Robust Online Visual Tracking with a Single Convolutional Neural Network

Hanxi Li[3,1], Yi Li[1,2], and Fatih Porikli[1,2]

[1] Canberra Research Laboratory, NICTA, Australia
[2] Research School of Engineering, Australian National University, Australia
[3] School of Computer and Information Engineering, Jiangxi Normal University, China

Abstract. Deep neural networks, albeit their great success on feature learning in various computer vision tasks, are usually considered as impractical for online visual tracking because they require very long training time and a large number of training samples. In this work, we present an efficient and very robust online tracking algorithm using a single Convolutional Neural Network (CNN) for learning effective feature representations of the target object over time. Our contributions are multifold: First, we introduce a novel truncated structural loss function that maintains as many training samples as possible and reduces the risk of tracking error accumulation, thus drift, by accommodating the uncertainty of the model output. Second, we enhance the ordinary Stochastic Gradient Descent approach in CNN training with a temporal selection mechanism, which generates positive and negative samples within different time periods. Finally, we propose to update the CNN model in a "lazy" style to speed-up the training stage, where the network is updated only when a significant appearance change occurs on the object, without sacrificing tracking accuracy. The CNN tracker outperforms all compared state-of-the-art methods in our extensive evaluations that involve 18 well-known benchmark video sequences.

1 Introduction

Image features play a crucial role in many challenging computer vision tasks such as object recognition and detection. Unfortunately, in many *online* visual trackers features are manually defined and combined [1–4]. Even though these methods report satisfactory results on individual datasets, hand-crafted feature representations would limit the performance of tracking. For instance, normalized cross correlation, which would be discriminative when the lighting condition is favourable, might become ineffective when the object moves under shadow. This necessitates good representation learning mechanisms for visual tracking that are capable of capturing the appearance effectively changes over time.

Recently, deep neural networks have gained significant attention thanks to their success on learning feature representations. Different from the traditional

hand-crafted features [5–7], a multi-layer neural network architecture can efficiently capture sophisticated hierarchies describing the raw data [8]. In particular, the Convolutional Neural Networks (CNN) has shown superior performance on standard object recognition tasks [9–11], which effectively learn complicated mappings while utilizing minimal domain knowledge.

However, the immediate adoption of CNN for online visual tracking is not straightforward. First of all, CNN requires a large number of training samples, which is often not be available in visual tracking as there exist only a few number of reliable positive instances extracted from the initial frames. Moreover, CNN tends to easily overfit to the most recent observation, *e.g.*, most recent instance dominating the model, which may result in drift problem. Besides, CNN training is computationally intensive for online visual tracking. Due to these difficulties, CNN has been treated only as an offline feature extraction step on predefined datasets [12, 13] for tracking applications so far.

In this work, we propose a novel tracking algorithm using CNN to automatically learn the most useful feature representations of particular target objects while overcoming the above challenges. We employ a tracking-by-detection strategy – a three-layer CNN model to distinguish the target object from its surrounding background. We update this CNN model in an online manner. Our CNN generates scores for all possible hypotheses of the object locations (object states) in a given frame. The hypothesis with the highest score is then selected as the prediction of the object state in the current frame.

Typically, tracking-by-detection approaches rely on predefined heuristics to sample from the estimated object location to construct a set of positive and negative samples. Often these samples have binary labels, which leads to a few positive samples and a large negative training set. As a result, the model deterioration in case of a slight inaccuracy during tracking might happen [4]. Besides, the object locations, except the one on the first frame, is not always reliable as they are estimated by the visual tracker and the uncertainty is unavoidable [14]. To address these two issues, our CNN model employs a special type of loss function that consists of a robust term, a structural term, and a truncated norm. The structural term makes it possible to obtain a large number of training samples that have different significance levels considering the uncertainty of the object location at the same time. The robust term enables considering multiple object location estimates during the tracking process rather than being confined into the single, best location estimates at each frame. The truncated norm is applied on the CNN response to reduce the number of samples in the back-propagation [9, 10] stage to significantly accelerate the training process.

We employ the Stochastic Gradient Decent (SGD) method to optimize the parameters in the CNN model. Since the standard SGD algorithm is not tailored for online visual tracking, we propose the following two modifications. First, to prevent the CNN model from overfitting to occasionally detected false positive instances, we introduce a *temporal sampling mechanism* to the batch generation in the SGD algorithm. This temporal sampling mechanism assumes that the object patches shall stay longer than those of the background in the memory.

Therefore, we store all the observed image patches into training sample pool, and we choose the positive samples from a temporal range longer than the negative ones. In practice, we found this is a key factor in the robust CNN-based tracker, because discriminative sampling strategy successfully regularizes the training for effective appearance model.

Second, we use multiple image *cues* (low-level image features, such as normalized gray-scale image and image gradient) as independent channels as network input. We update the CNN parameters by iteratively training each channel independently followed by a joint training using their fully-connected layers. This makes the training efficient and empirically we observed that this two-stage iterative procedure is more accurate than jointly training for all cues.

Finally, in order to increase the tracking speed of the CNN-based tracker to a practical level, we propose to update the CNN model in a "lazy" style. The intuition behind this lazy updating strategy is that we assume that the object appearance is more consistent over the video, compared with the background appearances. The CNN-model is only updated when a significant appearance change occurs on the object. In practice, this lazy updating strategy increases the tracking speed significantly without causing any observable accuracy loss.

To summarize, our main contributions include:

- A visual tracker based on online adapting CNN is proposed. As far as we are aware, this is the first time a single CNN is introduced for learning the best features for object tracking in an online manner.
- A robust, structural, and truncated loss function is exploited for the online CNN tracker. This enables us to achieve very reliable (best reported results in the literature) and robust tracking while achieving tracking speeds up to 2.2 fps.
- An iterative SGD method with a temporal sampling mechanism is introduced for competently capturing object appearance changes.

Our experiments on an extensive dataset of 18 videos from recent benchmarks demonstrate that our method outperforms 9 state-of-the-art algorithms and rarely loses the track of the objects. In addition, it achieves a practical tracking speed (from 0.8fps to 2.2fps depending on the sequence and settings), which is comparable to many other visual trackers.

2 CNN Architecture

2.1 CNN with multiple image cues

Our CNN consists of two convolutional layers, corresponding sigmoid functions as activation neurons and average pooling operators. The dark gray block in Fig. 1 shows the structure of our network, which can be expressed as $(32 \times 32) - (10 \times 10 \times 6) - (1 \times 1 \times 12) - (2)$ in conventional neural network notation.

The input is locally normalized 32×32 image patches, which draws a balance between the representation power and computational load. The first convolution

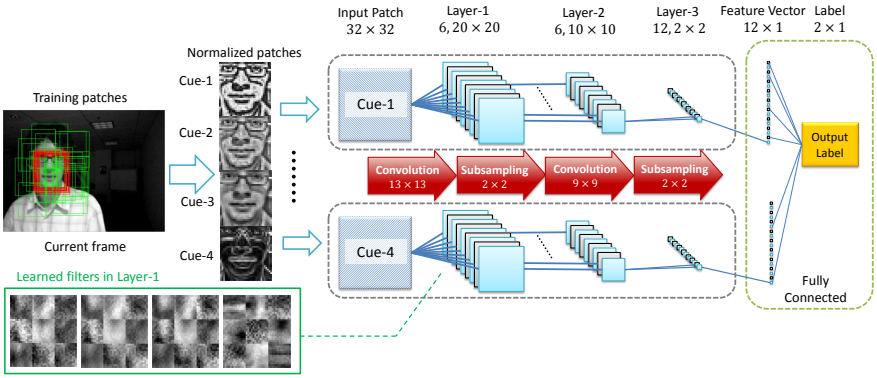


Fig. 1. The architecture of our CNN tracker with multiple image cues.

layer contains 6 kernels each of size 13×13 (an empirical trade-off between overfitting due to a very large number of kernels and discrimination power), followed by a pooling operation that reduces the obtained feature map (filter response) to a lower dimension. The second layer contains 72 kernels with size 9×9. This leads to a 12 dimensional feature vector in the second layer, after the pooling operation in this layer.

The fully connected layer is a logistic regression operation. It takes the 12D vector computed by the first two layers and generates the score vector $\mathbf{s} = [s_1, s_2]^{\mathrm{T}} \in \mathcal{R}^2$, with s_1 and s_2 corresponding to the positive score and negative score, respectively. In order to increase the margin between the scores of the positive and negative samples, we calculate the CNN score of the patch n as

$$f(\mathbf{x}_n; \Omega) = S_n = s_1 \cdot \exp(s_1 - s_2), \tag{1}$$

where $\mathbf{x}_n$ denotes the input and the CNN is parameterized by the weights Ω.

Effective object tracking requires multiple cues, which may include color, image gradients and different pixel-wise filter responses. These cues are weakly correlated yet contain complementary information. Local contrast normalized cues are previously shown [10] to produce accurate object detection and recognition results within the CNN frameworks. The normalization not only alleviates the saturation problem but also makes the CNN robust to illumination change, which is desired during the tracking. In this work, we use 4 image cues generated from the given gray-scale image, *i.e.*, three locally normalized images with different parameter configurations [4] and a gradient image. We let CNN to select the most informative cues in a data driven fashion. By concatenating the final responses of these 4 cues, we build a fully connected layer to the binary

[4] Two parameters r_μ and r_σ determine a local contrast normalization process. In this work, we use three configurations, *i.e.*, $\{r_\mu = 3, r_\sigma = 1\}$, $\{r_\mu = 3, r_\sigma = 3\}$ and $\{r_\mu = 5, r_\sigma = 5\}$, respectively.

output vector (the green dashed block in Fig. 1). Note that we can find similar architectures in the literature [15–17].

2.2 Robust, structural, truncated loss function

Structural loss: Let $\mathbf{x}_n$ and $\mathbf{l}_n \in \{[0,1]^{\mathrm{T}}, [1,0]^{\mathrm{T}}\}$ denote the cue of the input patch and its ground truth label (background or foreground[5]) respectively, and $f(\mathbf{x}_n; \Omega)$ be the predicted score of $\mathbf{x}_n$ with network weights Ω, the objective function of N samples in the batch is

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \|f(\mathbf{x}_n; \Omega) - \mathbf{l}_n\|_2 \tag{2}$$

when the CNN is trained in the batch-mode. Equation 2 is a commonly used loss function and performs well in binary classification problems. However, for object localization tasks, usually higher performance can be obtained by 'structurizing' the binary classifier. The advantage of employing the structural loss is the larger number of available training samples, which is crucial to the CNN training. In the ordinary binary-classification setting, one can only use the training samples with high confidences to avoid class ambiguity. In contrast, the structural CNN is learned based upon all the sampled patches.

We modify the original CNN's output to $f(\phi\langle \Gamma, \mathbf{y}_n\rangle; \Omega) \in \mathbb{R}$, where Γ is the current frame, $\mathbf{y}_n \in \mathbb{R}^o$ is the motion parameter vector of the target object, which determines the object's location in Γ and o is the freedom degree[6] of the transformation. The operation $\phi\langle \Gamma, \mathbf{y}_n\rangle$ suffices to crop the features from Γ using the motion $\mathbf{y}_n$. The associated structural loss is defined as

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \left[\Delta(\mathbf{y}_n, \mathbf{y}^*) \cdot \|f(\phi\langle \Gamma, \mathbf{y}_n\rangle; \Omega) - \mathbf{l}_n\|_2\right], \tag{3}$$

where $\mathbf{y}^*$ is the (estimated) motion state of the target object in the current frame. To define $\Delta(\mathbf{y}_n, \mathbf{y}^*)$ we first calculate the overlapping score $\Theta(\mathbf{y}_n, \mathbf{y}^*)$ [18] as

$$\Theta(\mathbf{y}_n, \mathbf{y}^*) = \frac{\mathrm{area}(r(\mathbf{y}_n) \bigcap r(\mathbf{y}^*))}{\mathrm{area}(r(\mathbf{y}_n) \bigcup r(\mathbf{y}^*))} \tag{4}$$

where $r(\mathbf{y})$ is the region defined by $\mathbf{y}$, $\bigcap$ and $\bigcup$ denotes the intersection and union operations respectively. Finally we have

$$\Delta(\mathbf{y}_n, \mathbf{y}^*) = \left| \frac{2}{1 + \exp(-(\Theta(\mathbf{y}_n, \mathbf{y}^*) - 0.5))} - 1 \right| \in [0, 1]. \tag{5}$$

And the sample label $\mathbf{l}_n$ is set as.

$$\mathbf{l}_n = \begin{cases} [1,0]^{\mathrm{T}} & \text{if} \quad \Theta(\mathbf{y}_n, \mathbf{y}^*) > 0.5 \\ [0,1]^{\mathrm{T}} & \text{elsewise} \end{cases}$$

[5] Here we follow the labeling style in conventional CNN training.
[6] In this paper $o = 3$, i.e., the bounding box changes in its location and the scale.

From Eq. 5 we can see that $\Delta(\mathbf{y}_n, \mathbf{y}^*)$ actually measures the importance of the training patch n. For instance, patches that are very close to object center and reasonably far from it may play more significant roles in training the CNN, while the patches in between are less important.

Structural Loss with a Robust Term and the Truncated Norm In visual tracking, when a new frame $\Gamma_{(t)}$ comes, we predict the object motion state $\mathbf{y}^*_{(t)}$ as

$$\mathbf{y}^*_{(t)} = \arg\max_{\mathbf{y}_n \in \mathcal{Y}} \left(f(\phi\langle \Gamma_{(t)}, \mathbf{y}_n \rangle; \Omega) \right), \tag{6}$$

where $\mathcal{Y}$ contains all the test patches in the current frame. Among all motion states $\mathbf{y}^*_{(t)},\ \forall t = 1, 2, \ldots, T$, only the first one $\mathbf{y}^*_{(1)}$ is always reliable as it is manually defined. Other motion states are estimated based on the previous observations. Thus, the uncertainty of the prediction $\mathbf{y}_{(t)},\ \forall t > 1$ is usually unavoidable. Recall that, the structural loss defined in Equation 4 could change significantly if a minor perturbation is imposed on $\mathbf{y}_{(t)}$, one requires a accurate $\mathbf{y}_{(t)}$ in every frame, which is, unfortunately, not feasible.

The Multiple-Instance-Learning (MIL) based approaches [14, 19] use the instance bags, rather than individual instances as training samples, and the learning goal is to maximize the maximum score in the positive bag while minimize those in the negative bags [14, 19]. Inspired by this idea, we regularize the ordinary structural loss using a set of positive instances rather than only one positive instance, which is novel in the state-of-the-art tracking-by-detection methods. In frame t ($t > 1$), we define the positive instance set as

$$\mathcal{Y}^* = \{\mathbf{y}_j \mid \Theta(\mathbf{y}_j, \mathbf{y}^*) < 0.5,\ f(\phi\langle \Gamma, \mathbf{y}_j \rangle; \Omega) > \eta f(\phi\langle \Gamma, \mathbf{y}^* \rangle; \Omega)\}. \tag{7}$$

It is easy to see that this set contains all the test samples with high scores and far away from the prediction $\mathbf{y}^*$. Here, we set $\eta = 0.975$, which is high enough to eliminate most of the true negatives. Then, we define a robust term based on $\mathbf{y}_n$ and $\mathcal{Y}^*$ as

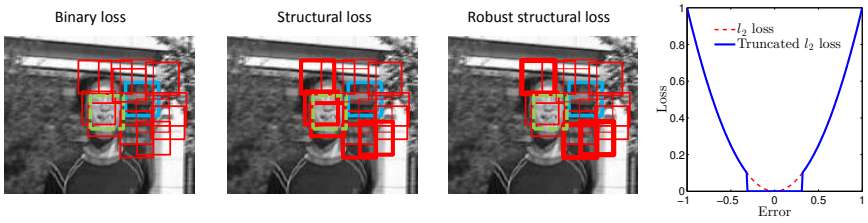
$$r(\mathbf{y}_n, \mathcal{Y}^*) = \max\left(0,\ \max_{\mathbf{y}_j \in \mathcal{Y}^*} \frac{2}{1 + \exp(-(\Theta(\mathbf{y}_n, \mathbf{y}_j) - 0.5))} - 1\right). \tag{8}$$

We obtain the structural loss regularized by the positive set $\mathcal{Y}^*$ as

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left[(\Delta(\mathbf{y}_n, \mathbf{y}^*) - r(\mathbf{y}_n, \mathcal{Y}^*)) \cdot \| f(\phi\langle \Gamma, \mathbf{y}_n \rangle; \Omega) - \mathbf{l}_n \|_2 \right], \tag{9}$$

We can treat the above structural loss function as a robust version of the loss defined in Equation 3. The weights of the training samples which have high CNN score but far away from the prediction $\mathbf{y}^*$ will be reduced significantly. In practice, we observed this structural loss reduces error accumulation, which usually starts from an incorrectly predicted $\mathbf{y}^*$.

The effect of proposed robust structural loss is shown in Figure 2a, comparing with other two conventional loss functions, *i.e.*, the binary loss and the normal



(a) (b)

Fig. 2. (a) Illustration of the effects of different loss functions on negative samples. The green block stands for the object location while the blue block is the prediction of the visual tracker. The red blocks are the negative samples labeled according to the prediction (which is incorrect). The thickness of the red blocks represent their importance scores in the following training procedure. (b) Truncated l_2 norm.

structural loss. We can see that, for a binary loss function, all the negative blocks (in red) share the same training weight. For the normal structural loss, those negative patches that overlap with the prediction (blue block) are assigned smaller weights than those far away from the prediction. As a result, the real object (green block) will be treated as a negative sample with a high importance, which might confuse the classifier. In contrast, the robust structural loss will reduce the weight around the green block as the prediction score for green block is also high, which is, in practice, usually true. Consequently, the incorrectly labeled object block will not play a dominant role in the consecutive training stages, and thus the learned tracker achieve more robustness.

Finally, we speed up the CNN training by employing a truncated l_2-norm in our model. We empirically observe that patches with very small error does not contribute much in the back propagation. Therefore, we can approximate the loss by counting the patches with errors that are larger than a threshold. Motived by this, we define a truncated l_2 norm as

$$\|e\|_{\mathbb{T}} = \|e\|_2 \cdot (1 - \mathbb{1}[\|e\|_2 \leq \beta]) \,, \tag{10}$$

where $\mathbb{1}[\cdot]$ denotes the indicator function while e is the prediction error, *e.g.*, $f(\phi\langle \mathit{\Gamma}, \mathbf{y}_n\rangle; \mathit{\Omega}) - \mathbf{l}_n$ for patch-n. This truncated norm is visualized in Fig. 2b and now Eq. 9 becomes:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left[(\mathit{\Delta}(\mathbf{y}_n, \mathbf{y}^*) - r(\mathbf{y}_n, \mathcal{Y}^*)) \cdot \|f(\phi\langle \mathit{\Gamma}, \mathbf{y}_n\rangle; \mathit{\Omega}) - \mathbf{l}_n\|_{\mathbb{T}} \right] , \tag{11}$$

It is easy to see that with the truncated norm $\|\cdot\|_{\mathbb{T}}$, the backpropagation [9] process only depends on the training samples with large errors, *i.e.*, $\|f(\phi\langle \mathit{\Gamma}, \mathbf{y}_n\rangle; \mathit{\Omega}) - \mathbf{l}_n\|_{\mathbb{T}} > 0$. Accordingly, we can ignore the samples with small errors and the backpropagation procedure is significantly accelerated. In this work, we use $\beta = 0.03$.

3 Optimization of CNN for Tracking

3.1 Online Learning: Iterative SGD with Temporal Sampling

Temporal Sampling Following other CNN-based approaches [9, 10], we used Stochastic Gradient Decent (SGD) for the learning of the parameters Ω. However, the SGD we employ is specifically tailored for visual tracking.

Different from detection and recognition tasks, the training sample pool grows gradually as new frames come in visual tracking. Moreover, it is desired to learn a consistent object model over *all* the previous frames and then use it to distinguish the object from the background in the *current* frame. This implies that we can effectively learn a discriminative model on a long-term positive set and a short-term negative set.

Based on this intuition, we tailor the SGD method by embedding in a temporal sampling process. In particular, given that the positive sample pool is $\mathbb{Y}^+_{1:t} = \{\mathbf{y}^+_{1,(1)}, \mathbf{y}^+_{2,(1)}, \cdots, \mathbf{y}^+_{N-1,(t)}, \mathbf{y}^+_{N,(t)}\}$ and the negative sample pool is $\mathbb{Y}^-_{1:t} = \{\mathbf{y}^-_{1,(1)}, \mathbf{y}^-_{2,(1)}, \cdots, \mathbf{y}^-_{N-1,(t)}, \mathbf{y}^-_{N,(t)}\}$, when generating a mini-batch for SGD, we sample the positive pool with the probability

$$\text{Prob}(\mathbf{y}^+_{n,(t')}) = \frac{1}{tN}, \tag{12}$$

while sample the negative samples with the probability

$$\text{Prob}(\mathbf{y}^-_{n,(t')}) = \frac{1}{Z}\exp\left[-\sigma(t-t')^2\right], \tag{13}$$

where $\frac{1}{Z}$ is the normalization term and we use $\sigma = 10$ in this work.

In a way, the above temporal selection mechanism can be considered to be similar to the "multiple-lifespan" data sampling [20]. However, [20] builds three different codebooks, each corresponding to a different lifespan, while we learn one discriminative model based on two different sampling distributions.

Iterative Stochastic Gradient Descent (IT-SDG) Recall that we use multiple image cues as the input of the CNN tracker. This leads to a CNN with higher complexity, which implies a low training speed and a high possibility of overfitting. By noticing that each image cue may be weakly independent, we train the network in a iterative manner. In particular, we define the model parameters as $\Omega = \{\mathbf{w}^1_{cov}, \cdots, \mathbf{w}^K_{cov}, \mathbf{w}^1_{fc}, \cdots, \mathbf{w}^K_{fc}\}$, where $\mathbf{w}^k_{cov}$ denotes the filter parameters in cue-k while $\mathbf{w}^k_{fc}$ corresponds to the fully-connected layer. After we complete the training on $\mathbf{w}^k_{cov}$, we evaluate the filter responses from all the cues in the fully-connected layer and then jointly update $\{\mathbf{w}^1_{fc}, \cdots, \mathbf{w}^K_{fc}\}$ with a small learning rate (see Algorithm 1). This can be regarded as a coordinate-descent variation of SGD. In practice, we found out both the temporal sampling mechanism and the IT-SDG significantly curb the overfitting problem.

Algorithm 1 Iterative SGD with temporal sampling

Inputs: Frame image $\mathit{\Gamma}_{(t)}$; Two sample pools $\mathbb{Y}^+_{1:t}$, $\mathbb{Y}^-_{1:t}$;
CNN model (K cues) $f(\phi\langle\mathit{\Gamma}_{(t)},\cdot\rangle;\Omega=\{\mathbf{w}^1_{cov},\cdots,\mathbf{w}^K_{cov},\mathbf{w}^1_{fc},\cdots,\mathbf{w}^K_{fc}\})$.
Estimated/given $\mathbf{y}^*$;
Learning rates $\hat{r}=\frac{r}{K}$; minimal loss ε; training step budget $M \gg K$.
procedure IT-SGD($\mathbb{Y}^+_{1:t}$, $\mathbb{Y}^-_{1:t}$, f, $\mathbf{y}^*$, $\hat{r}$, r, M)
  Sample samples from $\mathbb{Y}^+_{1:t}$ and $\mathbb{Y}^-_{1:t}$, according to 12 and 13.
  Save the selected samples in $\mathbb{Y}=\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_N\}$ with labels $\mathbf{l}_1,\mathbf{l}_2,\cdots,\mathbf{l}_N$.
  for $m \leftarrow 1,\ M-1$ **do**
    Calculate loss $\mathcal{L}_m = \frac{1}{N}\sum_{n=1}^{N}\left[(\mathit{\Delta}(\mathbf{y}_n,\mathbf{y}^*)-r(\mathbf{y}_n,\mathcal{Y}^*))\cdot\left\|f_m(\phi\langle\mathit{\Gamma}_{(t)},\mathbf{y}_n\rangle;\Omega)-\mathbf{l}_n\right\|_{\mathbb{T}}\right]$
    If $\mathcal{L}\le\varepsilon$, **break**;
    $k=\mathrm{mod}(m,K)+1$;
    Update $\mathbf{w}^k_{cov}$ using SGD with learning rate r for f_m;
    Jointly update $\{\mathbf{w}^1_{fc},\cdots,\mathbf{w}^K_{fc}\}$ for f_m, with step length $\hat{r}$;
    Save $f_{m+1}=f_m$;
  end for
end procedure
Outputs: New CNN model $f^*=f_{m^*}, m^*=\mathrm{argmax}_m\,\mathcal{L}_m$.

3.2 Lazy Update and the Overall Work Flow

It is straightforward to updating the CNN model using the IT-SGD algorithm at each frame. However, this could be computationally expensive as the complexity of training processes would dominate the complexity of the whole algorithm. On the other hand, in case the appearance of the object is not always changing, a well-learned appearance model can remain discriminant for a long time.

Motivated by this, we propose to update the CNN model in a lazy manner. When tracking the object, we only update the CNN model when the training loss $\mathcal{L}_1$ is above 2ε. Once the training start, the training goal is to reduce $\mathcal{L}$ below ε. As a result, usually $\mathcal{L}_1 < 2\varepsilon$ holds in a number of the following frames, and thus no training is required for those frames. This way, we accelerate the tracking algorithm significantly (Figure 3).

4 Experiments

Video sequences and algorithms compared We evaluate our method on 18 benchmark video sequences that cover most challenging tracking scenarios such as scale changes, illumination changes, occlusions, cluttered backgrounds and fast motion. The first frames of these sequences are shown in Figure 4.

We compare our method with 5 other state-of-the-art trackers including TLD [21], CXT [22], ASLA [23], Struck [4], SCM [24], and 4 classical tracking algorithms, *e.g.*, Frag [3], CPF [1], IVT [25] and MIL [14].

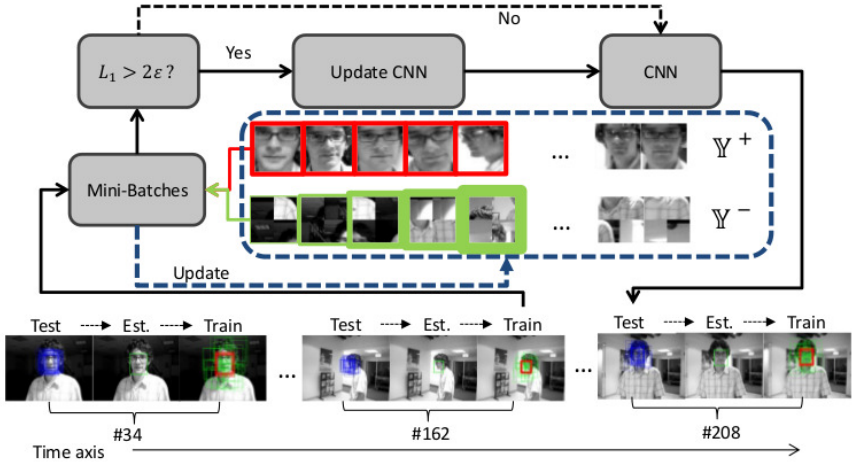


Fig. 3. Work flow of proposed algorithm. The bottom row shows the three-stages operations on a frame: test, estimation and training. In the training frames, the green bounding-boxes are the negative samples while the red ones denote the positive samples. The dashed block covers the positive sample pool $\mathbb{Y}^+$ (red) and negative sample pool $\mathbb{Y}^-$ (green). In each pool, the edges of the sample patches indicate their sampling importances. The thicker the edge, the more possible it will be selected for training.

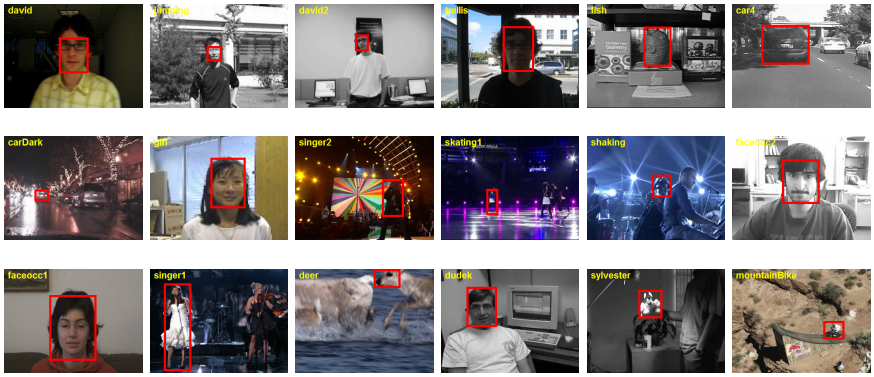


Fig. 4. The first frames of the selected video sequences. From top left to bottom right: David, Jumping, David2, Trellis, Fish, Car4, CarDark, Girl, Singer2, Skating1, Shaking, FaceOcc2, FaceOcc1, Singer1, Deer, Dudek, Sylvester, MountainBike. The red blocks are the initialization in the first frame.

Performance measurements The tracking results are evaluated via the following two measurements: 1) Tracking Precision (TP), the percentage of the frames whose estimated location is within the given distance-threshold (τ_d) to the ground truth, and 2) Tracking Success Rate (TSR), the percentage of the frames in which the overlapping score defined in Equation 4 between the estimated location and the ground truth is larger than a given overlapping-threshold (τ_o). For a fair comparison, we run our algorithm for 5 times and then report

the average TP/TSR scores. The results of other visual trackers are obtained from [26]. We run our algorithm in Matlab with an unoptimized code mixed with CUDA-PTX kernels for the CNN implementation. The hardware environment includes one quad-core CPU and a NVIDIA GTX580.

Parameter setting Most parameters of the CNN tracker are given in Sec. 2 and Sec. 3. In addition, there are some motion parameters for sampling the image patches. In this work, we only consider the displacement Δ_x, Δ_y and the relative scale s of the object[7]. In a new frame, we sample 1500 random patches in a Gaussian Distribution which centers on the previous predicted state. The standard deviation for the three dimensions are 10, 10 and 0.02, respectively. Note that, all parameters are fixed for all videos for most objective evaluation; no parameter tuning is performed for any specific video sequence.

Main comparison results Firstly, we evaluate all algorithms using fixed thresholds, i.e. $\tau_d = 15$, $\tau_o = 0.5$, which is a common setting in tracking evaluations. Results are given in Table 1. Specifically, the score of TP and TSR are shown in each table block. The average performance is also reported for each tracker.

We can see that, our method achieves much better overall average results compared with other trackers. The performance gap between our method and the reported best result in the literature are 9% for the TP measure: our method achieves 83% accuracy while the best state-of-the-art is 74% (SCM method). For the TSR measure, our method is 7% better than the existing methods: our method gives 83% accuracy while the best state-of-the-art is 76% (SCM method). Furthermore, our CNN tracker have ranked as the best method for 17 times. These numbers for Struck, ASLA and SCM are 16, 13, 9, respectively. Another observation from the Table 1 is that, our method rarely performs inaccurately; 89% of the time our score is within the top scores (no less then 80% of the highest score for one sequence). As visible, our tracker is robust to dramatic appearance changes, e.g. due to motion blurs (Jumping and Deer) or illumination variations (Fish, Trellis and Singer2).

In fact, the superiority of our method becomes more clear when the experiments with different measurement criteria (different τ_d, τ_o) are conducted. In specific, for TP, we evaluate the trackers with the thresholds $\tau_d = 1, 2, \cdots, 50$ while for TSR, we use the thresholds $\tau_o = 0$ to 1 at the step of 0.05. According to the scores under different criteria, we generate the precision curves and the success-rate curves for each tracking method, which is shown in Figure 5.

From the score plots we can see that, overall the CNN tracker ranks the first (red curves) for both TP and TSR evaluations. Our algorithm is very robust when $\tau_o < 0.68$ and $\tau_d > 7$ as it outperform all other trackers. The CNN tracker rarely misses the target completely. Having mentioned that when the overlap thresholds are tight (*e.g.* $\tau_o > 0.8$ or $\tau_d < 5$), our tracker has similar response to rest of the trackers we tested.

[7] $s = h/32$, where h is object's height

	CNN	TLD	CXT	ASLA	Struck	SCM	Frag	CPF	IVT	MIL
david	0.84/0.80	1.00/0.97	1.00/0.83	1.00/0.96	0.32/0.24	1.00/0.91	0.13/0.12	0.11/0.03	0.95/0.79	0.42/0.23
jumping	1.00/1.00	0.98/0.85	0.86/0.29	0.29/0.17	1.00/0.80	0.14/0.12	0.96/0.85	0.14/0.11	0.18/0.10	0.76/0.48
david2	1.00/1.00	1.00/0.95	1.00/1.00	1.00/0.95	1.00/1.00	1.00/0.91	0.31/0.30	1.00/0.46	1.00/0.93	0.69/0.32
trellis	0.99/0.98	0.48/0.47	0.88/0.81	0.86/0.86	0.83/0.78	0.86/0.85	0.35/0.36	0.22/0.17	0.32/0.31	0.17/0.24
faceocc2	0.90/0.92	0.69/0.83	0.98/0.95	0.67/0.81	0.99/1.00	0.74/0.87	0.54/0.75	0.29/0.35	0.91/0.91	0.55/0.94
faceocc1	0.59/1.00	0.04/0.83	0.19/0.77	0.13/0.31	0.24/1.00	0.65/1.00	0.83/1.00	0.18/0.52	0.34/0.98	0.15/0.76
dudek	0.19/0.33	0.50/0.84	0.73/0.92	0.61/0.90	0.78/0.98	0.80/0.98	0.48/0.59	0.45/0.69	0.84/0.97	0.57/0.86
singer1	0.74/0.89	0.98/0.99	0.80/0.32	1.00/1.00	0.56/0.30	1.00/1.00	0.23/0.22	0.94/0.32	0.81/0.48	0.33/0.28
deer	1.00/1.00	0.73/0.73	0.94/0.92	0.03/0.03	1.00/1.00	0.03/0.03	0.18/0.21	0.04/0.04	0.03/0.03	0.08/0.13
fish	1.00/1.00	0.98/0.96	1.00/1.00	1.00/1.00	1.00/1.00	0.84/0.86	0.52/0.55	0.09/0.10	1.00/1.00	0.34/0.39
car4	1.00/0.85	0.86/0.79	0.30/0.30	1.00/1.00	0.97/0.40	0.97/0.97	0.18/0.21	0.03/0.02	1.00/1.00	0.35/0.28
carDark	0.92/0.88	0.61/0.53	0.71/0.69	1.00/1.00	1.00/1.00	1.00/1.00	0.39/0.25	0.11/0.02	0.77/0.70	0.27/0.18
singer2	0.87/0.91	0.04/0.10	0.05/0.04	0.03/0.04	0.03/0.04	0.11/0.16	0.16/0.20	0.08/0.14	0.04/0.04	0.18/0.48
sylvester	0.82/0.58	0.94/0.93	0.76/0.75	0.78/0.75	0.93/0.93	0.89/0.89	0.68/0.68	0.79/0.71	0.68/0.68	0.54/0.55
girl	0.94/0.84	0.87/0.76	0.74/0.64	1.00/0.91	1.00/0.98	1.00/0.88	0.63/0.54	0.69/0.54	0.36/0.19	0.51/0.29
mountainbike	0.55/0.68	0.25/0.26	0.28/0.28	0.78/0.86	0.86/0.90	0.86/0.96	0.13/0.14	0.08/0.15	0.87/0.98	0.52/0.57
shaking	0.79/0.93	0.33/0.40	0.05/0.11	0.25/0.38	0.08/0.17	0.70/0.90	0.07/0.07	0.14/0.12	0.01/0.01	0.18/0.23
skating1	0.77/0.34	0.26/0.23	0.20/0.12	0.76/0.69	0.29/0.37	0.72/0.42	0.14/0.12	0.20/0.19	0.08/0.07	0.12/0.10
Overall	**0.83/0.83**	**0.64/0.69**	**0.64/0.60**	**0.68/0.70**	**0.71/0.71**	**0.74/0.76**	**0.38/0.40**	**0.31/0.26**	**0.57/0.56**	**0.37/0.41**

Table 1. The tracking scores of the proposed method and other visual trackers. The reported results are shown in the order of "TP/TSR". The top scores are shown in red for each row. For CNN tracker, a score is shown in blue if it is higher than 80% of the highest value in that row.

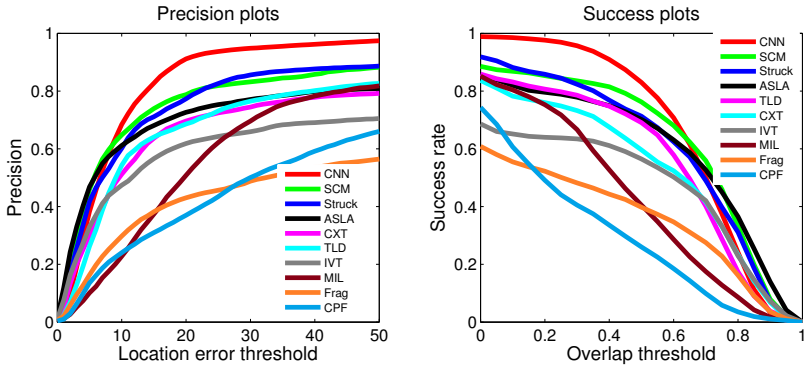


Fig. 5. The Precision Plot (left) and the Success Plot (right). The color of one curve is determined by the rank of the corresponding trackers, not their names.

In many applications, it is more important to not to loose the target object than very accurately locate its bounding box. As visible, our tracker rarely looses the object. Usually the object is much smaller than the frame and there is no big difference between 68% overlapping and 90% for users in this scenario.

Verification for loss function and the temporal sampling Here we verify the two proposed modifications to the CNN model. We rerun the whole experiment using the CNN tracker without one or both of the modifications. The scores of our CNN tracker with different settings are reported in Table 2.

From the table we can see that, both the robust structural loss and the temporal sampling method contribute the success of our CNN tracker. In particular, the temporal sampling plays a more important role and the robust structural

	david	*Jumping*	*david2*	*trellis*	*faceOcc2*	*faceocc1*	*dudek*	*singler1*	*deer*	
CNN-std	0.67/0.64	0.05/0.05	0.27/0.26	0.76/0.79	0.93/0.95	0.46/0.88	0.01/0.21	0.95/0.91	0.95/1.00	
CNN-NoTemp	0.65/0.62	0.29/0.28	0.65/0.63	0.78/0.79	0.79/0.72	0.35/0.81	0.06/0.22	0.83/0.97	0.99/1.00	
CNN-NoStruct	0.88/0.69	0.68/0.68	1.00/0.99	0.97/0.96	0.79/0.72	0.40/0.99	0.12/0.27	0.68/0.81	1.00/1.00	
CNN-ours	0.84/0.80	1.00/1.00	1.00/1.00	0.99/0.98	0.90/0.92	0.59/1.00	0.19/0.33	0.74/0.89	1.00/1.00	
	fish	*car4*	*carDark*	*singer2*	*sylvester*	*girl*	*mountainbike*	*shaking*	*sylvester*	**overall**
CNN-std	1.00/1.00	0.15/0.18	1.00/0.99	0.54/0.60	0.54/0.44	0.08/0.08	0.93/0.92	0.43/0.50	0.61/0.09	**0.57/0.58**
CNN-NoTemp	0.97/1.00	0.85/0.66	0.88/0.67	0.54/0.59	0.67/0.50	0.52/0.35	0.85/0.81	0.22/0.32	0.39/0.08	**0.63/0.61**
CNN-NoStruct	1.00/1.00	1.00/0.87	1.00/1.00	0.83/0.90	0.86/0.75	0.85/0.63	0.59/0.82	0.75/0.84	0.54/0.28	**0.77/0.79**
CNN-ours	1.00/1.00	0.99/0.85	0.92/0.88	0.87/0.91	0.82/0.58	0.94/0.84	0.55/0.68	0.79/0.93	0.77/0.34	**0.83/0.83**

Table 2. The performance comparison between CNN tracker variations.

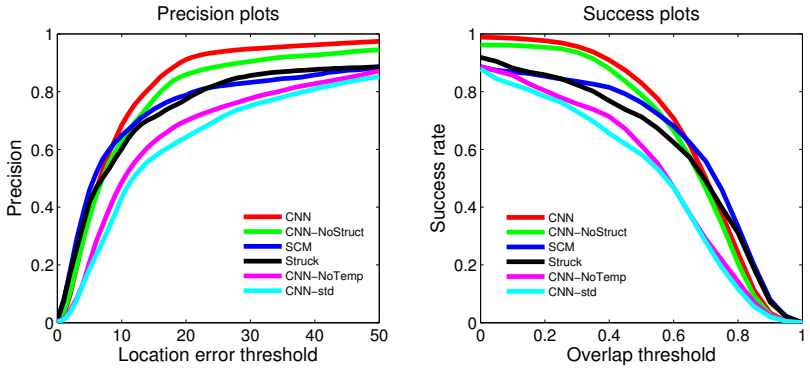


Fig. 6. The Precision Plot (left) and the Success Plot (right). The color of one curve is determined by the rank of the corresponding trackers, not their names.

loss further increase the accuracy by 5%. Similar to the previous evaluation, we plot the precision-curves and the success-rate curves for the CNN tracker with different settings. The curves consistently go up when the components are added into the CNN model. That indicates the validity of the propose modifications.

Some tracking examples In Figure 7 we show the tracking results of our CNN tracker comparing with three state-of-the-art trackers (SCM, Struck and ASLA) and three classical tracking methods (IVT, MIL, CPF), on 9 video sequences. Here we show some good results obtained by using our algorithm (row 1 to row 7) and some sequences on which the CNN tracker is outperformed by the other trackers (the last two rows). We can see that, even for the "failure" cases, the CNN tracker does not lose the target while is only "kidnapped" by some local part of the object (the human for the *mountainbike*).

5 Conclusion

We introduced a CNN based online object tracker. We employed a CNN architecture and a robust structural loss function that handles multiple input cues. We also proposed to modify the ordinary Stochastic Gradient Descent for visual tracking by iteratively update the parameters and add a temporal sampling

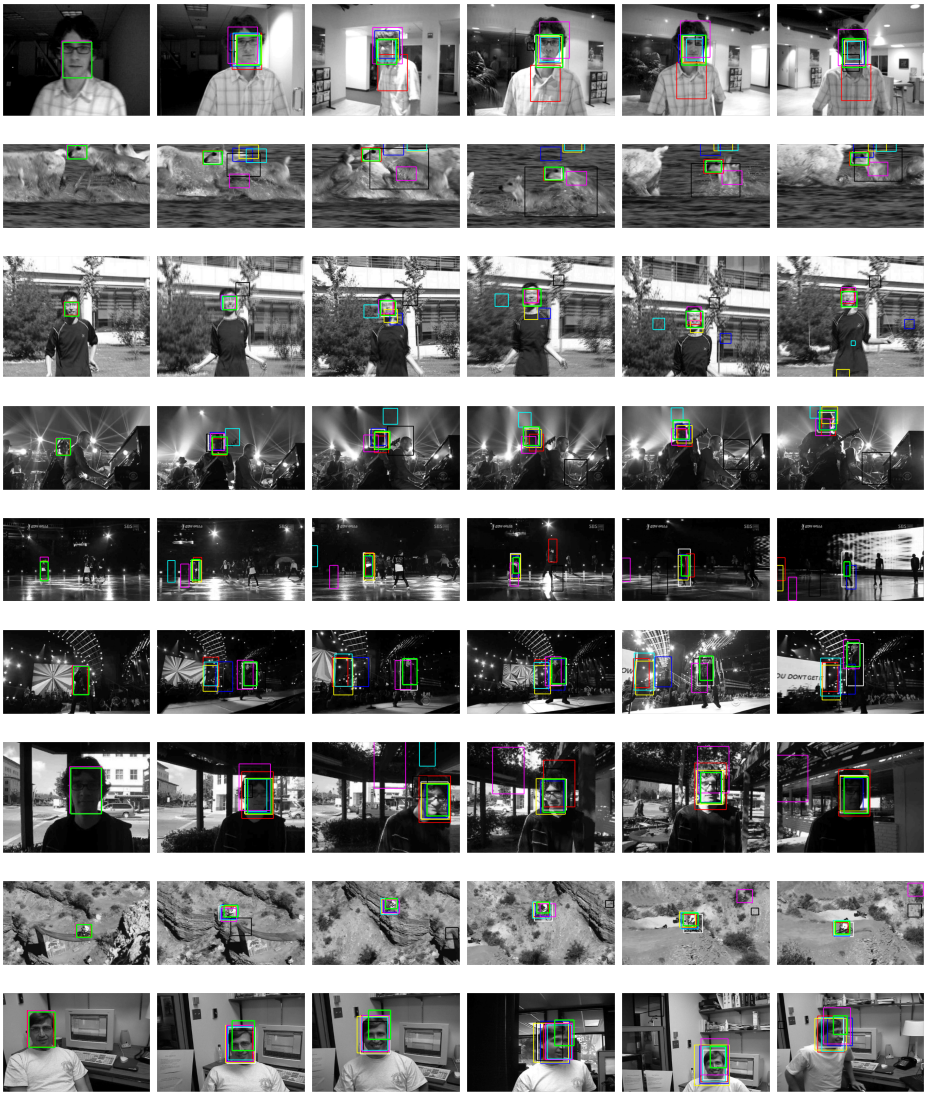


Fig. 7. Tracking results of the CNN tracker compared with other 6 visual trackers. The tracker are shown in different colors: green–CNN tracker; red–Struck; blue–SCM; black–CPF; yellow–ASLA; cyan–IVT; magenta–MIL; white–ground-truth. In each row, the 6 frames roughly span the whole sequence.

mechanism in the mini-batch generation. This tracking-tailored SGD algorithm increase the speed and the robustness of the training process significantly. Our experiments demonstrate that the CNN-based tracking algorithm performs very well on 18 benchmark sequences and achieves the comparable tracking speed to some state-of-the-art trackers.

References

1. Pérez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. (In: ECCV 2002)
2. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005) 1631–1643
3. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR 2006. (Volume 1.)
4. Hare, S., Saffari, A., Torr, P.H.: Struck: Structured output tracking with kernels. In: ICCV 2011, (IEEE) 263–270
5. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60** (2004) 91–110
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1., IEEE (2005) 886–893
7. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on **28** (2006) 2037–2041
8. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35** (2013) 1798–1828
9. Kavukcuoglu, K., Sermanet, P., Boureau, Y.L., Gregor, K., Mathieu, M., LeCun, Y.: Learning convolutional feature hierachies for visual recognition. (In: NIPS 2010)
10. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. (In: NIPS 2012)
11. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. (In: CVPR 2012)
12. Fan, J., Xu, W., Wu, Y., Gong, Y.: Human tracking using convolutional neural networks. Trans. Neur. Netw. **21** (2010) 1610–1623
13. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. (In: NIPS 2013)
14. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. Transactions on Pattern Analysis and Machine Intelligence (2011)
15. Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: Web-Age Information Management. Springer (2014) 298–310
16. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 3642–3649
17. Cireşan, D., Meier, U., Masci, J., Schmidhuber, J.: Multi-column deep neural network for traffic sign classification. Neural Networks **32** (2012) 333–338
18. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Intl J. of Comp. Vis. **88** (2010) 303–338
19. Viola, P., Platt, J., Zhang, C., et al.: Multiple instance boosting for object detection. In: NIPS. Volume 2. (2005) 5
20. Xing, J., Gao, J., Li, B., Hu, W., Yan, S.: Robust object tracking with online multi-lifespan dictionary learning. In: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE (2013) 665–672

21. Kalal, Z., Matas, J., Mikolajczyk, K.: Pn learning: Bootstrapping binary classifiers by structural constraints. In: CVPR 2010, (IEEE) 49–56
22. Dinh, T.B., Vo, N., Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: CVPR 2011, (IEEE) 1177–1184
23. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: CVPR 2012, (IEEE) 1822–1829
24. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparsity-based collaborative model. In: CVPR 2012, (IEEE) 1838–1845
25. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Intl. J. Comp. Vis **77** (2008) 125–141
26. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. (CVPR 2013)