INCREMENTAL LEARNING AND VALIDATION OF SEQUENTIAL PREDICTORS IN VIDEO BROWSING APPLICATION

David Hurych and Tomáš Svoboda
Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University in Prague, Karlovo Náměstí 13, Prague, Czech Republic

Keywords: Sequential, Linear, Predictor, Video, Browsing, Unsupervised, Incremental, Learning.

Abstract: Loss-of-track detection (tracking validation) and automatic tracker adaptation to new object appearances are attractive topics in computer vision. We apply very efficient learnable sequential predictors in order to address both issues. Validation is done by clustering of the sequential predictor responses. No aditional object model for validation is needed. The paper also proposes an incremental learning procedure that accommodates changing object appearance, which mainly improves the recall of the tracker/detector. Exemplars for the incremental learning are collected automatically, no user interaction is required. The aditional training examples are selected automatically using the tracker stability computed for each potential aditional training example. Coupled with a sparsely applied SIFT or SURF based detector the method is employed for object localization in videos. Our Matlab implementation scans videosequences up to eight times faster than the actual frame rate. A standard-length movie can be thus searched through in terms of minutes.

1 INTRODUCTION

Learnable visual trackers have recently proved their wide applicability in object tracking in video. The tracking poses essentially two main challenges: i) adapting to changing appearance, ii) detecting tracker failure – loss of track. The paper addresses both issues but contributes mainly to the adaptation problem. We propose to solve the adaptation problem by an incremental learning, which accomodates changing appearance whilst tracking. We also suggest a fast method for tracking validation (i.e. loss-of-track detection) which uses the same model as for tracking and does not need any additional object model. The predictor needs only a very short (seconds) offline learning stage before the tracking starts. The tracking itself is then tremendously efficient, much faster than real-time.

Tracker adaptation and loss-of-track detection have been active topics for many years. Jepson et al. (Jepson et al., 2008) proposed WSL tracker (3 components - Wandering, Stable and Lost) - an *adaptive* appearance model which deals with partial occlusion and change in object appearance. It is a wavelet-based model, which allows to maintain a natural measure of the *stability* of the observed image structure during tracking. This approach is robust and works well with slowly changing object appearance. However, a high computational overhead precludes real-time applications. Lim et al. (Ross et al., 2008) propose an algorithm for incremental learning and adaptation of low dimensional eigenspace object representation with update of the sample mean and eigenbasis. Their approach appears to be robust to sudden illumination changes and does not need offline learning phase before tracking however, the algorithm speed does not fit our needs. For template-based trackers the adaptation means continuous update of the tracked template. Tracking systems with *naive updates* update the template after every tracking step (Shi and Tomasi, 1994). Sub-pixel errors inherent to each match are stored in each update and these errors gradually accumulate resulting in the template drifting off the feature. Despite this drawback, naive update is still usually better choice than no update at all. Matthews et al. in (Matthews et al., 2004) propose a *strategic update* approach, which trades off mismatch error and drift. It is a simple but effective extension of the naive update. There are two template models kept during the tracking. The *updating template* is used for an initial alignment and the template from the first frame is than used in the error correction phase after alignment. If the size of correction is too large, the algorithm acts conservatively by preventing the tem-

Hurych D. and Svoboda T. (2010).
INCREMENTAL LEARNING AND VALIDATION OF SEQUENTIAL PREDICTORS IN VIDEO BROWSING APPLICATION.
In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 467-474
DOI: 10.5220/0002836504670474

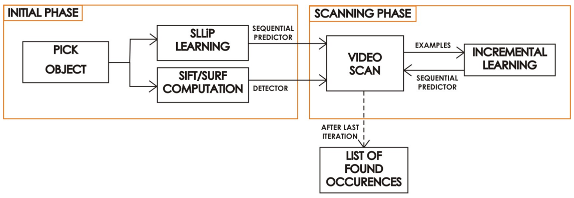

Figure 1: Video browsing procedure.

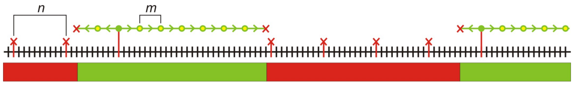

Figure 2: A typical video scan process. Vertical red lines depict frames, where the object detection was run. Red cross means negative detection or tracking failure. Green line shows backward and forward object tracking. Green circle means positive object detection and yellow circle depicts successful validation.

plate to be updated from the current frame.

Recently, some authors wanted to bypass an exhaustive off-line learning stage. Purely on-line learning has been proposed by Ellis et al. in (Ellis et al., 2008), where a bank of local linear predictors (LLiPs), spatially disposed over the object, are on-line learned and the appearance model of the object is learnt on-the-fly by clustering sub-sampled image templates. The templates are clustered using the medoidshift algorithm. The clusters of appearance templates allow to identify different views or aspects of the target and also allow to choose the bank of LLiPs most suitable for current appearance. The algorithm also evaluates the performance of particular LLiPs. When the performance of some predictor is too low, it is discarded and a new predictor is learned on-line as a replacement. In comparison to our work, we do not throw away the predictors in sequence, but we incrementally train them with new object appearances in order to improve their performance.

Our learnable and adaptive tracking method, coupled with a sparsely applied SIFT (Lowe, 2004) or SURF (Bay et al., 2006) based detector, is applied for faster than real-time linear video browsing. The goal is to find all object occurrences in a movie. One of possible solutions of video browsing task would be to use a general object detector in every frame. As it appears (Yilmaz et al., 2006), (Murphy-Chutorian and Trivedi, 2009), it is preferable to use a combination of an object detector and a tracker in order to speed up the browsing algorithm and also to increase the true positive detections. We indeed aim at processing rates higher than real-time which would allow almost interactive processing of lengthy videos. Our yet preliminary Matlab implementation can search through videos up to eight times faster than the real video frame rate.

2 LEARNING, TRACKING, VALIDATION AND INCREMENTAL LEARNING

User initiates the whole process by selecting a rectangular patch with the object of interest in one image. This sample patch is artificially perturbed and a sequential predictor is learned (Zimmermann et al., 2009). Computation of a few SIFT or SURF object descriptors completes the initial phase of the algorithm, see Figure 1. The scanning phase of algorithm combines predictor based tracking, its validation, and a sparse object detection. The predictor is incrementally re-trained for new object appearances. Examples for the incremental learning are selected automatically with no user interaction.

The scanning phase starts with the object detection running every $n-$th frame (typically with the step of 20 frames) until the first object location is found. The tracker starts from this frame on the detected position both in backward and forward directions. Backward tracking scans frames which were skipped during the detection phase and runs until the loss-of-track or until it reaches the frame with last found occurrence of the object. Forward tracking runs until the loss-of-track or end of sequence. The detector starts again once the track is lost. Tracking itself is validated every $m-$th frame (typically every 10 frames). The scanning procedure is depicted on Figure 2.

One object sample represents only one object appearance. The predictor is incrementally re-trained as more examples become available from the scanning procedure. The next iteration naturally scans only images where the object was not tracked in the preceding iterations.

Training examples for incremental learning are selected automatically. The most problematic images-examples are actually the most useful for incremental training of the predictor. In order to evaluate the usefulness of a particular example we suggest a stability measure. The measure is based on few extra predictions of the predictor on a single frame. It means, that we let the sequential predictor track the object in a single static image and we observe the predictors' behavior. See Section 2.3 for more details.

The sequential linear predictor validates itself. Naturally, an object detector may be also used to validate the tracking. For example well trained face detector will do the same or better job when used to validate human face tracking. Motivation for using the sequential predictor for validation is its extreme efficiency, and robust performance. For more details about the tracking validation, see section 2.2.

2.1 Incremental Learning of Sequential Learnable Linear Predictor

We extend min-max learning of the Sequential learnable linear predictors (SLLiP) by Zimmermann et al. (Zimmermann et al., 2009) in order to predict not only translation but also the affine deformation of the object. Next extension is the incremental learning of new object appearances. The predictor essentially estimates motion and deformation parameters directly from image intensities. It requires an offline learning stage before the tracking starts. The learning stage consists of generating exemplars and estimation of regression functions. We use 2 SLLiPs - first for 2D motion estimation (2 parameters) and second for affine warp estimation (4 parameters). We have experimentally verified that, especially for low number of training examples, this configuration is more stable than using just one SLLiP to predict all 6 parameters at once. Using smaller training set decreases the necessary learning time which is important for the foreseen applications. Because of speed we opted for least squares learning of SLLiPs similarly, as suggested by Zimmermann et al. in their any-time learning algorithm (Zimmermann et al., 2009).

Let denote the translation parameters vector $\mathbf{t}_t = [\Delta x, \Delta y]^T$ estimated by the first SLLiP, and the affine warp is parameterized by the parameters vector $\mathbf{t}_a = [\alpha, \beta, \Delta s_x, \Delta s_y]^T$ which is estimated by the second SLLiP. The 2×2 affine warp matrix $\mathtt{A}$ is computed as

$$\mathtt{A} = \mathtt{R}_\alpha \mathtt{R}_{-\beta} \mathtt{S} \mathtt{R}_\beta\,, \tag{1}$$

where $\mathtt{R}$ are standard 2D rotation matrices parameterized by the angles α, β and $\mathtt{S}$ is the scale matrix

$$\mathtt{S} = \begin{bmatrix} 1+\Delta s_x & 0 \\ 0 & 1+\Delta s_y \end{bmatrix}. \tag{2}$$

Than the image point $\mathbf{x} = [x, y]^T$ is transformed between two consecutive images using estimated parameters accordingly

$$\begin{aligned} \mathbf{x}' &= \mathtt{A}\mathbf{x} + \mathbf{t}_t \\ &= \mathtt{R}_\alpha \mathtt{R}_{-\beta} \mathtt{S} \mathtt{R}_\beta \mathbf{x} + \mathbf{t}_t, \end{aligned} \tag{3}$$

Tracking, learning and incremental learning will be explained for SLLiP with general parameters vector $\mathbf{t}$. Equations are valid for both SLLiPs, which we use. SLLiP is simply a sequence of linear predictors. Predictors in this sequence estimate the parameters one after each other (see equation 4), thus each improving the result of previous predictor estimation and lowering the error of estimation. SLLiP tracks according to

$$\begin{aligned} \mathbf{t}_1 &= \mathtt{H}_1 I(X_1) \\ \mathbf{t}_2 &= \mathtt{H}_2 I(\mathbf{t}_1 \circ X_2) \\ \mathbf{t}_3 &= \mathtt{H}_3 I(\mathbf{t}_2 \circ X_3) \\ &\vdots \\ \mathbf{t} &= \bigcirc_{(i=1,\ldots,k)} \mathbf{t}_i, \end{aligned} \tag{4}$$

where I is current image and X is a set of 2D coordinates spread over the the object patch - it is called *support set*. $I(X)$ is a vector of image intensities collected at image coordinates X. Operation $\circ$ means transformation of support set points using Equation 3, i.e. aligning the support set to fit the object using parameters estimated by the previous predictor in the sequence. Final result of the prediction is vector $\mathbf{t}$ which combines results of all predictions in the sequence. The model θ_s for SLLiP is formed by the sequence of predictors $\theta_s = |\{\mathtt{H}_1, X_1\}, \{\mathtt{H}_2, X_2\}, \ldots, \{\mathtt{H}_k, X_k\}|$. Matrices $\mathtt{H}_1, \mathtt{H}_2, \ldots, \mathtt{H}_k$ are linear regression matrices which are learned from training data.

In our algorithm, the SLLiP is learned from one image only and it is incrementally (re-)learned after each video scan. A few thousands of training examples are artificially generated from the first image using random perturbations of parameters in vector $\mathbf{t}$, warping the support set accordingly and collecting the image intensities. The column vectors of collected image intensities are stored in matrix $\mathtt{D}_i$ and perturbed parameters in matrix $\mathtt{T}_i$ columnwise. Each regression matrix in SLLiP is trained using the least squares method $\mathtt{H}_i = \mathtt{T}_i \mathtt{D}_i^T \left(\mathtt{D}_i \mathtt{D}_i^T\right)^{-1}$. The initial learning phase takes 5 or 6 seconds on a standard PC.

More images (around 400) are selected for incremental learning from all images gathered during last scanning iteration. From each of the additional exemplars 10 training examples are generated. This procedure provides additional 4000 training examples after each particular video scan. It is worth to note that this process is completely automatic, no user interaction is required. Incremental learning comprises update of regression matrices $\mathtt{H}_i, i = 1, \ldots, k$. An efficient way of updating regression matrices was proposed by Hinterstoisser et al. in (Hinterstoisser et al., 2008).

Each regression matrix $\mathtt{H}_i$ may be computed alternatively

$$\mathtt{H}_i = \mathtt{Y}_i \mathtt{Z}_i, \tag{5}$$

where $\mathtt{Y}_i = \mathtt{T}_i \mathtt{D}_i^T$ and $\mathtt{Z}_i = \left(\mathtt{D}_i \mathtt{D}_i^T\right)^{-1}$. Lets denote $\mathtt{Y}_i^j, \mathtt{Z}_i^j$, where j indexes the training examples. New training example $\mathbf{d} = I(X)$ with parameters $\mathbf{t}$ is incorporated into the predictor as follows

$$\begin{aligned} \mathtt{Y}_i^{j+1} &= \mathtt{Y}_i^j + \mathbf{t}\mathbf{d}^T \\ \mathtt{Z}_i^{j+1} &= \mathtt{Z}_i^j - \frac{\mathtt{Z}_i^j \mathbf{d}\mathbf{d}^T \mathtt{Z}_i^j}{1 + \mathbf{d}^T \mathtt{Z}_i^j \mathbf{d}}. \end{aligned} \quad (6)$$

After updating matrices $\mathtt{Y}_i$ and $\mathtt{Z}_i$ we may also update the regression matrices $\mathtt{H}_i$ using equation 5. For more details about incremental learning see (Hinterstoisser et al., 2008).

2.2 Validation by Voting

To validate the tracking (i.e. detecting loss-of-track) we use the same sequential linear predictor as for tracking. We utilize the fact that the predictor is trained to point to the center of this object when initialized in a close neighborhood. On the contrary, when initialized on the background, the estimation of 2D motion is expected to be random.

We initialize the predictor several times on a regular grid (validation grid - depicted by red crosses in Figure 3) in the close neighborhood of current position of the tracker. The close neighborhood is defined as 2D motion range, for which the predictor was trained. In our case the range is $\pm(patch\_width/4)$ and $\pm(patch\_height/4)$. The validation grid is deformed according to estimated parameters. Then we observe the 2D vectors, which should point to the center of the object, i.e. current tracker position in the image. When all (or sufficient number of) the vectors point to the same pixel, which is also the current tracker position, we consider the tracker to be on its track. Otherwise, when the 2D vectors are pointing to some random directions, we say that the track is lost, see Figure 3.

A threshold value is needed in order to recognize if the sum of votes, which point to the center of object, is big enough to pass the validation. The threshold is set automatically from examples collected during the video scan. At first iteration, when no threshold is available, first few (tens) validations are done by the object detector and SLLiP simultaneously. When the detector votes for positive validation, also the current sum of votes is taken as positive example. Negative examples (sums of votes) are collected by placing the validation grid on other parts of the image, where the object does not appear. Gaussian distributions are fitted into positive and negative examples and the classical Bayes threshold is found. Both negative and positive cases are considered to appear equally likely. In subsequent iterations, the additional training examples are also used for threshold update.

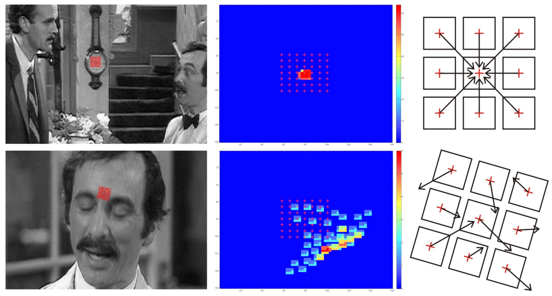

Figure 3: Example of predictor validation. The first row shows successful validation of clock tracking. Second row shows loss-of-track caused by a sudden scene change just after a video cut. Red crosses depict pixels, where the predictor was initialized - validation grid. Right column of pictures illustrates the idea of validation using linear predictors and the middle column shows the collected votes for the center of the object in normalized space.

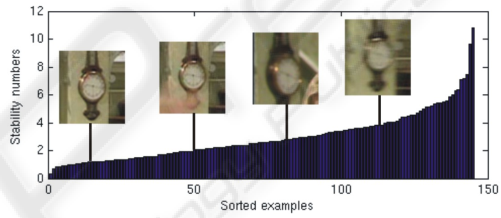

Figure 4: Blue bars depict sorted stability numbers. The left most clock image was used for predictor training. The other occurrences obtained during tracking were automatically evaluated as more difficult examples for the tracker. Clearly, the higher stability measure, the more difficult case for the predictor.

2.3 Stability Measure and Examples Selection for the Incremental Learning

Selecting only *relevant* examples for training may speed up the learning as well as increase the performance. Clearly relevant examples are those which contain the object but were not included in the previous training examples. The predictor has of course problems to handle new object appearances and it is likely, that it will loose the track. It is reasonable to presume, that these new difficult (and useful) examples should appear near frames, where the loss-of-track was detected. We need to examine the object occurrences, which appeared near loss-of-track frames, in order to capture the most interesting examples for incremental learning. We propose the *stability measure* for evaluation of these object occurrences.

When we let the predictor track object on a single frame, we would expect the tracker to stay still in objects' position with no additional change of parameters. However, due to inherent noise in the data the

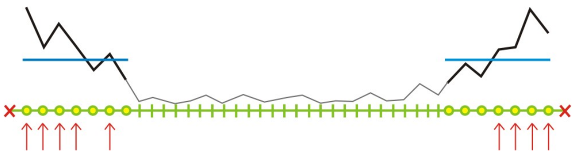

Figure 5: Illustration of examples selection for incremental learning. Green line depicts one interval - subsequence of video frames, where the object was found during scanning procedure. Only few images near the beginning and end of the interval are examined. Yellow circles mean successful validation. The black curve depicts computed stability measure on particular frames. The examples with stability number above the blue line are considered as useful for incremental learning. Selected examples are marked by red arrows.

predictor predicts non-zero parameters even when initiated on the correct position. The parameters changes are accumulated and their sum-of-squares is computed after 10 tracking steps. Let $\mathbf{t}$ be the vector of parameters estimated during tracking and $\mathbf{p}_i$ vector of parameters obtained in $i-$th step of this single frame tracking. The *stability number* s for current frame is computed as $s = \sum_{i=1}^{10} \parallel \mathbf{t} - \mathbf{p}_i \parallel^2$. Clearly, the higher value the more difficult example, see Figure 4. Parameters changes in both vectors are made relative to particular ranges, in order to obtain stability number, which is not dependent on different parameters units. Using this stability number we may evaluate how useful (difficult) is the examined object occurrence.

We use this stability number to select a fixed number of additional training examples from each interval obtained during one video scan. Each interval is a continuous subsequence of images from the whole video sequence (one interval is depicted as a green line in Figure 2).

We search for the best additional training examples near the borders of each interval. We go through fixed number of images from the start of the interval forwards and backwards from the end of the interval, while computing the stability number on tracker positions. Finally, the algorithm selects the examples with high stability number for incremental learning. Tracker positions in these images have also passed validation and we expect them to be well aligned to the object. The procedure of examples selection is depicted in Figure 5.

3 EXPERIMENTS

Real sequences used in experiments includes an episode from Fawlty Towers series (33 minutes, 720×576), and Groundhog Day movie (1 hour 37 minutes, 640×384). Several objects were tested, see Figure 6. The ground truth data for the Groundhog Day were kindly provided by Josef Šivic and they are the same as in (Sivic and Zisserman, 2009). We have manually labeled ground truth for two tested objects in Fawlty Towers. Third tested sequence captures a human moving in front of the camera (2 minutes 50 seconds, 640×480), see Figure 7. Matlab implementation of the algorithm was used for all experiments. SIFT and SURF object detectors are publicly available MEX implementations. Mostly, the standard *precision* and *recall* are used to evaluate the results. Let TP denote the *true positives*, FP denote the *false positives* and FN denote the *false negatives*. Than the precision and recall are computed accordingly

Figure 6: Tested objects are marked with red rectangle.

Figure 7: Here you may see examples of human face data used in experiment. All images are extracted from one video sequence. Note significant deformations and variations in illumination.

$$precision = \frac{TP}{TP+FP} \quad (7)$$

$$recall = \frac{TP}{TP+FN}. \quad (8)$$

The experiments are organized as follows. The first experiment (Section 3.1) shows the effect of incremental learning on resulting precision and recall. In Section 3.2 we evaluate the overall performance of the algorithm. Next we compare tracking validation by SIFT and by SLLiP. Finally Table 3, Table 4 and Table 5 show comparison of SIFT detection in every frame with one iteration of our algorithm.

Table 1: Incremental learning evaluation for the clock object from the Fawlty towers episode. The video scan was running 76 frames per second in average.

	precision	recall	cumulative time
iter_0	0.86	0.61	13 min 42 sec
iter_1	0.81	0.63	23 min 18 sec
iter_2	0.81	0.64	32 min 21 sec

Table 2: Incremental learning evaluation for human face. The first iteration of video scan was running 21 frames per second in average. Browsing time was increased by using the face detector instead of SURF.

	precision	recall	cumulative time
iter_0	0.99	0.70	4 min 5 sec
iter_1	0.98	0.79	5 min 2 sec
iter_2	0.98	0.81	5 min 27 sec

3.1 Incremental Learning Evaluation

This experiment shows the improvement gained by the automatic incremental learning. At first we run one iteration of video scan using sequential predictor trained on one image only (in Table 1 denoted as *iter_0*). Next, we evaluate results after first and second incremental learning (*iter_1* and *iter_2*). Two objects were tested. First was the picture object in Fawlty Towers video (see Figure 6 top right image). The SURF based detector was used for picture detection with step $n = 20$ and sequential predictor for validation with step $m = 10$. Incremental learning improves the recall while keeping high precision, see Table 1.

Second tested object was a human face (see Figure 7). In this case the object was difficult to track with SLLiP learned only from one image, because the appearance of the face changed significantly during the sequence. The lighting conditions were challenging and the human face undergoes various rotations and scale changes. We have chosen this sequence in combination with the face detector (instead of SIFT/SURF) to see how the incremental learning helps to improve tracking results on complex non-rigid object. In this case incremental learning also improved the performance of the tracker. See Table 2 for results.

The high precision obtained in the face experiment was caused by flawless face detection, which did not return any false positive. You may see a few images of SLLiP tracker aligned on human face on Figure 8.

3.2 Results of Detection and Tracking

One iteration of the algorithm in Fawlty Towers series runs 3−times faster than real-time and more than

Figure 8: Examples of face tracking results. Red rectangle depicts SLLiP tracker aligned on human face.

8−times faster for the Groundhog Day movie. The detector was run every $n = 20$ frames and validation every $m = 10$ frames while tracking. In the sequence with human face the browsing time was almost twice the real-time, even for detection step $n = 40$. It was caused by the face detector which runs much slower than SURF. The difference in browsing times in Fawlty Towers and Groundhog Day is caused mainly by the different video resolution. Processing of higher resolution images and more complex scenes is slown down by the object detector. Even shorter browsing times may be achieved by increasing the detection interval n. Selecting the right interval depends on our expectation of the shortest time interval, where the object may appear. Reasonable values for detection interval are between 20 and 60 frames. Increasing the validation interval m to more than 10 generates more false positives and since the validation runs very fast, it is not necessary to validate with a bigger step.

Next we compare the performance of predictor validation with SIFT validation. The average time of one SIFT validation was 179 milliseconds and average time of one predictor validation was 33 milliseconds. The resulting recall of video browsing for clock object was 0.58 with SIFT and 0.61 with predictor validation, while the precision was 0.9 for SIFT validation and 0.86 for SLLiP validation. Recall for the picture object was 0.94 with SIFT validation and 0.95 with predictor, while the precision was 0.84 for both. Predictor validation gives comparable precision and

Table 3: Comparison of SIFT object detection only and one iteration of our algorithm on Fawlty Towers - clock and picture.

	clock (FT)	**picture**
SIFT detector on every frame - without tracking		
browsing time	32 h. 56 m.	33 h. 29 m.
scanning speed (fps)	0.4	0.4
obtained occurrences	2440	2140
true positives	2411	1996
false positives	29	144
precision	0.99	0.93
recall	0.43	0.89
SURF detect., SLLiP track. and valid.		
browsing time	13 m. 42 s.	11 m. 40 s.
scanning speed (fps)	61	71
obtained occurrences	4026	2520
true positives	3462	2131
false positives	564	389
precision	0.86	0.85
recall	0.61	0.95

Table 4: Comparison of SIFT object detection only and one iteration of our algorithm on Groundhog Day - alarm clock and clock.

	alarm clock	**clock** (GhD)
SIFT detector on every frame - without tracking		
browsing time	48 h. 43 m.	48 h. 13 m.
scanning speed (fps)	0.8	0.8
obtained occurrences	1888	855
true positives	1811	801
false positives	77	54
precision	0.96	0.94
recall	0.37	0.29
SURF detect., SLLiP track. and valid.		
browsing time	16 m. 46 s.	12 m. 48 s.
scanning speed (fps)	144	189
obtained occurrences	1345	2034
true positives	1125	1520
false positives	220	514
precision	0.84	0.75
recall	0.23	0.55

recall in much shorter time, which also saves time in the whole scanning iteration. Tables 3, 4 and 5 show the results for 5 tested objects obtained in one scanning iteration. The results of the video browsing algorithm are compared to the results produced by the SIFT detector only.

SURF detection on every frame was tested too, but the results contained large number of false positives.

Table 5: Comparison of SIFT object detection only and one iteration of our algorithm on Groundhog Day - PHIL sign.

	PHIL sign
SIFT detector on every frame - without tracking	
browsing time	48 h. 7 m.
scanning speed (fps)	0.8
obtained occurrences	2597
true positives	2293
false positives	304
precision	0.88
recall	0.72
SURF detect., SLLiP track. and valid.	
browsing time	15 m. 15 s.
scanning speed (fps)	159
obtained occurrences	4038
true positives	2361
false positives	1677
precision	0.58
recall	0.74

Recall was comparable to SIFT detector, but precision was very low. We are using the SURF detector because it runs much faster than SIFT, but we need to use predictor validation after every positive detection because of the large number of false positives. The results show that even after a single iteration the algorithm gives results comparable with SIFT detection only.

4 CONCLUSIONS

We have shown that an incremental learning of sequential predictor significantly improves its robustness. It increases the recall while keeping high precision. Proposed method for collecting additional training examples is completely automatic and requires no user interaction. Stability number well describes the condition of the tracker on particular image and prooves to be good criteria for training examples selection. Validation by clustering SLLiP responses works reliably and very fast.

When coupled with a sparsely applied object detector the system can search for objects through videos several times faster than real time despite the current rather preliminary Matlab implementation. The complete system for video browsing works very well with simple objects. Performance for more complex 3D objects (when using SIFT/SURF for detection) are not yet entirely satisfactory. It is mainly the detector that hinders the recognition rate. The tracker itself may be incrementally learned for new appearances of the object and it works better with every iteration. This was verified on face tracking where a robust face detector was applied. We plan to extend the sequential predictor in a way that would allow its application as a detector.

REFERENCES

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2006). Speeded-up robust features. In *Proceedings of IEEE European Conference on Computer Vision*, pages 404–417.

Ellis, L., Matas, J., and Bowden, R. (2008). On-line learning and partitioning of linear displacement predictors for tracking. In *Proceedings of the 19th British Machine Vision Conference*, pages 33–42.

Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., and Lepetit, V. (2008). Online learning of patch perspective rectification for efficient object detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Jepson, A., Fleet, D., and El-Maraghi, T. (2008). Robust online appearance models for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 415–422.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110.

Matthews, I., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815.

Murphy-Chutorian, E. and Trivedi, M. (2009). Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626.

Ross, D., Lim, J., Lin, R., and Yang, M. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141.

Shi, J. and Tomasi, C. (1994). Good features to track. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600.

Sivic, J. and Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4):13–36.

Zimmermann, K., Svoboda, T., and Matas, J. (2009). Anytime learning for the NoSLLiP tracker. *Image and Vision Computing, Special Issue: Perception Action Learning*, 27(11):1695–1701.