This paper was published on 3rd On-line Learning for Computer Vision Workshop 2009, Kyoto, Japan, IEEE CS.

# Online learning of robust object detectors during unstable tracking

Zdenek Kalal
University of Surrey
Guildford, UK
z.kalal@surrey.ac.uk

Jiri Matas
Czech Technical University
Prague, Czech Republic
matas@cmp.felk.cvut.cz

Krystian Mikolajczyk
University of Surrey
Guildford, UK
k.mikolajczyk@surrey.ac.uk

## Abstract

*This work investigates the problem of robust, long-term visual tracking of unknown objects in unconstrained environments. It therefore must cope with frame-cuts, fast camera movements and partial/total object occlusions/dissapearances. We propose a new approach, called Tracking-Modeling-Detection (TMD) that closely integrates adaptive tracking with online learning of the object-specific detector. Starting from a single click in the first frame, TMD tracks the selected object by an adaptive tracker. The trajectory is observed by two processes (growing and pruning event) that robustly model the appearance and build an object detector on the fly. Both events make errors, the stability of the system is achieved by their cancelation. The learnt detector enables re-initialization of the tracker whenever previously observed appearance re-occurs. We show the real-time learning and classification is achievable with random forests. The performance and the long-term stability of TMD is demonstrated and evaluated on a set of challenging video sequences with various objects such as cars, people and animals.*

## 1. Introduction

In this paper, we propose a method addressing a problem of *long-term online tracking with minimum prior information*. "Long-term" refers to sequences of possibly infinite length that contain frame-cuts, fast camera movements and the object may temporarily disappears from the scene. "Online" means that the tracking does not exploit information from the future and processes the footage in one pass. "Minimum prior information" indicates that the object is not known in advance and the only information about it comes from the first frame where it was selected by the user.

Standard tracking approaches [11] that perform frame-to-frame tracking assume no complete occlusion or disappearance. The research in such methods focuses on speed, precision or on the development of more reliable methods that extend the "lifetime" of the tracker [5], but do not address directly the post-failure behavior and therefore can not be directly used in the long-term tracking problem. We refer to this group of algorithms as *short-term trackers*.

Clearly the solution of the long-term tracking problem requires some detection capability, to re-detect the object after a period when it is not in the field of view or after tracking failure. Tracking-by-detection methods [9] or methods integrating a tracker and a detector [1, 17] address the problem. However, detectors have to be designed or trained before tracking starts and thus cannot be used when the object of interest is not known in advance. The training of these detectors either requires a large hand-labeled training sets [20], generates the training set by warping the patches [9] or extracts the training data using some sophisticated method [16, 18, 19]. All these methods strictly separate the training and testing phase which means that appearance variability not represented in the training set never becomes part of the model.

The appearance change problem is addressed by adaptive tracking methods. These methods can be roughly split into two categories based the model updating strategy. *Every-frame-update* is most common for adaptive trackers [2, 4, 7, 6]. The tracker is expected to perform correctly and under this assumption, each observation updates the object model. Such approaches enable to quickly adapt to appearance changes but may also lead to acceleration of the tracking failure. *Selective-update* strategies take into account the fact that the tracker is not always correct. Therefore the update may be done only if the tracker is not far from the model [13] or the new, unlabeled data can be integrated to the model in a semi-supervised framework [8].

In this paper we propose a new approach to the long-term tracking problem that exploits three components: tracking, modeling and detection (TMD). The object is tracked by an adaptive short-term tracker based on Lucas-Kanade method [11]. During the tracking, the appearance is modeled in a novel unsupervised manner based on two events. The model is iteratively extended by so called *growing events* and refined by *pruning events*. These two events are designed to correct errors of each other, which makes

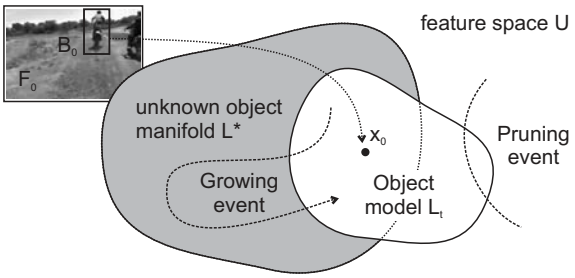


Figure 1. The online model is initialized in the first frame by the selected sample $x_0$. It is expanding by "growing events", refined by "pruning events" and slowly converges toward the unknown object manifold.

the modeling robust to inevitable failures of the short-term tracker. The result of the modeling is a collection of templates that represent the selectively pruned memory of the system. Based on this collection, the object detector is built online. The detector runs in parallel with the short-term tracker and enables re-initialization after its failure. It has a form a randomized forest, enables incremental update of its decision boundary and real-time sequential evaluation during run-time.

The paper has the following contributions: (i) formulation of a new approach (TMD) that addresses the long-term tracking problem, (ii) introduction of a novel learning method based on two error-canceling events that bootstraps the object model from a single click, (iii) design of an efficient detector structure enabling real-time learning/classification, (iv) new efficient local features, (v) introduction of new sequences for the long-term tracking problem. In the experiments we observe that TMD performs robust long-term tracking that automatically progress from adaptive tracking to tracking-by-detection without the need for off-line training. Also we empirically show that the system performance monotonically improves over time.

The rest of the paper is organized as follows: Sec. 2 introduces the TMD framework and the events; Sec. 3 discusses our implementation of TMD; Sec. 4 first compares several growing events and than evaluates TMD on standard and new sequences; the paper's observations are wrapped up in Conclusions.

## 2. TMD Framework

In this section we present our tracking system and its components (see Fig. 1 for illustration). Let $F_t$ and $B_t$ be an image frame and an object bounding box in time $t$. The pixel area (patch) within $B_t$ is represented by a feature vector $x_t$ that encodes the object appearance. The set of consecutive bounding boxes defines the track $T_t = \{B_0, B_1, \ldots, B_t\}$ of length $t+1$ that represents a trajectory of the target in image space. $T_t^f$ denotes the corresponding trajectory in feature space $U$ within which exist a subspace $L^*$. $L^*$ represents all possible object appearances (manifold) and is unknown apart from one single patch $x_0 \in L^*$ that was selected for tracking. This sample represents our online model $L_0 = \{x_0\}$ in time $t = 0$.

The components of TMD interact as follows. The selected object is tracked by a short-term tracker. The trajectory in feature space is analyzed by two processes that continuously attempt to extend or to restrict the space covered by the online model. $L_t$ is extended with samples that are likely to contain the object of interest. These samples are identified by *growing events*. $L_t$ is pruned from samples that are considered as wrong. These samples are identified by *pruning events*. The two events work in parallel with the aim to converge $L_t \rightarrow L^*$ (in Fig. 1 this corresponds to fitting the white blob to the gray blob).

The main purpose of creating $L_t$ is to represent the "memory" of the system and to build an object detector that is continuously updated and evaluated. It scans each input frame $F_t$ and outputs a set of bounding boxes with appearances contained in $L_t$. These bounding boxes represent an alternative hypothesis to the position returned by the tracker. The hypothesis fusion is performed by taking the position that minimizes the distance to $L_t$. It follows that if the patch given by the tracker is very close to $L_t$, false responses of the detector do not affect the track unless their distance to the online model is even closer. The minimal distance to $L_t$ becomes the negated confidence score of the TMD system. Based on the confidence score, the tracker decides if the object is visible or not.

**Algorithm 1** TMD framework

**Require:** Select $x_0$, $L_0 = \{x_0\}$
  **for** $t = 1 : \infty$ **do**
    Track last patch $x_{t-1}$.
    Detect patches contained in online model $L_{t-1}$.
    $L_t \leftarrow L_{t-1} \cup$ Positive samples from growing.
    $L_t \leftarrow L_{t-1} \setminus$ Negative samples from pruning.
    $x_t \leftarrow$ Most confident patch (detected or tracked).
  **end for**

So far we introduced the events that observe the unstable tracker with the purpose to learn the object appearance. In the following, we specify in more detail how they work. For this purpose, we distinguish two parts of $L_t$, the correct part $L_t^c \in L^*$ and the error part $L_t^e \notin L^*, L_t^c \cup L_t^e = L_t, L_t^c \cap L_t^e = \emptyset$. *Coverage* represents the proportion of the object appearance that was already discovered by the unsupervised learning process, i.e. $\text{coverage}(L_t) = |L_t^c|/|L^*|$. *Impurity* represents the fraction of $L_t$ that is incorrect, i.e. $\text{impurity}(L_t) = |L_t^e|/|L_t|$. Operator $|.|$ denotes size of the set.

**Growing events** At time $t$ the short-term tracker has produced a trajectory $T_t^f = \{x_0, x_1, \ldots, x_t\}$. A growing event first selects certain parts of this trajectory that are considered to be positive, $P \in T_t^f$. The online model $L_t$ is then updated, i.e. $L_t = L_{t-1} + P$. After the update, the coverage of the online model increases if $P$ contains at least one sample from $L^*$. In this paper we propose a new growing event that takes advantage of the drift of an adaptive tracker. This strategy will be discussed in section 3.2.

**Pruning events** It is impossible to devise an example selection strategy that would allow only correct samples. Therefore the impurity of $L_t$ is increasing. Since our goal is to converge $L_t \rightarrow L^*$, the events that lead to reduced impurity of online model $L_t$ are essential. If online model $L_t$ is characterized by a certain level of impurity, a pruning event is to identify a subset $N$ of the online model that is considered incorrect and removes it, i.e. $L_t = L_{t-1} - N$.

Growing event alone leads to high impurity and therefore detector with low precision. Pruning event serves as a "negative feedback": the higher the impurity of the model, the more negative samples are identified and removed from the model. The dynamic interaction of growing and pruning is crucial in making the TMD system "stable" as we empirically show in the experimental section.

## 3. Implementation

**Short-term tracker.** Our short-term tracker is based on the Lucas-Kanade (LK) method [11]. First, a set of feature points is sampled from a rectangular grid within the object bounding box. Next, LK tracks the points from one frame to another, resulting in a sparse motion field. Based on the motion field, the bounding box displacement and scale change are robustly estimated as a median over the parameter distribution. In each frame a completely new set of feature-points is tracked which makes the tracker is very adaptive.

**Online model.** The online model is represented by a set of 15x15 intensity normalized patches. Distance between two patches is defined using normalized cross-correlation, i.e. $\text{distance}(x_i, x_j) = 1 - \text{NCC}(x_i, x_j)$. The distance of sample $x_i$ to the online model $L_t$ is defined as $\text{distance}(x_i, L_t) = \min_{x \in L_t} \big(\text{distance}(x_i, x)\big)$. The model is based on patches to provide complementarity to efficient but less discriminative detector features discussed in the following section.

### 3.1. Object detector

It is crucial to build fast and reliable detector able to localize patches contained in the online model and to efficiently adjust its decision boundary by the growing and pruning events.

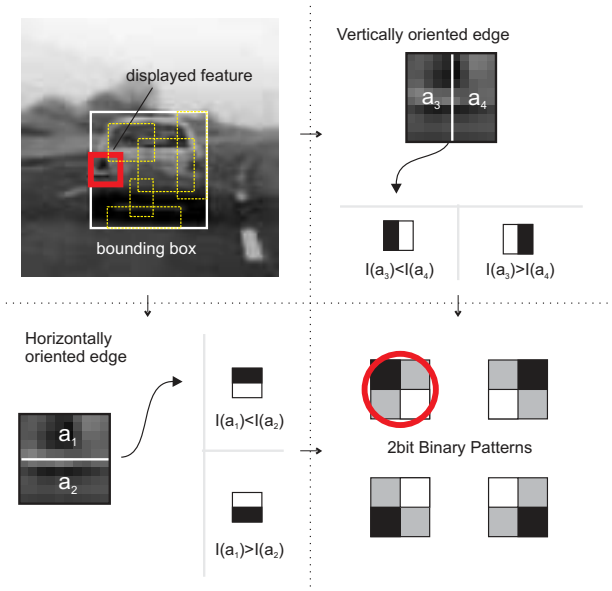


Figure 2. Local 2bit Binary Patterns used in our object detector. Features encode local gradient orientation withing the object bounding box.

Realtime state-of-the art object detectors are typically based on the AdaBoost [20] algorithm. They require large training sets and are computationally expensive to train. This approach is not applicable for the online setting. Existing online detectors [7] enable efficient training but their purpose is to "adapt" to new samples and gradually dismiss the old ones. In our case, the detector should rather "absorb" new samples and keep them in the model as long as they are not removed by the pruning event.

**Features.** Our object detector is based on new features that we call 2bit Binary Patterns (2bitBP). These features measure gradient orientation within a certain area, quantize it and output four possible codes (see Fig. 2 for illustration).

2bitBP were inspired by Local Binary Patterns [14] but differ since standard LBP encodes $3 \times 3$-pixel surrounding and represents a certain area by a distribution of the codes, but 2bitBP encodes the area by a single code. In this sense the 2bitBP is similar to haar-like features [20] and the multi-scale measurement is achieved using integral images. Moreover, 2bitBP outputs just 4 codes in contrast to 256 for standard LBP, which increases resistance to overfitting.

**Sequential randomized forest.** Each image patch is described by a number of local 2bitBP which position, scale and aspect ratio were generated at random. These features are randomly partitioned into several groups of the same size. Each group represents a different view of the patch appearance. The response of each group is represented by a discrete vector that will be called a "branch".

The classifier used for detection has the form of a randomized forest [3]. It enables online update and sequential evaluation. The forest consists of several trees, each of them is build from one group of features. Every feature in the group represents a measurement taken at a certain level of the tree.

Growing and pruning of the forest is incremental – one example is processed at a time. At the beginning, every tree contains one single branch defined by the selected patch. With every positive example selected by the growing event, a new set of branches is added to the forest. Tree pruning corresponds to removal of branches selected by the pruning event.

Evaluation of an unknown patch by the tree is very efficient. The features within each tree are measured sequentially. If the patch reaches the end of the tree it is considered as positive, otherwise if it differs from the defined branches the measurement is terminated and the patch is considered as negative. The sequential nature of the randomized tree enables real-time evaluation on a large number of positions and is different from [9, 15] where a fixed number of measurements have to be taken. The input image is scanned with a sliding window. At each position, every tree outputs a decision whether the underlying patch is in the model or not. The final decision is obtained by the majority vote.

One of the main differences from other implementations of randomized trees is that training is performed using positive samples only. Negative samples influence the training indirectly. If the pruning event finds a false positive detection its corresponding branches are removed. In order to increase the detection precision we further filter out detections that are not similar to the online model (measured by cross-correlation). This is performed only for the most promising patches since the majority of them have been already rejected by the randomized forest.

The structure of the randomized forest was estimated empirically on a set of sequences in order to achieve high recall with constraint on real-time performance. We use 8 trees, each consists of 10 features, which corresponds to $4^{10}$ possible leaves.

### 3.2. Events

The proposed TMD framework does not specify the growing/pruning events explicitly as they can be designed for a problem specific task. In this work we use events that are generally applicable to a wide range of adaptive trackers and detectors based on a scanning window. These events are based on a similarity threshold $\theta$: two patches on the tracker's trajectory are considered *similar* if their distance is less than $\theta$.

**Growing events.** Growing events consists of: selection of appropriate samples from the tracker's trajectory and model update. Three selection strategies were implemented and tested within our framework; the model update is performed identically in all of them (as discussed in section 'Online model' and 'Sequential randomized forest')

1. *Absolute Distance from First Patch* (ABS) Approves all patches $x_t$ that are similar to the first patch $x_0$.

2. *Difference Between Consecutive Patches* (DIFF). Approves patch $x_t$ if it is similar to patch $x_{t-1}$. This strategy accepts slow changes of the appearance.

3. *Temporal Pattern* (LOOP). This strategy, one of our contributions in this paper, first converts the tracker's trajectory to a sequence of distances to the online model and then searches for certain patterns in this sequence. The *closed loop* pattern is defined as follows: starting from a patch similar to the online model, the distance first exceeds the threshold $\theta$ and after a number of frames it becomes similar again (Fig. 1). In this case, all the frames within the pattern are used for update. This strategy exploits the property of an adaptive tracker. If the tracker drifts away from the object, it adapts to the appearance of the background and therefore it is very unlikely that it accidentally returns back to the object. If it returns, it strongly suggests that the increase of distance was due to changed appearance or other image perturbations. This strategy allows to accept patches with strong appearance variations but still representative for the object.

**Pruning events.** Our pruning event assumes that the tracked object is unique within the scene. If the tracker and the detector agree on the object position, all remaining detections are considered false positives and removed from the model. Section 4.2 demonstrates how pruning stabilizes the modeling.

## 4. Experiments

This section first analyzes the proposed growing strategies to identify the most reliable one. Next, we demonstrate the need for an object detector in the case of long-term tracking. We empirically show that growing alone is not sufficient and has to be coupled with model pruning to obtain a stable system. Finally, TMD is tested on standard benchmark sequences and new challenging videos.

**Evaluation of trajectories.** Fig. 3 depicts the continuous trajectory of the tracker (blue) and the corresponding ground truth (red) in the image space. Suppose that a validation process selects only certain parts (blue thick) of the raw trajectory; the non-selected parts represents frames where

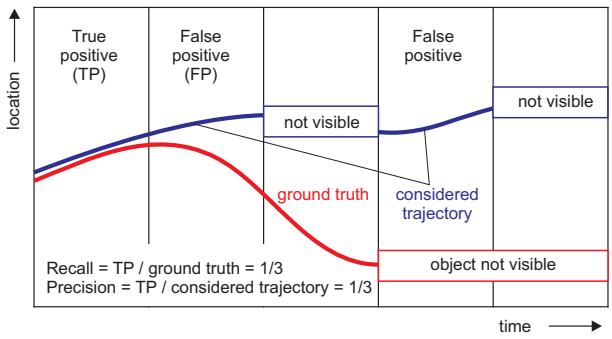


Figure 3. Introduction of precision/recall on imaginary tracking in 1D. Tracker's trajectory (blue) is compared with ground truth (red). The tracker discovers 33% of the ground truth (Recall = 33%) with every 3rd sample being correct (Precision = 33%).

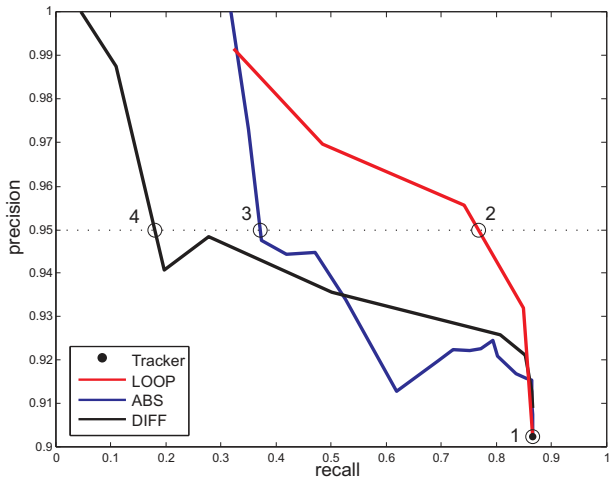


Figure 4. Precision/Recall characteristics of growing events as a function of parameter $\theta$ defining when two patches are "similar".

the object is not visible. By comparing the selected trajectory (STR) with the ground truth (GT), each frame is assigned a label: *True Positive* (TP) if the bounding box of the tracker overlaps with the ground truth by more than 70%, otherwise *False Positive* (FP) is assigned. The overlap is defined as a ratio of union to the intersection of the bounding boxes. The overall tracker performance is evaluated by two measures. Precision is the percentage of the selected trajectory that is correct: Precision $= |\text{TP}|/|\text{STR}|$. Recall represents the percentage of the ground truth that overlaps with the selected trajectory: Recall $= |\text{TP}|/|\text{GT}|$. These measures will be used for evaluation of the whole system and also for evaluation of quality of growing strategies.

### 4.1. Selection of the growing event

This experiment focuses on the performance of different growing events. Our goal is to identify the event that maximizes recall for a given precision (i.e. 95%) and to determine the value of parameter $\theta$. A benchmark face tracking video sequence from [12] is used for evaluation. Several frames from the sequence are shown in Fig. 8 (a). The sequence contains 1006 frames and shows four face targets in an indoor environment that undergo a variety of changes including: fast motion, out of plane rotation, partial and full occlusions. The subject that undergoes the most significant changes was selected for tracking. The target trajectory was manually marked; it was split into 16 continuous segments, separated by full occlusions. In this experiment, we are interested in the ability of identifying correct parts of the object's trajectory, therefore the tracker is initialized at the beginning of every segment and is run up to its end, including the occlusion. This produces 16 trajectories. The trajectories are analyzed by the growing event that accepts certain parts of them (see Sec. 3.2). The quality of the selection and its dependence upon the strategy's internal parameter is compared using precision/recall characteristics.

Fig. 4 shows the resulting precision/recall curves. The black dot in the bottom-right corner (point 1) corresponds to performance of the short-term tracker alone (the entire trajectory is considered) where recall is 87% and precision equals 90%. This performance corresponds to the case when the track is manually re-initialized after occlusion (i.e. as if an error-free detector is available). If the tracker is initialized only once, the recall drops to 23% since the object disappears from the scene and the tracker fails in all frames thereafter. The drop from 90% to 23% demonstrates that the object detector is essential for long-term tracking. All growing strategies produce the same result (point 1) if a large distance threshold $\theta$ is allowed, i.e. all samples on the trajectory are accepted by the growing events. As $\theta$ decreases, the strategies become more and more selective. The recall of ABS and DIFF significantly drops compared to LOOP strategy at precision of 92%. Notice the precision level of 95% denoted by the black dotted line. LOOP correctly identified almost 80% of the ground truth (point 2), while ABS 40% (point 3) and DIFF only 20% (point 4).

This experiment shows that the tracker trajectory can be analyzed by different strategies, characterized by the precision/recall tradeoff. The LOOP strategy performs significantly better than other approaches and will be used in following experiments with parameter $\theta$ corresponding to the point no. 2 in Fig. 4.

### 4.2. System performance

The aim of this experiment is to quantitatively evaluate the TMD system. As the system consists of several components, we demonstrate the performance/functionality of each of them. We start from the simplest form and progressively add one component at a time: (i) adaptive tracker, similar to [2, 4, 7], (ii) tracker + detector: the tracker is re-initialized if the detector finds an appearance similar to the

first frame, $L_t = \{x_0\}$, similar to [8], (iii) tracker + detector + growing: $L_t$ is growing and therefore allows more frequent re-initialization, (iv) full TMD: in addition to the previous case, the model is pruned. The combinations were tested on the same face sequence as in the previous experiment but the tracker is initialized just once.

**The evaluation of system's components** Fig. 5 shows the resulting precision/recall curves for different variants. The baseline short-term tracker (black) correctly tracks the target in the first few frames, but later the target moves away from the field of view and the tracker drifts away. The trajectory thus produced has 90% precision and 23% recall (point 3). By adding detection to the tracking process, no improvement was obtained, as the object model represented only by the first patch was never detected again. By incorporating model growing into the process (red curve) the detector gives many responses. This influences the recall in two ways. First, the recall increases since correct detections initialize the tracker in different frames. Second, the recall decreases since false positives incorrectly re-initialize the tracker. These two effects almost cancel each other resulting in slight increase of recall compared to simple short-term tracker. Precision drops significantly to 30% due to growing number of false positives. However, if the learning is coupled with pruning (blue curve), the approach significantly outperforms all the other variants. The maximum recall for precision 100% is 40% (point 2). If the entire trajectory is considered the system is able to localize 55% of target appearances and makes an error of 10% (point 1). Fig. 5 shows that there is a clear gain in performance when both model growing and pruning are employed. The pruning significantly reduces the number of false examples in the model and thus reduces the probability of re-initialization on an incorrect target.

Fig. 6 takes another look at the same experiment, i.e. displays cumulative number of false detections as a function of time for runs without (iii) and with pruning (iv). Notice the black circle; up to this point no growing update occurs, hence both approaches give the same number of detections. After the update, the number of false detections significantly increases for the simple growing strategy. Growing coupled with pruning keeps the number of false alarms always very low.

**System stability and detector improvement.** The TMD system was run repeatedly on the face video sequence, with the online model/detector being continuously updated. The results are presented in table 1. After each run (7 in total) the TMD system was evaluated using precision/recall. The performance of TMD was compared with the performance of the improving internal detector. This experiment was repeated four-times for different parameter $\theta$.

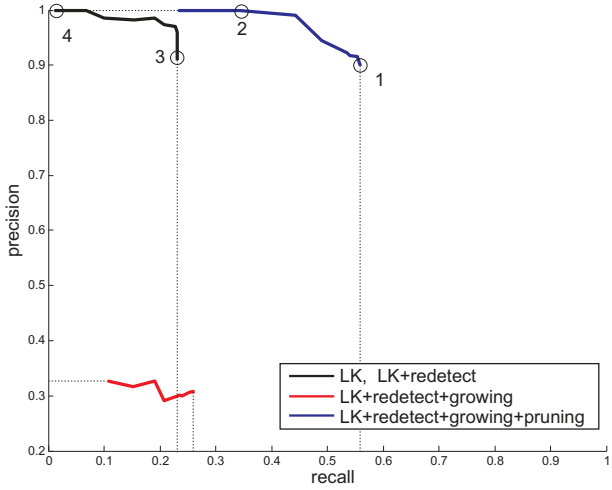


Figure 5. Performance of the adaptive tracker (black), tracker + detector + growing (red) and the full TMD system (blue). The curves are produced by thresholding the confidence score of resulting trajectories.

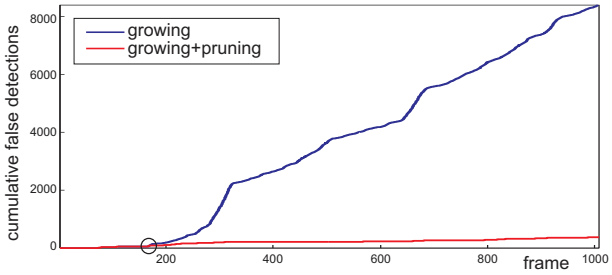


Figure 6. Cumulative number of false detections during tracking increases rapidly if only model growing is used. The growth is stabilized by introduction of model pruning.

Notice the third sub-table with $\theta = 0.20$ which corresponds to the value selected earlier. In the first iteration, TMD discovers 22% of the trajectory, while the detector 8% at the same precision of 100%. With every iteration, the recall of both of them increases. At the 7th iteration, TMD finds 72% and the detector 64% of the target trajectory with high precision of 87% and 89% respectively. The model obtained after the 7th iteration is depicted in Fig. 7.

For $\theta = 0.05$ the learning process does not start at all, since the first appearance is never discovered again (the requirement on similarity is too high). This demonstrate the weakness of the LOOP strategy, i.e. the object may never come back to the first appearance. For $\theta = 0.30$ the overall performance first increases but around 5th iteration it slightly drops and oscillates.

The experiment can be concluded by the following observations: (i) TMD is improving over time for wide range of parameter $\theta$, which suggests that the learning have potential to work if other growing/pruning events are used instead, (ii) the benefit from the short-term tracker is decreas-

| $\theta = 0.05$ | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| TMD | Prec | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Rec | 0.01 | 0.01 | 0.02 | 0.02 | 0.03 | 0.03 | 0.03 |
| DT | Prec | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | Rec | 0.00 | 0.01 | 0.01 | 0.01 | 0.02 | 0.02 | 0.02 |
| $\theta = 0.10$ | | | | | | | | |
| TMD | Prec | 1.00 | 1.00 | 0.99 | 0.99 | 0.99 | 0.95 | 0.95 |
| | Rec | 0.04 | 0.35 | 0.59 | 0.68 | 0.70 | 0.69 | 0.71 |
| DT | Prec | 1.00 | 0.99 | 1.00 | 0.99 | 0.99 | 0.98 | 0.98 |
| | Rec | 0.01 | 0.25 | 0.49 | 0.58 | 0.58 | 0.61 | 0.63 |
| $\theta = 0.20$ | | | | | | | | |
| TMD | Prec | 1.00 | 0.97 | 0.94 | 0.91 | 0.91 | 0.90 | 0.87 |
| | Rec | 0.22 | 0.59 | 0.70 | 0.72 | 0.72 | 0.71 | 0.72 |
| DT | Prec | 1.00 | 1.00 | 0.97 | 0.93 | 0.95 | 0.94 | 0.89 |
| | Rec | 0.08 | 0.46 | 0.59 | 0.61 | 0.65 | 0.65 | 0.64 |
| $\theta = 0.30$ | | | | | | | | |
| TMD | Prec | 0.98 | 0.99 | 0.94 | 0.94 | 0.90 | 0.87 | 0.87 |
| | Rec | 0.09 | 0.57 | 0.73 | 0.79 | 0.78 | 0.77 | 0.65 |
| DT | Pre | 1.00 | 1.00 | 0.99 | 0.97 | 0.94 | 0.90 | 0.93 |
| | Rec | 0.01 | 0.35 | 0.57 | 0.66 | 0.68 | 0.67 | 0.59 |

Table 1. Performance of TMD and the internal online detector (DT) as a function of number of iterations.

Figure 7. Online model obtained by the LOOP growing event contains examples that are very diverse but correctly represent the object. The LOOP exploits drift of adaptive trackers. The patch selected for tracking is denoted by the red square.

ing as the system learns the object appearance and the tracking gradually progresses to tracking-by-detection without the need for off-line training.

**Long-term tracking of diverse objects.** The aim of this experiment is to evaluate the proposed approach on videos containing diverse objects. The system was tested on four videos, two of them are publicly available, the other two are new and extremely challenging for standard approaches. The following videos are used: (1) *Plush Toy* [10], consisting of 1344 frames (1:07) that include slow movements, illumination changes and small appearance changes; (2) *Pedestrian* from Caviar dataset, TwoEnterShop2cor, frames 81-550 (0:23) that include full occlusion; (3) *Volkswagen* on highway, containing 8576 frames (7:08) that include occlusions, disappearance from the field of view, similar objects and camera shaking; (4) *Motorbike*, containing 2917 frames (2:33) that include occlusion, disappearance from the field of view, fast movements, significant appearance changes. Fig. 8 shows some frames from the sequences. The sequences and hand labeled ground truth are available online.[1]

Table 2 shows the resulting performance, where our approach (TMD) is compared to the short-term tracker (LK),

| Sequence | LK | | Online [7] | | Semi [8] | | TMD | |
|---|---|---|---|---|---|---|---|---|
| | Prec | Rec | Prec | Rec | Prec | Rec | Prec | Rec |
| Plush Toy | 0.83 | 0.83 | 0.62 | 0.62 | 0.39 | 0.39 | **0.98** | **0.98** |
| Pedestrian | 0.31 | 0.46 | 0.26 | 0.39 | 0.20 | 0.29 | **0.96** | **0.81** |
| Volkswagen | 0.75 | 0.05 | 0.18 | 0.31 | 0.04 | 0.07 | **0.88** | **0.82** |
| Motocross | 1.00 | 0.02 | 0.01 | 0.02 | 0.04 | 0.02 | **0.96** | **0.54** |

Table 2. System performance evaluated on standard (Plush Toy, Pedestrian) and extremely challenging (Volkswagen, Motocross) sequences.

Online Boosting [7] and Semi Boosting [8]. The performance of TMD improves with multiple runs, but only the first run is considered here. The Online and Semi-Boosting tracker do not track scale changes which significantly affects their results. The "Plush Toy" sequence can be successfully tracked by LK alone, except for a slight drift at the end. TMD is able to correct for this drift, leading to higher performance. The "Pedestrian" sequence is more challenging as it contains full occlusion in the middle. TMD is able to recover after the occlusion and continue successfully in tracking. The appearance of the selected object does not change much. The "Car" and "Motorbike" sequences are very challenging for standard trackers and we are not aware of any method able to achieve similar performance to ours with no off-line training.

## Conclusions

In this paper we introduced a new approach for general long-term tracking. The system is capable of continuous tracking while building a reliable online detector without using any prior information on the target. The key components, the growing and pruning events are novel for such systems and crucial for long term tracking as we demonstrate in the experiments. Furthermore, we explicitly address the problem of recovering from tracking failures, which are inevitable in any long term tracking system. Simple solutions have been adopted to make the approach highly efficient and suitable for real time video processing in surveillance applications. The current Matlab implementation operates at 20 fps. The proposed system was extensively tested on a variety of objects and the results show clear improvement compared to standard trackers. In the future we plan to perform thorough analysis of the learning based on growing and pruning, apply the TMD framework to different combinations of trackers and detectors and extend the approach for multi-target tracking.

**Acknowledgment.** This research was supported by EU VIDI-Video IST-2-045547 grant (ZK, KM) and EC ICT-215078 project DIPLECS (JM). The authors would like to thank Helmut Grabner for kind offering of his tracking results.

[1] http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/

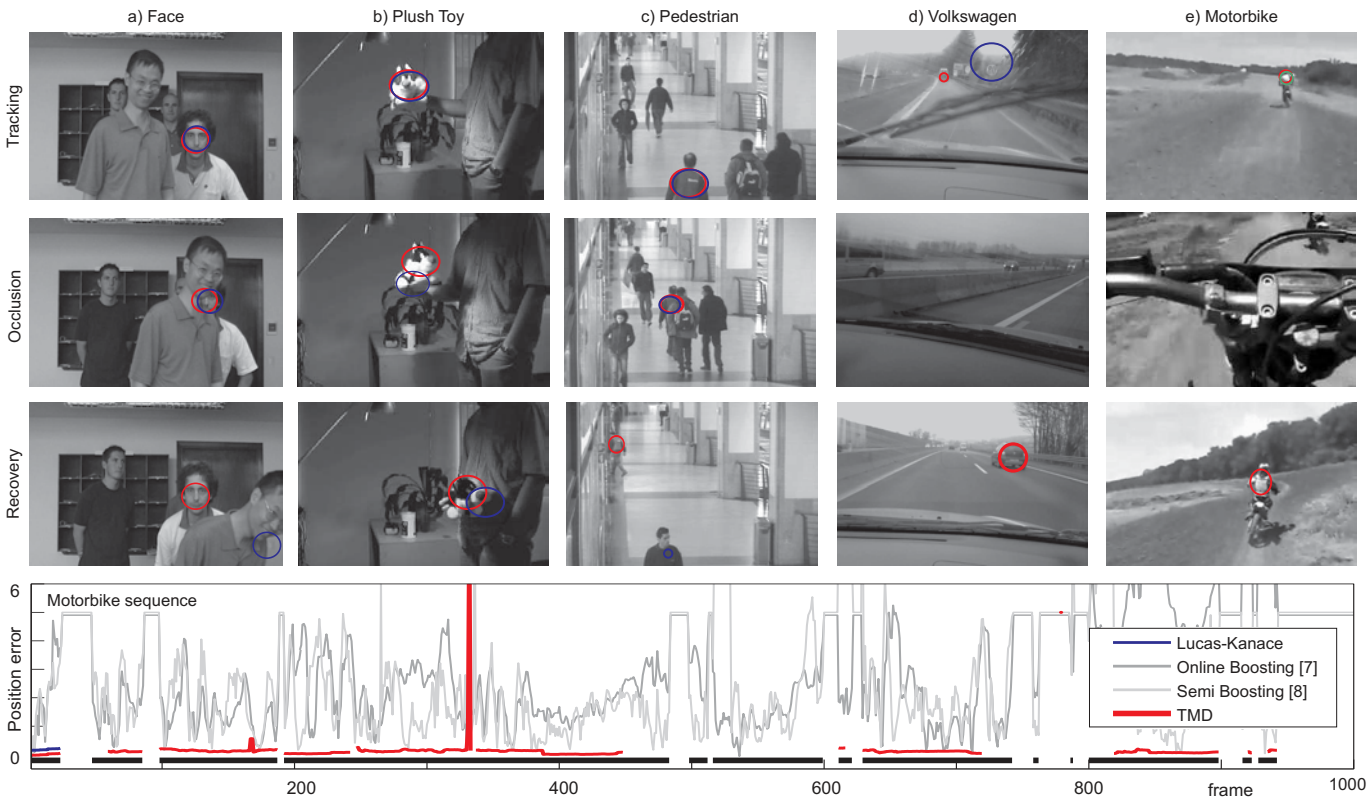


Figure 8. Examples from tested sequences: (red) TMD, (yellow) LK. Error of object localization on the Motocross sequence. Black line indicates the object visibility. See the webpage of the first author for full sequences.

## References


[1] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. *CVPR*, 2008. 1

[2] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271, 2007. 1, 5

[3] L. Breiman. Random forests. *ML*, 45(1):5–32, 2001. 4

[4] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005. 1, 5

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–577, 2003. 1

[6] L. Ellis, N. Dowson, J. Matas, and R. Bowden. Linear predictors for fast simultaneous modeling and tracking. pages 1–8, 2007. 1

[7] H. Grabner and H. Bischof. On-line boosting and vision. *CVPR*, 2006. 1, 3, 5, 7

[8] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. *ECCV*, 2008. 1, 6, 7

[9] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. *CVPR*, 2005. 1, 4

[10] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. *NIPS*, 2005. 7

[11] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, 81:674–679, 1981. 1, 3

[12] E. Maggio, E. Piccardo, C. Regazzoni, and A. Cavallaro. Particle phd filtering for multi-target visual tracking. *ICASSP*, 2007. 5

[13] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *PAMI*, 26(6):810–815, 2004. 1

[14] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002. 3

[15] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *CVPR*, 2007. 4

[16] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. *ECCV*, 2006. 1

[17] D. Ramanan, D. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. *CVPR*, 2005. 1

[18] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. *WACV*, 2005. 1

[19] P. Roth, M. Donoser, and H. Bischof. On-line learning of unknown hand held objects via tracking. *ICVS*, 2006. 1

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001. 1, 3