International Conference on Pattern Recognition, 23-26 August, 2010, Istambul, Turkey

Forward-Backward Error: Automatic Detection of Tracking Failures

Zdenek Kalal
CVSSP, UK
z.kalal@surrey.ac.uk

Krystian Mikolajczyk
CVSSP, UK
k.mikolajczyk@surrey.ac.uk

Jiri Matas
CMP, Czech Republic
matas@cmp.felk.cvut.cz

Abstract

This paper proposes a novel method for tracking failure detection. The detection is based on the Forward-Backward error, i.e. the tracking is performed forward and backward in time and the discrepancies between these two trajectories are measured. We demonstrate that the proposed error enables reliable detection of tracking failures and selection of reliable trajectories in video sequences. We demonstrate that the approach is complementary to commonly used normalized cross-correlation (NCC). Based on the error, we propose a novel object tracker called Median Flow. State-of-the-art performance is achieved on challenging benchmark video sequences which include non-rigid objects.

1. Introduction

Point tracking is a common computer vision task: given a point location in time t, the goal is to estimate its location in time $t+1$. In practice, tracking often faces with a problem where the points dramatically change appearance or disappear from the camera view. Under such conditions, tracking often results in failures. We study the problem of failure detection and propose a novel method that enables any tracker to self-evaluate its reliability.

The proposed method is based on so called forward-backward consistency assumption that correct tracking should be independent of the direction of time-flow. Algorithmically, the assumption is exploited as follows. First, a tracker produces a trajectory by tracking the point *forward* in time. Second, the point location in the last frame initializes a validation trajectory. The validation trajectory is obtained by *backward* tracking from the last frame to the first one. Third, the two trajectories are compared and if they differ significantly, the forward trajectory is considered as incorrect. Fig. 1 *(top)* illustrates the method when tracking a point between two images (basic trajectory). Point no. 1 is visible in both images and the tracker is able to localize it correctly. Tracking this point forward or backward results in identical trajectories. On the other hand, point no. 2 is not visible in the right image and the tracker localizes a different point. Tracking this point backward ends in a different location then the original one. The inconsistency can be easily identified and as we show in the experimental section, it is highly correlated with real tracking failures.

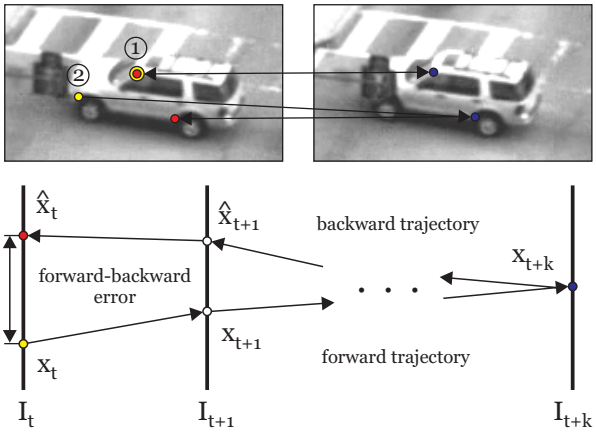

Figure 1. The Forward-Backward error penalizes inconsistent trajectories. Point 1 is visible in both images, tracker works consistently forward and backward. Point 2 is occluded in the second image, forward and backward trajectories are inconsistent.

A commonly used approach to detect tracking failures is to describe the tracked point by a surrounding patch $\mathcal{R}$ which is compared from time t to $t+1$ using sum-of-square differences (SSD) [3, 9]. This differential error enables detection of failures caused by occlusion or rapid movements, but does not detect slowly drifting trajectories. The detection of a drift can be approached by defining an absolute error, such as a comparison between the current patch and affine warps of the initial appearance [11]. This method is applicable only to planar targets. Recently, a general method for assessing the tracking performance was proposed [13], which is based on a similar idea to the one explored in

this paper. The method was designed for particle filters with a static measurement model. Adaptation of their method to point tracking was not suggested.

The rest of the paper is organized as follows: Sec. 2 formalizes a novel error measure, called Forward-Backward error and compares it quantitatively with SSD in Sec. 3. In Sec. 4 the approach will be applied to the feature point selection. Sec. 5 proposes Median Flow tracker which outperforms state-of-the-art approaches on benchmark sequences. The paper finishes with conclusions and future work.

2. Forward-Backward Error

This section defines the Forward-Backward (FB) error of feature point trajectories, see Fig. 1 *(bottom)* for illustration. Let $S = (I_t, I_{t+1}..., I_{t+k})$ be an image sequence and $\mathbf{x}_t$ be a point location in time t. Using an arbitrary tracker, the point $\mathbf{x}_t$ is tracked forward for k steps. The resulting trajectory is $T_f^k = (\mathbf{x}_t, \mathbf{x}_{t+1}, ..., \mathbf{x}_{t+k})$, where f stands for *forward* and k indicates the length. Our goal is to estimate the error (reliability) of trajectory T_f^k given the image sequence S. For this purpose, the validation trajectory is first constructed. Point $\mathbf{x}_{t+k}$ is tracked *backward* up to the first frame and produces $T_b^k = (\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_{t+1}, ..., \hat{\mathbf{x}}_{t+k})$, where $\hat{\mathbf{x}}_{t+k} = \mathbf{x}_{t+k}$. The Forward-Backward error is defined as the distance between these two trajectories: $\mathrm{FB}(T_f^k|S) = \mathrm{distance}(T_f^k, T_b^k)$. Various distances can be defined for the trajectory comparison. For the sake of simplicity, we use the Euclidean distance between the initial point and the end point of the validation trajectory, $\mathrm{distance}(T_f^k, T_b^k) = ||\mathbf{x}_t - \hat{\mathbf{x}}_t||$.

The main advantage of the proposed FB error is that it can be applied to a range of trackers and is easy to implement. Trackers typically fail if the data violate the assumptions. For instance when the motion of the target is faster than expected. In that case, the tracker produces a trajectory which is to some extent random. Tracking backward produces yet another random trajectory. These trajectories are likely to be different. Recently, a novel tracking algorithm [12] was proposed that exploits the time-reversibility directly in its predictions. Therefore, the forward-backward trajectories are consistent by definition. In this case the FB error can not be applied.

3. Detection of Tracking Failures

The ability of the FB error to identify correct tracking was quantitatively evaluated on synthetic data. One hundred images of natural scenes was warped by random affine transformations and added with Gaussian noise. A set of points was initialized on a regular grid

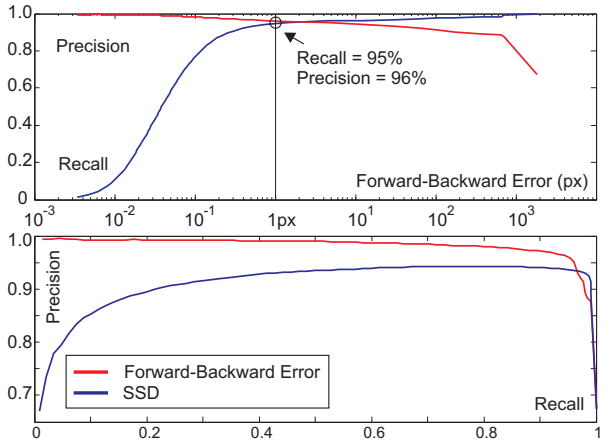

Figure 2. *(top)* **Thresholding the Forward-Backward error enables to identify correctly tracked points.** *(bottom)* **Precision/recall characteristics of a classifier based on FB and SSD. FB is significantly better than SSD for the majority of working points.**

(with 5 pixel interval) in the original images and projected to the distorted images to create the ground truth trajectories. Displacements of the points between the original and warped images were estimated by Lucas-Kanade tracker [8, 11]. Displacements that ended up closer then 2 pixels from the ground truth were labeled as inliers (65%). The trajectories were then evaluated by an error (FB, SSD) and classified as inliers if the error was smaller than a threshold. The results were compared to the ground truth and the precision/recall statistics were computed.

Fig. 2 *(top)* shows the resulting performance of FB as a function of the threshold. Notice the threshold of 1 pixel (1px) where the recall is of 95% and precision of 96%. This high performance demonstrates that the FB error detects correct trajectories by thresholding. The bottom figure shows the corresponding precision/recall curves for FB in comparison to standard SSD. FB is significantly better than SSD for majority of working points. SSD was unable to detect inliers for small thresholds, its precision starts below 70% (random guessing would start at 65%). We have carried out another experiments where the regular grid of features was replaced by "FAST" feature point detector [10] and consistent observations were made.

4. Trajectory Selection from Video

Tracking performance is sensitive to the initialization and therefore the selection of points to track is important. The selection is typically done in the first frame by detecting keypoints [11, 10] that are easy to track. The problem is that even these points can become oc-

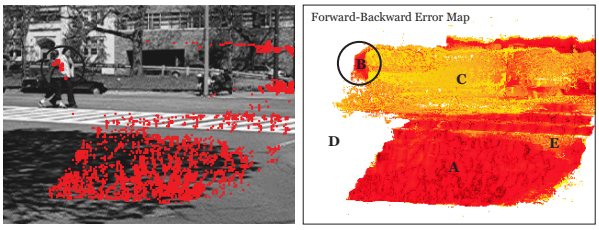

Figure 3. Point selection from video. *(left)* **The first frame from the sequence** PEDESTRIAN 1**. The red dots indicate 1% of points with most reliable trajectories.** *(right)* **Error Map of the same sequence. The pixel intensity is inversely proportional to the trajectory reliability.**

cluded, disappear or change their appearance later on in the sequence leading the tracker to a failure. Therefore, if possible, the selection mechanism should incorporate the information from the whole video.

Here we propose a brute force selection mechanism based on the FB error: 1) track every pixel from the first frame through the whole sequence, 2) evaluate the trajectories by the FB error, 3) assign each pixel the error of its trajectory. The resulting image is called *Error Map* and shows which points are tracked reliably throughout the whole sequence. The selection of the points is done by thresholding.

The Error Map has been constructed for the sequence PEDESTRIAN 1, used in [14], and the result is shown in Fig. 3. The left panel shows the first frame of the sequence. The red points indicate 1% of the most reliable pixels. The right panel shows the Error Map. Dark colors indicate low FB error: tree shadow (A), upper bodies of two pedestrians (B). Any point selected from these areas is tracked accurately in the whole sequence. Brighter colors areas which are difficult or impossible to track. These areas may become occluded (C), disappear from the camera view (D) or lack enough structural information (E). This experiment demonstrates that the FB error can be used to select feature points for which the tracking is reliable in the whole video sequence. This is in contrast to standard feature selection methods [11, 10] which are based on the information from a single image.

5. Object Tracking by Median Flow

Previous sections were discussing an approach for tracking of points which were considered as independent. However, in natural videos, the points are rarely independent but are parts of bigger units which move together. These units will be called objects (cars, pedestrians, human face, etc.). This section exploits a point tracking algorithm and the proposed FB error measure,

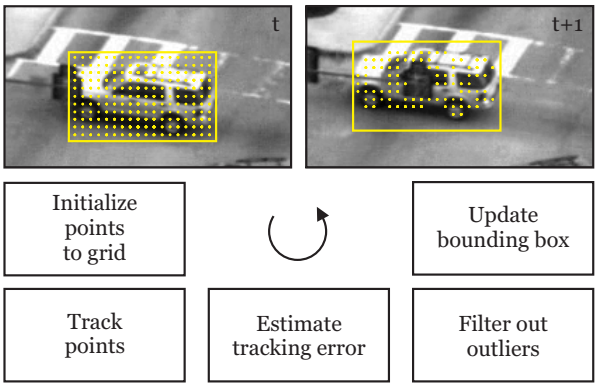

Figure 4. The Median Flow tracker accepts a bounding box and a pair of images. A number of points within the bounding box are tracked, their error is estimated and the outliers are filtered out. The remaining estimate the bounding box motion.

and designs a novel robust object tracker with superior performance.

A block diagram of the proposed tracker is shown in Fig. 4. The tracker accepts a pair of images I_t, I_{t+1} and a bounding box β_t and outputs the bounding box β_{t+1}. A set of points is initialized on a rectangular grid within the bounding box β_t. These points are then tracked by Lucas-Kanade tracker which generates a sparse motion flow between I_t and I_{t+1}. The quality of the point predictions is then estimated and each point is assigned an error (e.g. FB, NCC, SSD). 50% of the worst predictions are filtered out. The remaining predictions are used to estimate the displacement of the whole bounding box. We refer to this tracker as Median Flow.

Estimation of the bounding box displacement from the remaining points is performed using median over each spatial dimension. Scale change is computed as follows: for each pair of points, a ratio between current point distance and previous point distance is computed; bounding box scale change is defined as the median over these ratios. An implicit assumption of the point-based representation is that the object is composed of small rigid patches. Parts of the objects that do not satisfy this assumption (object boundary, flexible parts) are not considered in the voting since they are rejected by the error measure.

A number of variants of the Median Flow tracker was tested. The baseline tracker T_0, also used in [5], does not evaluate the point tracking error and estimates the bounding box displacement based on all points. Trackers T_{FB}, T_{NCC}, T_{SSD} estimate the error by FB, NCC and SSD. 50% of the outliers are filtered out. Tracker $T_{\mathrm{FB+NCC}}$ combines FB and NCC, each error independently filters out 50% of outliers. These trackers were compared with the state-of-the-art ap-

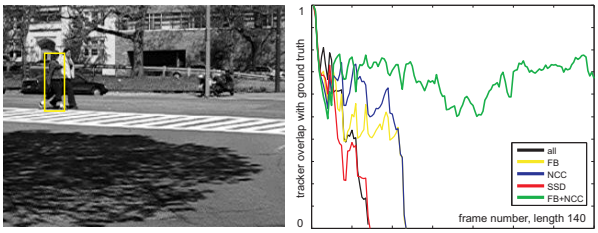

Figure 5. Sequence PEDESTRIAN 1**: Performance comparison of various error measures measures.**

Sequence	Frames	[7]	[4]	[1]	[2]	T_0	T_{FB}	T_{NCC}	T_{SSD}	$T_{\mathrm{FB+NCC}}$
David	761	17	n/a	94	135	93	761	144	28	**761**
Jumping	313	75	313	44	313	36	76	87	79	170
Pedestrian 1	140	11	6	22	101	15	37	40	12	**140**
Pedestrian 2	338	33	8	118	37	97	97	97	97	97
Pedestrian 3	184	50	5	53	49	52	52	52	52	52
Car	945	163	n/a	10	45	248	510	394	353	**510**
Best	n/a	0	1	2	1	0	2	0	0	3

Table 1. Comparison with state-of-the-art approaches in terms of the number of correctly tracked frames.

proaches [7, 4, 1, 2] on 6 video sequences from [14]. The objects were manually initialized in the first frame and tracked up to the end of the sequence. The trajectory was considered correct if the bounding box overlap with ground truth was larger than 50%. Performance was assessed as the maximal frame number up to which the tracker was correct.

Fig. 5 demonstrate the influence of the error measure on performance of the Median Flow tracker. The baseline T_0 method as well as T_{SSD} perform the worst. This supports our claims made earlier that SSD is less reliable in identifying correct point trajectories. T_{FB} and T_{NCC} perform similarly and are able to follow the target for twice as long than the baseline method. $T_{\mathrm{FB+NCC}}$ clearly dominates and is able to track the target throughout entire sequence. This shows that FB is independent to NCC and their combination leads to significant improvement in tracking. Table 1 shows the quantitative results for all sequences. The last row shows the number of times the particular algorithm performed best. The best results obtained the Median Flow based on a combination of FB and NCC error. This tracker was able to score best three times and defines new state-of-the-art performance on these sequences.

6. Conclusions

This paper proposed a novel measure, Forward-Backward error, that estimates reliability of a trajectory. A validation trajectory is constructed by backward tracking and compared to the trajectory in question. The implementation only involves applying the same tracking algorithm on a reversed sequence of images. The measure was applied to Lucas-Kanade tracker and its benefits and complementarity to SSD and NCC were shown. Based on the proposed Forward-Backward error we designed a novel tracker, called Median Flow, that achieved state-of-the-art performance on several benchmark sequences. The FB error can also be implemented as a constraint and used in a novel tracking framework introduced in [6].

Acknowledgment. This research was supported by UK EPSRC EP/F0034 20/1 and the BBC R&D grants (ZK, KM) and by EC project ICT-215078 DIPLECS (JM).

References

[1] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271, 2007.
[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. *CVPR*, 2009.
[3] J. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *Technical report, OpenCV Document, Intel Microprocessor Research Labs*, 1999.
[4] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.
[5] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. *OLCV*, 2009.
[6] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *CVPR*, 2010.
[7] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. *NIPS*, 2005.
[8] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, 81:674–679, 1981.
[9] K. Nickels and S. Hutchinson. Estimating uncertainty in SSD-based feature tracking. *IVC*, 20(1):47–58, 2002.
[10] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *ECCV*, May 2006.
[11] J. Shi and C. Tomasi. Good features to track. *CVPR*, 1994.
[12] H. Wu, R. Chellappa, A. Sankaranarayanan, and S. Zhou. Robust visual tracking using the time-reversibility constraint. *ICCV*, 2007.
[13] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. In situ evaluation of tracking algorithms using time reversed chains. *CVPR*, 2007.
[14] Q. Yu, T. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. *ECCV*, 2008.