top and left: DUDEK [8], DOG [3], MUG and PAGE.

Sequence	DUDEK	DOG	MUG	PAGE
LT-FLO	3.3	6.3	4.8	3.1
FLO	0.2	0.3	0.4	0.3
TLD	2.0	4.3	2.6	2.7
LGT	2.1	3.4	2.8	2.5
FoT	131.7	405.5	230.6	212.4

Table 4: Speed comparison of different trackers (in FPS).

of the estimated object size to its ground truth size). The values were averaged over 20 executions.

It is apparent that the LT-FLO is at least as good as FLOtrack (DOG, PAGE, while significantly outperforming the other trackers), or better (DUDEK, MUG). In particular, higher robustness to out-of-plane rotation improves performance beyond frame 800 of the DUDEK sequence, where FLO experiences drift. LT-FLO as well as FLO and TLD recover successfully from the occlusion in frame 200, while the other trackers do not (however TLD takes longer to do so). In the MUG scene, the trackers sometimes lock onto the background, however the number of such situations is reduced in LT-FLO, resulting in significant reduction of the mean position error. Compared to the other trackers, on the MUG sequence, FoT works well (comparable to LT-FLO) while TLD sometimes gets lost and often underestimates the size of the object. LGT cannot track the sequence, due to a lack of texture information.

7.3. Long-term Tracking

For the evaluation of long-term tracking, different sequences are necessary. Not only should they be longer, but more importantly they must include full occlusions, background clutter and scale and illumination changes [9]. A common source of such data are traffic scenes such as car chases, when a vehicle is followed by another (possibly aerial) vehicle. The majority of sequences in this evaluation are therefore of this type. These sequences explore redetection capability and predisposition to drift. Their properties are tabulated in Table 3 and the first frames are shown in Figure 9.

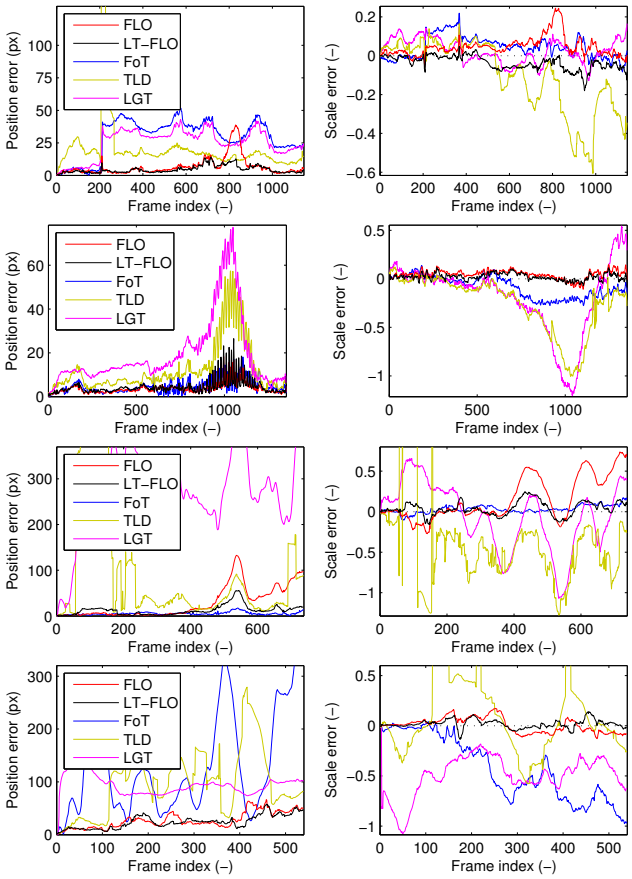


Figure 8: Results of short-term tracking evaluation. From top to bottom: DUDEK, DOG, MUG and PAGE.

In a long-term tracking scenario, it is necessary to check whether the detection/tracking is precise (when the overlap with the ground truth bounding box is higher than 50 %) as well as to check for successful detection of object disappearance. Kalal *et al.* [9] applied the precision/recall/F-measure comparison. To check sensitivity of trackers to the initialisation, we run the experiments multiple times with the bounding box in the first frame shifted by 5 % in both horizontal and vertical direction and also scaled up and down by 5 % (averaging the results over all 7 possibilities). We also introduce the concept of partial occlusion into the evaluation process. When the object is not fully visible, but not fully occluded, the tracker can receive a score for either overlap of bounding boxes or for reporting disappearance.

We compare our tracker with TLD [9], the first explicitly long-term tracker. The results can be found in Table 5. The trackers were initialised by a tight bounding box. It should be noted that the experiments of TLD on their own dataset have significantly different results if their own ini-

Figure 9: Long-term tracking dataset. From top and left: CARCHASE [9], PANDA [9], VOLKSWAGEN [9], LIVERRUN^N and NISSAN^N.

tialisation is used (better on the CARCHASE and PANDA sequences and worse on the VOLKSWAGEN sequence, relating to F-measures of 0.45, 0.49 and 0.57 respectively). This indicates high sensitivity to the initialisation. Experiments were carried out with the short-term trackers as well, but results of these are not tabulated as they lost tracking at or before the first full occlusion (*e.g.* frame 440 for LIVERRUN or 1860 for NISSAN).

Name	LT-FLO	TLD
CARCHASE	**.40** (3.2)	.16 (11.3)
PANDA	.19 (4.1)	**.23** (12.0)
VOLKSW.	.48 (3.0)	**.63** (5.4)
LIVERRUN	**.52** (6.1)	.34 (4.0)
NISSAN	**.74** (4.1)	.65 (4.4)
mean	**.49** (4.9)	.36 (6.0)

Table 5: Results of long-term tracking. Tabulated values: *F-measure (FPS)*. The mean is weighted by number of frames.

The results show that LT-FLO is capable of long-term tracking with competitive performance. In three out of five of the tested sequences it performed significantly better than TLD and comparably on one sequence. Also the average performance is better than TLD. The worst results of LT-FLO were acquired on the PANDA sequence. This highlights sensitivity to highly non-rigid objects with out of plane rotation.

8. Conclusion

In this paper, a novel long-term, edge-based tracking algorithm is presented. A two module approach is used, where the short-term tracker finds the frame-to-frame transformations, using correspondences of lines tangent to edges. The long-term module facilitates learning of the changing appearance of the tracked object, recovery from failures and redetection after full occlusions. Strict learning rules make it very robust to drift. This, in conjunction with a guided redetection framework, renders LT-FLOtrack capable of tracking extremely long sequences robustly.

LT-FLOtrack covers several challenging cases, where traditional trackers often fail. However, it does not attain the highest score on every sequence. Typically, performance is degraded for highly non-rigid or articulated objects (*e.g.* in the PANDA sequence), and by strong out of plane rotations, or edge-distorting compression artefacts.

Acknowledgements This work was supported by the EPSRC grant "Learning to Recognise Dynamic Visual Content from Broadcast Footage" (EP/I011811/1) and The Czech Science Foundation Project GACR P103/12/G084.

References

[1] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Cluster*, 2009.

[2] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *PAMI*, 2013.

[3] M. Chen, S. K. Pang, T. J. Cham, and A. Goh. Visual tracking with generative template model based on riemannian manifold of covariances. In *ICIF*, 2011.

[4] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *PAMI*, 2002.

[5] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[6] C. Harris and C. Stennett. Rapid – a video rate object tracker. In *BMVC*, 1990.

[7] Z. Hong, X. Mei, and D. Tao. Dual-force metric learning for robust distracter-resistant tracker. In *ECCV*, 2012.

[8] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *PAMI*, 2003.

[9] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *PAMI*, 2012.

[10] K. Lebeda, J. Matas, and R. Bowden. Tracking the untrackable: How to track when your object is featureless. In *ACCV DTCE*, 2012.

[11] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized RANSAC. In *BMVC*, 2012.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Imaging Underst. Workshop*, 1981.

[13] J. Matas and T. Vojir. Robustifying the flock of trackers. In *CVWW*, 2011.

[14] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *Image Processing*, 1997.

[15] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 2008.

[16] P. Smith, I. Reid, and A. Davison. Real-Time Monocular SLAM with Straight Lines. In *BMVC*, 2006.

[17] Y. Tsin, Y. Genc, Y. Zhu, and V. Ramesh. Learn to track edges. In *ICCV*, 2007.