MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization

Jianming Zhang, Shugao Ma, and Stan Sclaroff

Department of Computer Science, Boston University, USA
{jmzhang,shugaoma,sclaroff}@bu.edu

Abstract. We propose a multi-expert restoration scheme to address the model drift problem in online tracking. In the proposed scheme, a tracker and its historical snapshots constitute an expert ensemble, where the best expert is selected to restore the current tracker when needed based on a minimum entropy criterion, so as to correct undesirable model updates. The base tracker in our formulation exploits an online SVM on a budget algorithm and an explicit feature mapping method for efficient model update and inference. In experiments, our tracking method achieves substantially better overall performance than 32 trackers on a benchmark dataset of 50 video sequences under various evaluation settings. In addition, in experiments with a newly collected dataset of challenging sequences, we show that the proposed multi-expert restoration scheme significantly improves the robustness of our base tracker, especially in scenarios with frequent occlusions and repetitive appearance variations.

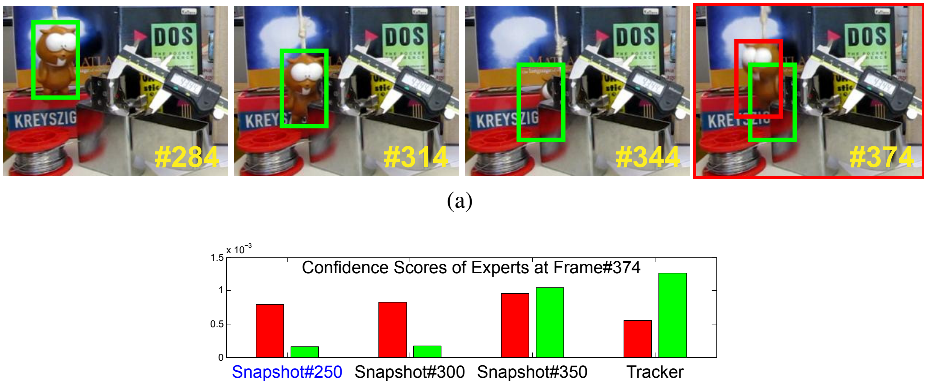
1 Introduction

In this paper, we focus on the problem of model-free online tracking of an object, given only the object's initial position and previous observations, within a tracking-by-detection framework. In many online trackers, an object model is maintained via online updates, which are intended to account for appearance changes of the target. However, the process of updating the model also brings the model drift problem, which is a key challenge in online visual tracking.

Model drift occurs because factors like tracking failure, occlusions and misalignment of training samples can lead to bad model updates. One remedy is to incorporate the first frame template or prior knowledge in the online model update procedure [20,15]. However, relying on a fixed model prior tends to restrict the tracker's ability to handle large object appearance changes. Other trackers [22,32,14] use a "censorship mechanism" where an update is prevented when certain criteria are met (or not met). The detection of good or bad updates usually relies upon smoothness assumptions for motion and appearance changes, which are often violated in challenging scenarios. And once the censorship mechanism fails, these trackers will either miss the chance to evolve or get trapped in a background region, due to the fact that the model can only evolve forward, without a mechanism to correct for past mistakes.

Instead of trying to prevent bad updates from happening, we propose a formulation that can correct the effects of bad updates *after* they happen. For this purpose, we introduce a multi-expert tracking framework, where a discriminative tracker and its former

D. Fleet et al. (Eds.): ECCV 2014, Part VI, LNCS 8694, pp. 188–203, 2014.



(a)

(b)

Fig. 1. In (a), green rectangles show the results of our base tracker. After a period severe occlusion, some tracker snapshots give a different prediction in red at frame #374. The chart in (b) shows the confidence scores of the tracker and its three recent snapshots for the two different predictions at frame #374 in corresponding colors. Our multi-expert mechanism favors the snapshot at frame #250, which is less ambiguous when selecting between the red and green hypotheses, even though the current tracker gives the highest confidence score for the green prediction.

snapshots constitute an expert ensemble, and the best expert is selected based on a minimum loss criterion to restore a tracker when a disagreement among the experts occurs. Traditional loss functions, which measure the discrepancy between the prediction and the true label, are only applicable in supervised settings. To get around this, we propose a novel formulation of the tracking-by-detection problem, so as to naturally introduce an entropy-regularized optimization function [10] as our expert selection criterion.

The key observation motivating our approach is that bad model updates usually contaminate a tracker's appearance model with inconsistent training samples, thus leading to ambiguous inference. An example is shown in Fig. 1. During a period of severe occlusion, the tracker's online updates incorporate the wrong foreground image patch. After the target reappears, although the tracker is still responsive to the true target in red, it starts to over-fit the wrong patch in green, yielding an incorrect prediction. In contrast, our formulation maintains a set of tracker snapshots throughout the tracking process. A past snapshot can be identified to localize the target with less ambiguity. This "low ambiguity" model prior is formulated as an entropy term in our expert selection criterion, which can be used to identify (and correct for) model drift.

To implement the base tracker in our multi-expert framework, we adopt an online SVM algorithm [26] that approximates the offline version by employing compact prototype sets, which summarize the effects of all previous training samples near the decision boundary and thereby avoids hard pruning of training samples, as needed in [31,21,11]. However, in [26] the algorithm is not evaluated in the tracking problem, so we carefully reformulated it to account for specific characteristics of the tracking problem. Furthermore, we use a linear kernel and the feature mapping technique of [19] to efficiently find nonlinear decision boundaries in the original feature space.

The main contribution of this paper is a tracking method based on a novel Multi-Expert Entropy Minimization (MEEM) restoration scheme, which allows a tracker to evolve backwards to undo undesirable model updates. On a standard benchmark dataset of 50 videos [27], our method improves over previous leading methods by more than 15% under various evaluation settings, *e.g.* with random spatial or temporal initialization perturbations. Furthermore, on a newly collected dataset of 10 challenging video sequences, the proposed MEEM restoration scheme is shown to significantly improve the robustness of our base tracker, especially in challenging scenarios with occlusions and repetitive appearance variations.

2 Related Work

The following is a brief review of some closely related works in visual tracking. For more comprehensive literature reviews, readers are directed to [18,28].

Tracking-by-Detection. Many discriminative trackers have been proposed. Avidan [1] utilizes an off-line trained SVM classifier in an optical flow based tracker. In [2,7], weak classifiers are combined and updated by a boosting algorithm for model-free tracking. The formulation of [4] combines weak SVM classifiers via randomized weighting vectors. In [3], multiple instance learning is used to avoid the error-prone, hard-labeling process. Structured SVM is proposed by [11] for tracking. While many previous works focus on designing a robust learning mechanism, our method tries to correct the past mistakes of online learning by allowing the tracker to evolve backward.

Hybrid Multi-Tracker Methods. Some tracking methods maintain a tracker ensemble, so the failure of a single tracker can be compensated by other trackers. For example, hybrid generative-discriminative methods are used in [29,32]. In [24] two SVM classifiers are employed in a co-training framework. Kwon *et al.* [16] integrate decorrelated trackers via an MCMC framework. In [17], multiple trackers from a tracker space are sampled and combined to handle challenging scenarios. Our multi-expert scheme differs in that our expert ensemble is made of a single tracker and its previous snapshots, and only one tracker needs to be updated in our system.

Training Sample Management. When memorizing training samples is needed, some trackers keep a fixed set of recent training samples [24,16], and others dynamically maintain a subset of the previous training samples using heuristics. In [11], support vectors that have the least influence on the current decision plane are discarded. In [31,21], templates with the least importance will be replaced when they cannot well represent the target appearance. In [13], older templates are replaced less frequently, assuming that they can be more accurate. Instead of using such heuristics, our tracker maintains a compact prototype set to summarize the effects of all previous training samples, and thereby avoids hard pruning of training samples.

Drift Prevention. Some approaches are designed to detect tracking failures and occlusions, to avert bad updates [4,22,32]. Others employ machine learning methods that are robust to sample labeling errors [3,8]. TLD [14] utilizes two experts to generate positive and negative samples, one based on spatial constraints and the other based on temporal constraints, to alleviate drift. In [23], a self-paced learning framework is proposed for long-term tracking, in which the training set is carefully augmented by iteratively revisiting previous frames. Some other methods address the drift problem by incorporating the original template in the updates [20], or by leveraging additional knowledge about the target [15,9]. Our method differs from these past works in either of the following two respects: first, it does not constrain the model update with fixed prior knowledge; second, it is possible for our tracker to undo negative effects after the bad updates that inevitably happen.

3 Multi-Expert Tracking Using Entropy Minimization

In this section, we introduce the multi-expert tracking framework using the minimum entropy criterion. This tracking framework is general and independent of the implementation of the base tracker.

3.1 Expert Selection for Tracking Using Entropy Minimization

We assume that a binary classifier, *i.e.* a discriminative tracker $\mathcal{T}$ is given, and it keeps updating with incoming training samples. We do not differentiate between a discriminative tracker and a binary classifier in this paper, assuming that other status information of a tracker, *e.g.* predictions, is not retained by the tracker.

$\mathcal{S}_t$ denotes a snapshot of the classifier $\mathcal{T}$ at time t. Then $\mathbf{E} := \{\mathcal{T}, \mathcal{S}_{t_1}, \mathcal{S}_{t_2}, \ldots\}$ is an expert ensemble. Let E denote an expert in the ensemble. Each expert E is assigned a loss $\mathfrak{L}_E^t$ at each step t, and the best expert is determined by its cumulative loss within a recent temporal window:

$$E^* = \arg\min_{E \in \mathbf{E}} \sum_{k \in [t-\Delta, t]} \mathfrak{L}_E^k, \tag{1}$$

where Δ is the size of the temporal window.

It is a key task to derive a proper loss function in our multi-expert framework. One straightforward option, which is in the same spirit of many ensemble based tracking methods [16,7,4], is to base the loss function (or weighting function) on the likelihood of the experts, in other words, how well the experts fit the labeled training samples. However, for online model-free tracking, training samples are labeled by the tracker. Therefore, the current tracker always tend to be more confident about its own predictions, and when model drift happens, the high confidence score about the wrong predictions will become completely misleading for the expert selection.

To derive a proper loss function that avoids the aforementioned problem, we employ a formulation that was originally developed for the semi-supervised partial-label learning (PLL) problem [10]. In the PLL problem, learning is based on partially labeled training samples $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{z}_i)\}$, where $\mathbf{z}_i$ represents a possible label set that contains the true label $\mathbf{y}_i$ of instance $\mathbf{x}_i$.

In [10], the PLL is solved within a MAP framework that maximizes the log posterior probability of the model parameterized by θ,

$$\mathcal{C}(\theta, \lambda; \mathcal{L}) = L(\theta; \mathcal{L}) - \lambda H_{emp}(Y|X, Z; \mathcal{L}, \theta), \tag{2}$$

where $L(\theta; \mathcal{L})$ is the log likelihood of the model parameters θ, and $H_{emp}(Y|X, Z; \mathcal{L}, \theta)$ is the empirical conditional entropy of class labels conditioned on the training data and the possible label sets, *i.e.* an empirical approximation of the logarithm of the prior probability of θ. The scalar λ controls the tradeoff between the likelihood and the prior. Readers are referred to [10] for more details. The entropy regularization term favors models with low ambiguity with respect to the partial label sets. For example, when a label set contains two possible labels, then a model giving equally high confidence scores to both labels is less favored than a model giving a high confidence score to one label and a low confidence score to the other.

To use the above minimum entropy criterion in a completely different context, *i.e.* expert selection for tracking, we propose a novel formulation of the tracking-by-detection problem in a multiple instance PLL setting. At each frame, the expert ensemble $\mathbf{E}$ proposes a bag of instances $\mathbf{x} = \{x^1, \ldots, x^n\}$. Each x^i is a candidate image patch cropped from the frame, which is labeled by $y^i = (\omega^i, l^i)$, where $\omega^i \in \{-, +\}$ denotes the foreground-background label and $l^i \in \mathbb{Z}^2$ denotes the pixel-quantized 2D location of the candidate image patch x^i. Without loss of generality, we can now think of the bag $\mathbf{x}$ as a hyper-instance, whose ground truth label $\mathbf{y} = (y^1, \ldots, y^n)$ lies in a high dimensional label space $\mathcal{Y} = (\{-, +\} \times \mathbb{Z}^2)^n$.

We assume that the instance bag $\mathbf{x}$ contains the target, and the candidate image patches do not substantially overlap each other[1]. Thus, only one image patch in the bag can be the true target. Since the location l^i of a candidate image patch x^i is known, the ground truth label $\mathbf{y}$ must be contained in a small possible label set $\mathbf{z} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, where for each $\mathbf{y}_j = ((\omega_j^1, l_j^1), \ldots, (\omega_j^n, l_j^n))$, l_j^i equals l^i, and ω_j^i is labeled as positive only when $i = j$.

Now for each frame, we have an instance bag $\mathbf{x}$ that encodes the appearance of the candidate image patches, and a possible label set $\mathbf{z}$ that encodes the specific constraints of the tracking problem. Therefore, according to Eq. 2, we have the following loss function for our expert selection problem (Eq. 1),

$$\mathfrak{L}_E(\mathbf{x}, \mathbf{z}) = -L(\theta_E; \mathbf{x}, \mathbf{z}) + \lambda H(\mathbf{y}|\mathbf{x}, \mathbf{z}; \theta_E), \tag{3}$$

where we define the log likelihood as

$$L(\theta_E; \mathbf{x}, \mathbf{z}) = \max_{\mathbf{y} \in \mathbf{z}} \log P(\mathbf{y}|\mathbf{x}; \theta_E), \tag{4}$$

and the entropy term is computed by

$$H(\mathbf{y}|\mathbf{x}, \mathbf{z}; \theta_E) = \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \theta_E) \log P(\mathbf{y}|\mathbf{x}, \mathbf{z}; \theta_E). \tag{5}$$

[1] Note that in our algorithm, the instance bag is constructed in a way that candidate image patches do not substantially overlap (see Section 5). This is different from the multiple instance formulation in the MIL tracker [3], where multiple significantly overlapping image patches are sampled purposely so that the true target may align well with multiple patches.

To compute $P(\mathbf{y}|\mathbf{x};\theta_E)$, we assume that each sub-label $y^i = (\omega^i, l^i)$ only depends on x^i. We further assume $P(l^i|\omega^i, x^i) = P(l^i|\omega^i)$, which means their graphical model can be represented by $x^i \rightarrow \omega^i \rightarrow l^i$, *i.e.* the image patch's appearance x^i provides information about its location l^i only through the appearance based posterior $P(\omega^i|x^i;\theta_E)$ and the spatial prior $P(l^i|\omega^i)$. Then we can have the following decomposition:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x};\theta_E) &= \prod_i P(\omega^i, l^i|x^i;\theta_E) \\ &= \prod_i P(l^i|\omega^i)P(\omega^i|x^i;\theta_E), \end{aligned} \tag{6}$$

where the spatial prior $P(l_i|\omega_i = +)$ can be used to encode the motion model. It follows that

$$P(\mathbf{y}|\mathbf{x},\mathbf{z};\theta_E) = \frac{\delta_{\mathbf{z}}(\mathbf{y})P(\mathbf{y}|\mathbf{x};\theta_E)}{\sum_{\mathbf{y}'\in\mathcal{Y}}\delta_{\mathbf{z}}(\mathbf{y}')P(\mathbf{y}'|\mathbf{x};\theta_E)}, \tag{7}$$

which is the Kullback-Leibler projection of $P(\mathbf{y}|\mathbf{x};\theta_E)$. The function $\delta_{\mathbf{z}}(\mathbf{y})$ takes 1 if $\mathbf{y} \in \mathbf{z}$ and 0 otherwise. This concludes all the required computations for Eq. 1.

3.2 Tracking Using Multiple Experts

Given the above formulation, the main loop of the multi-expert tracking framework is composed of the following steps. First, to update the expert ensemble, a snapshot of the tracker is saved every φ frames. The oldest snapshots will be discarded if the number of experts exceeds $\widehat{N}$. Then the expert ensemble proposes an instance bag, which will be detailed in Sec. 5. Given the instance bag and the possible label set described in Section 3.1, the loss function is evaluated for each expert using Eq. 3-7. After that, if a disagreement among the experts is detected, the best expert according to Eq. 1 will be assigned to the current tracker. Note that if the current tracker is the best one, then no restoration occurs. Finally, the tracker outputs the prediction, based on which the tracker is updated. A summary of our multi-expert tracking framework is given in Alg. 1.

4 Online Linear SVM Tracker

The base tracker for our multi-expert framework is inspired by the online SVM algorithm of [26], which makes use of prototype sets to gain an improved approximation to the offline SVM. For our base tracker, the algorithm of [26] is reformulated to better suit the tracking problem. Note that the following formulation is for a stand-alone tracker, which is independent of the multi-expert framework.

The SVM tracker $\mathcal{T}$ contains a compact prototype set $Q = \{\zeta_i = (\phi(q_i), \omega_i, s_i)\}_1^B$ to summarize the previous training data, where $\phi(q_i)$ is the feature vector of an image patch q_i, ω_i is a binary label, and s_i is a counting number that indicates how many support vectors are represented by this instance. We re-train the SVM classifier at each frame using the prototype set and the new data $\mathcal{L} = \{(x_i, y_i)\}_1^J$ by minimizing

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\{\sum_{i=1}^{B} \frac{s_i}{N_{\omega_i}} L_h(\omega_i, q_i;\mathbf{w}) + \sum_{i=1}^{J} \frac{1}{N_{y_i}} L_h(y_i, x_i;\mathbf{w})\}, \tag{8}$$

Alg. 1. MEEM-TRACK

```
input : frames $\{I_t\}_0^T$, initial bounding box $\mathbf{b}_0$
output: bounding box predictions$\{\mathbf{b}_t\}_1^T$

initialize tracker $\mathcal{T}$ using $I_0$ and $\mathbf{b}_0$
$\mathbf{E} \leftarrow \{\mathcal{T}\}$
for $t = 1 : T$ do
    if $\mod(t, \varphi) = 0$ then
        $\mathbf{E} \leftarrow \mathbf{E} \cup \{\mathcal{S}_t \leftarrow \mathcal{T}\}$, discard the oldest expert when $|\mathbf{E}| > \widehat{N}$
    get the instance bag and the label set $(\mathbf{x}, \mathbf{z})$ from $\mathbf{E}$
    foreach $E \in \mathbf{E}$ do compute $\mathfrak{L}_E^t$ by Eq. 3-7 if a disagreement among the experts is detected
    then
        $\mathcal{T} \leftarrow E^*$ by expert selection using Eq. 1
    predict $\mathbf{b}_t$ by $\mathcal{T}$
    re-train $\mathcal{T}$ using $I_t$ and $\mathbf{b}_t$
```

Alg. 2. SVM-UPDATE

```
input : Tracker $\mathcal{T} = (\mathbf{w}, b, Q)$, training samples $\mathcal{L} = \{(x_i, y_i)\}$
output: Updated tracker $\mathcal{T}$

compute $(\mathbf{w}, b)$ using $\mathcal{L}$ and $Q$, given in Eq. 8 and Eq. 9
$Q \leftarrow Q \cup \{(\phi(x_{i_k}), y_{i_k}, 1) : (x_{i_k}, y_{i_k}) \in \mathcal{L}$ is a support vector$\}$
while $|Q| > \widehat{B}$ do
    if $|Q^+| > \widehat{B}^+$ then
        $(i_1, i_2) \leftarrow \arg\min_{\{(i_1,i_2):\omega_{i1}=\omega_{i2}\}} \|\phi(q_{i1}) - \phi(q_{i2})\|$
    else $(i_1, i_2) \leftarrow \arg\min_{\{(i_1,i_2):\omega_{i1}=\omega_{i2}=-\}} \|\phi(q_{i1}) - \phi(q_{i2})\|$ $\zeta^* \leftarrow$MERGE($\zeta_{i_1}, \zeta_{i_2}$) by
    Eq. 10
    delete $\{\zeta_{i_1}, \zeta_{i_2}\}$ from $Q$, and add $\zeta^*$ to $Q$
```

where L_h is the hinge loss, and

$$N_+ = \sum_{\omega_i=+} s_i + \sum_{y_i=+} 1, \;\; N_- = \sum_{\omega_i=-} s_i + \sum_{y_i=-} 1 \tag{9}$$

are used to equalize the total weight of the positive samples and that of the negative samples. This is to account for the imbalance of training samples, which is not considered in [26]. From Eq. 8, it can be seen that prototype instances with larger counting numbers have greater influence on training.

After training, support vectors from the new training data are added to the prototype set with counting number 1. When the size of the prototype set is larger than a predefined budget $\widehat{B}$, the pair of prototype instances of the same label with the minimal distance in the feature space are merged into $\zeta^* = (\phi(q^*), \omega^*, s^*)$, where

$$\phi(q^*) = \frac{s_{i_1}\phi(q_{i_1}) + s_{i_2}\phi(q_{i_2})}{s_{i_1} + s_{i_2}}, \;\omega^* = \omega_{i_1}, \;\; s^* = s_{i_1} + s_{i_2}. \tag{10}$$

Since positive samples usually have much lower diversity than negative ones in the tracking problem, the algorithm of [26] tends to make the positive prototype instances collapse into a single instance. To avoid this for our SVM tracker, positive prototype instances are not merged until their number $|Q^{+}|$ reaches a predefined bound $\widehat{B}^{+}$. The complete online SVM algorithm is described in Alg. 2. In our implementation, we use $C = 100$, $\widehat{B} = 80$ and $\widehat{B}^{+} = 10$. We have found that, in practice, the performance of our tracker tends to be insensitive to these settings.

To obtain nonlinear decision boundaries with linear SVM, we use the feature mapping technique proposed in [19] to approximate the min kernel SVM. Suppose that each component a of a feature vector $\mathbf{v} = [a_i]$ is in the range $[0, 1]$, and we discretize $[0, 1]$ into K levels. Then the mapping is defined as

$$\phi(a) = \mathcal{U}(\mathcal{R}(Ka)), \tag{11}$$

where $\mathcal{R}(.)$ is a rounding function and $\mathcal{U}(.)$ is a unary representation transformation. For example, when $K = 5$, $\phi(0.6) = \mathcal{U}(3) = [1, 1, 1, 0, 0]$. Then $\phi(\mathbf{v}) = [\phi(a_i)]$ is fed to the SVM classifier for training and inference.

5 Implementation Details

Base Tracker. In the implementation of the SVM tracker, only translation is considered for efficiency. Search for the target is conducted on a Cartesian grid of unit step ϵ_{step} within a radius of $\sqrt{wh}$ of the previous prediction, where (w, h) is the template size. The predicted position gives the positive sample, and the local image patches that do not significantly overlap the prediction (IOU < 0.5) are the negative ones.

Images are transformed into CIE Lab color space. To provide robustness to drastic illumination variations, a non-parametric local rank transform [30] is applied on the L channel of the image. This transform produces a feature map that is invariant to any monotonically increasing transformations of pixel intensities. This feature map and the Lab channels constitute a 4-channel source image, where the appearance of an image patch is represented by its spatially down-sampled version using a sample step that equals ϵ_{step}. This down-sampled 4-channel image patch is reshaped into a vector, which is to be transformed by the feature mapping technique with the quantization number $K = 4$ for our base tracker. The sample step ϵ_{step} is automatically set at runtime, so that the final feature dimension of an image patch is approximately 2000. Training involves about 200 training samples, which takes less than 0.1s in our Matlab implementation.

Multi-Expert Framework. To get the candidate instance bag $\mathbf{x} = \{x^1, \ldots, x^n\}$ for a frame, each expert E outputs a confidence map $\mathcal{F}_E$ for the search region by computing

$$\mathcal{F}_E^{ij} = P(l^{ij}|+)P(+|\phi^{ij}; \theta_E) \tag{12}$$

on the search grid (i, j). l^{ij} and ϕ^{ij} are the location and the feature respectively. For $P(+|\phi^{ij}; \theta_E)$, the SVM scores are transformed to the probability form by a Gaussian cumulative distribution with mean of 0 and STD of 1, and thereby $P(-|\phi^{ij}; \theta_E) = 1 - P(+|\phi^{ij}; \theta_E)$. The spatial prior $P(l^{ij}|+)$ is a 2D Gaussian distribution centered at the previously predicted location with STD σ, and $P(l^{ij}|-)$ is a uniform distribution. Both the spatial prior density functions are normalized for the search grid so that

$\sum_{ij} P(l^{ij}|+) = \sum_{ij} P(l^{ij}|-) = 1$. Each confidence map $\mathcal{F}_E$ is then shifted and scaled to range from 0 to 1.

After non-maxima suppression of $\mathcal{F}_E$ with an $r \times r$ kernel, image patches corresponding to the local maxima with confidence value greater than ψ are added to the candidate instance bag $\mathbf{x}$. If the center distance of two candidate image patches proposed by different experts is smaller than r, we merge the pair to the image patch at their mean position, so that the candidate patches do not substantially overlap.

The global maximum on each confidence map $\mathcal{F}_E$ serves as the prediction of E. If any of the predictions of the experts deviates from their mean position by a distance more than r, a disagreement of the experts is detected. Then the expert selected via Eq. 1 will be assigned to the current tracker.

In our experiments, we use $\sigma = 15$, $r = 5$, both in grid units, and $\psi = 0.9$. We set $\Delta = 5$ in Eq. 1 and $\lambda = 10$ in Eq. 3. These parameters are set via grid search on a small training set. The maximum number of experts $\widehat{N}$ and the time interval for saving a snapshot φ are set to 4 and 50 respectively. We find that increasing $\widehat{N}$ only slightly improves the performance in practice, but more computation would be needed to evaluate the experts at each time step.

Our algorithms are implemented in Matlab and C. It on average runs at roughly 10fps on a 2.93GHz CPU with 8GB memory. Source code is available on our website[2]. All parameters of our tracker are fixed throughout the experiments.

6 Experiment I: General Comparison

In this section, we report an extensive evaluation of the proposed tracking method, denoted as MEEM, in comparison with other state-of-the-art trackers. Testing a tracker on a small number of sequences can sometimes cause biased evaluation because of the peculiarities of the selected sequences. To avoid this problem, we use the benchmark dataset of 50 sequences proposed by [27]. This dataset contains many sequences used in the previous literature, and covers a variety of challenging scenarios for visual tracking.

Evaluation Setting and Metrics. Three experiments are performed as in [27]: one pass evaluation (OPE), temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). TRE randomizes the starting frame and runs a tracker through the rest of the sequences, and SRE randomizes the initial bounding boxes by shifting and scaling. We use the same spatial and temporal randomization as in [27], and refer readers to [27] for more details. As pointed out by [27], traditional one-pass evaluation cannot fully reflect the robustness of a tracker, and sometimes even a small perturbation can cause very different tracking results.

Following [27], two evaluation methods are used: precision plot and success plot. Both plots show the percentage of successfully tracked frames vs. the threshold. The precision plot thresholds the center location error (in pixels) and the success plot thresholds the intersection over union (IOU) metric. As discussed in [27,3], the precision plot and the success plot are more informative than some widely used metrics, *e.g.* the success rate and the average center location error. To rank the trackers, two types of ranking

[2] http://www.cs.bu.edu/groups/ivc/software/MEEM/

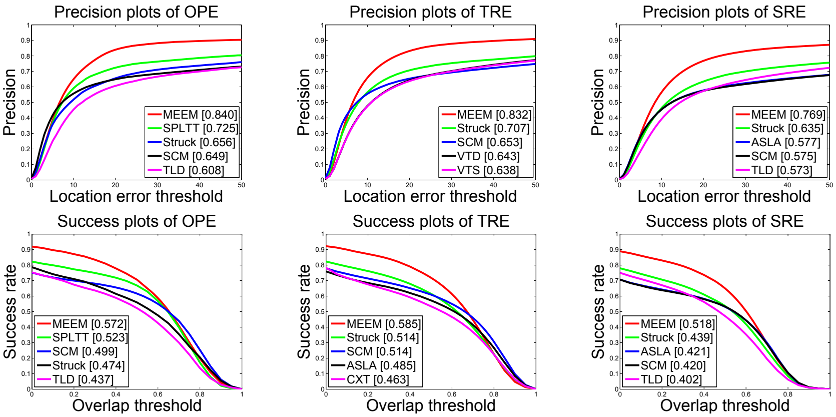


Fig. 2. Average precision plots (top row) and success plots (bottom row) for OPE, SRE and TRE (best viewed in color). The top five trackers with respect to the ranking scores are shown in each plot. The numbers in the square brackets are the ranking scores of the trackers, averaged over all 50 test sequences. Note that the line style of a curve is determined by the ranking of the corresponding tracker in the plot, not by the name of the tracker.

metrics are used as in [27]: the representative precision score at threshold $= 20$ for the precision plot, and the Area Under the Curve (AUC) metric for the success plot. Plots and ranking metrics are computed using the software and annotations provided by [27].

Compared Algorithms. Results of 29 trackers on this benchmark dataset are reported in [27]. For a more complete comparison, we also include three more recent trackers in this experiment: LSHT [12], LSST [25] and SPLTT [23]. SPLTT is only evaluated for OPE due to limited computational resources. We also note that SPLTT employs batch processing for all previous frames on each model update; thus, it is not directly comparable with the other trackers that assume a constant memory budget.

Results. Precision and success plots are shown in Fig. 2. A tracker's curve on a plot is computed by averaging its curves on all 50 test sequences. Due to limited space, only the results of the top five trackers are reported in each plot (SPLTT [23], Struck [11], SCM [32], TLD [14], VTD [16], VTS [17], ASLA [13] and CXT [6]). For results of other trackers, we refer the readers to our supplementary materials. Note that the rankings of the trackers vary on different plots.

From Fig. 2, it can be seen that MEEM attains the best overall performance by a significant margin in all evaluation settings. For example, in the precision plots, the ranking score of MEEM outperforms the second best score by over 0.11 in OPE, TRE and SRE, which is a performance gain of over 15%. Trackers usually give higher ranking scores in TRE and lower ones in SRE than in OPE. This is because in TRE, a tracker is tested by multiple runs starting at different time positions of a sequence, and thus a tracker may skip the challenging parts of a sequence. In contrast, SRE is more challenging due to the misalignment of the initial bounding box.

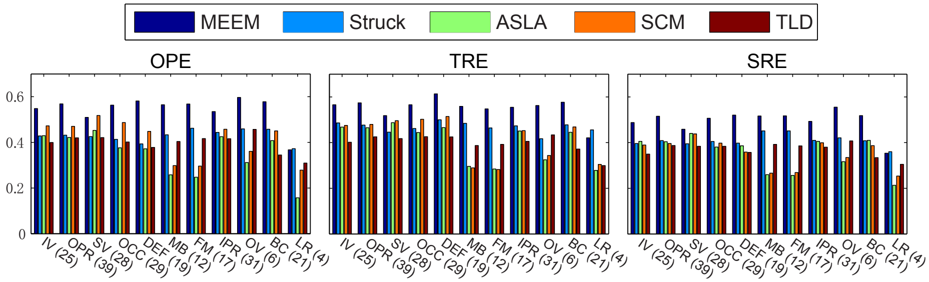


Fig. 3. Average AUC ranking scores of the five leading trackers on different subsets of test sequences in OPE, TRE and SRE (best viewed in color). Each subset of sequences corresponds to an attribute, such as illumination variation (IV), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-view (OV), background clutter (BC), low resolution (LR). The number after each attribute name is the number of sequences that have this attribute. Trackers displayed here are selected based on their AUC ranking scores in SRE.

Each of the 50 benchmark videos is also annotated with attributes that indicate what kinds of challenging factors occur within it. Fig. 3 shows the AUC ranking scores of the leading trackers on different groups of sequences, where each group corresponds to a different attribute. For example, the group of “deformation” (DEF) contains all the sequences in which the target undergoes non-rigid deformation. See [27] for detailed definitions of the annotations. Note that a single video may include multiple attributes.

From Fig. 3, it can be seen that in all evaluation settings, MEEM substantially outperforms the other state-of-the-art trackers on such attribute groups as “deformation” (DEF), “out-of-plane rotation” (OPR), “in-plane rotation (IPR)”, “out-of-view (OV)” and “occlusion” (OCC). The ranking scores of precision plots of MEEM show a similar trend, and they are provided as supplementary materials. To be more specific, MEEM tends to better handle those sequences like “basketball”, “bolt”, “david3”, “ironman”, “lemming”, “liquor”, “matrix”, and “soccer”, which feature either severe occlusions or large appearance variations. This observation is consistent with the overall formulation of our tracking method. Our base tracker, based on an online SVM algorithm with prototype set maintenance and the explicit feature mapping [19], can efficiently find the nonlinear decision boundary through online training. Moreover, when model drift is inevitable, the proposed multi-expert restoration scheme can also help to correct that drift. We also note that even though our tracking method does not account for scale changes, it still compares favorably with the state-of-the-art trackers on sequences with substantial scale variations (SV), as shown in Fig. 3.

To gain further insight into the performance of MEEM, we also compare it with several baselines: (1) MEEM-lkh, a version of MEEM using only the likelihood term for expert selection, *i.e.* $\lambda = 0$ in Eqn. 3; (2) SVM-avg, the model average of the experts; (3) SVM-base, our base tracker. The results of this comparison are summarized in Table 1 and detailed tables are provided as supplementary material. MEEM outperforms its baselines in all evaluation settings. MEEM gives significantly better performance

Table 1. Average ranking scores of MEEM and SVM-base

	Precision			Success		
	OPE	TRE	SRE	OPE	TRE	SRE
MEEM	0.840	0.832	0.769	0.572	0.585	0.518
MEEM-lkh	0.815	0.819	0.748	0.561	0.578	0.504
SVM-avg	0.804	0.817	0.746	0.559	0.574	0.503
SVM-base	0.804	0.817	0.747	0.559	0.574	0.503

than its baselines on a few of the sequences, such as “david3”, “lemming”, “jogging-1” and “jogging-2”, where factors like occlusions and out-of-plane rotations could lead to model drift. This indicates the proposed entropy-regularized restoration scheme is especially useful in those scenarios. We give further analysis of the our tracking method on more sequences with such challenging factors in the next section.

7 Experiment II: Analysis of MEEM

We now further analyze and illustrate the benefit of the proposed MEEM framework on a newly collected video dataset that better reflects the real world scenarios of frequent occlusions and repetitive appearance variations.

Dataset. To control factors irrelevant to our analysis, *e.g.* large scale changes and highly non-rigid motions, we gathered a new dataset of ten sequences with moderate scale variations, where the target object can be approximately represented by a rigid rectangular template. Most sequences are from Youtube, except “ped1” and “ped2”, which are from [5]. These sequences feature severe occlusions (“dance”, “boxing1”, “boxing2”, “ped1”, “ped2”), abrupt illumination changes (“carRace”, “billieJean”), low contrast (“ball”, “ped2”, “rocky”, “billieJean”), and large repetitive appearance variations (“latin”, ‘ball”, “carRace”, “dance”, “billieJean”). The total number of frames in this dataset is more than 7500. These sequences tend to cause the drift problem and tracking failure for many state-of-the-art trackers. Sample frames from these sequences are shown in Fig. 4. Test sequences and annotations are available on our website.

Evaluation Setting and Metrics. We use both OPE and SRE for evaluation, so that our analysis will not be sensitive to the perturbation of initialization. Note that TRE is less

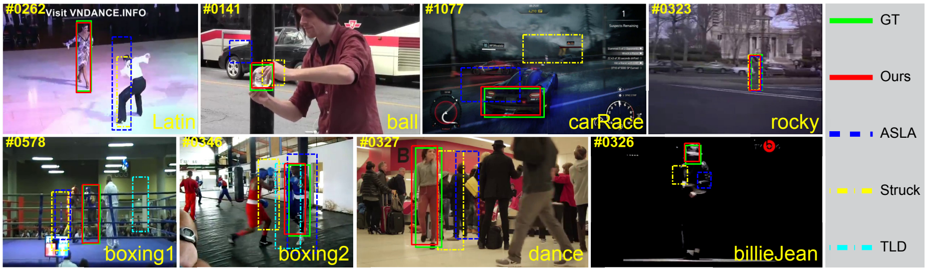
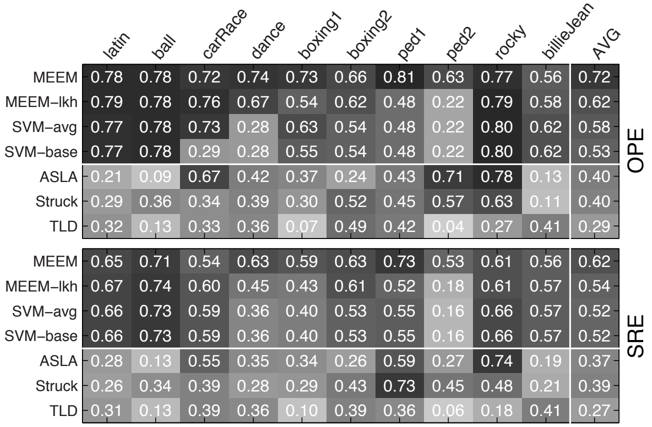


Fig. 4. Example frames from the test sequences (best viewed in color)



		latin	ball	carRace	dance	boxing1	boxing2	ped1	ped2	rocky	billieJean	AVG
OPE	MEEM	0.78	0.78	0.72	0.74	0.73	0.66	0.81	0.63	0.77	0.56	0.72
	MEEM–lkh	0.79	0.78	0.76	0.67	0.54	0.62	0.48	0.22	0.79	0.58	0.62
	SVM–avg	0.77	0.78	0.73	0.28	0.63	0.54	0.48	0.22	0.80	0.62	0.58
	SVM–base	0.77	0.78	0.29	0.28	0.55	0.54	0.48	0.22	0.80	0.62	0.53
	ASLA	0.21	0.09	0.67	0.42	0.37	0.24	0.43	0.71	0.78	0.13	0.40
	Struck	0.29	0.36	0.34	0.39	0.30	0.52	0.45	0.57	0.63	0.11	0.40
	TLD	0.32	0.13	0.33	0.36	0.07	0.49	0.42	0.04	0.27	0.41	0.29
SRE	MEEM	0.65	0.71	0.54	0.63	0.59	0.63	0.73	0.53	0.61	0.56	0.62
	MEEM–lkh	0.67	0.74	0.60	0.45	0.43	0.61	0.52	0.18	0.61	0.57	0.54
	SVM–avg	0.66	0.73	0.59	0.36	0.40	0.53	0.55	0.16	0.66	0.57	0.52
	SVM–base	0.66	0.73	0.59	0.36	0.40	0.53	0.55	0.16	0.66	0.57	0.52
	ASLA	0.28	0.13	0.55	0.35	0.34	0.26	0.59	0.27	0.74	0.19	0.37
	Struck	0.26	0.34	0.39	0.28	0.29	0.43	0.73	0.45	0.48	0.21	0.39
	TLD	0.31	0.13	0.39	0.36	0.10	0.39	0.36	0.06	0.18	0.41	0.27

Fig. 5. AUC ranking scores of MEEM, its baselines, and other state-of-the-art trackers for each sequence. Darker cells indicate higher scores. The last column shows the average AUC scores.

suitable for our purposes, since it can make the drift problem less obvious by skipping some parts of a sequence. The same evaluation metrics, precision plot and success plot, are used as in Sec. 6.

Compared Algorithms. We focus on the comparison of MEEM and its baselines, MEEM-lkh, SVM-avg and SVM-base (see Section 6). Scores of some state-of-the-art trackers, ASLA [13], Struck [11] and TLD [14], are also reported, to give a sense of the difficulty levels of the test sequences.

Results. Fig. 5 reports the AUC ranking scores of MEEM, its baselines, and other trackers on each test sequence. On average, MEEM outperforms its baselines and other compared trackers by at least 15% in terms of the AUC ranking score in both OPE and SRE. SVM-avg gives similar overall performance as SVM-base, and MEEM-lkh is slightly better than SVM-avg and SVM-base. This indicates that the the proposed restoration scheme can better alleviate the drift problem than model averaging on the test sequences. In general, model averaging will sacrifice the adaptivity of a tracker, which is not favorable when the target undergoes large appearance changes. The substantial improvement of MEEM over MEEM-lkh verifies the advantage of the entropy regularization term in our expert selection function. On all the test sequences, MEEM is comparable with, if not better than, its best baseline. Significant performance improvement of MEEM over at least one of its baselines is observed on "carRace" (OPE), "dance" (OPE, SRE), "boxing1" (OPE, SRE), "boxing2" (OPE, SRE), "ped1" (OPE, SRE) and "ped2" (OPE, SRE).

In "carRace", the appearance of the car often changes abruptly due to illumination variation and out-of-plane rotation. In "dance", "boxing1", "boxing2", "ped1" and "ped2", the tracked person undergoes different levels of occlusion, non-rigid motion and out-of-plane rotation. These challenging factors often cause the baselines and the other compared trackers to drift on those sequences. In contrast, when the appearance of the

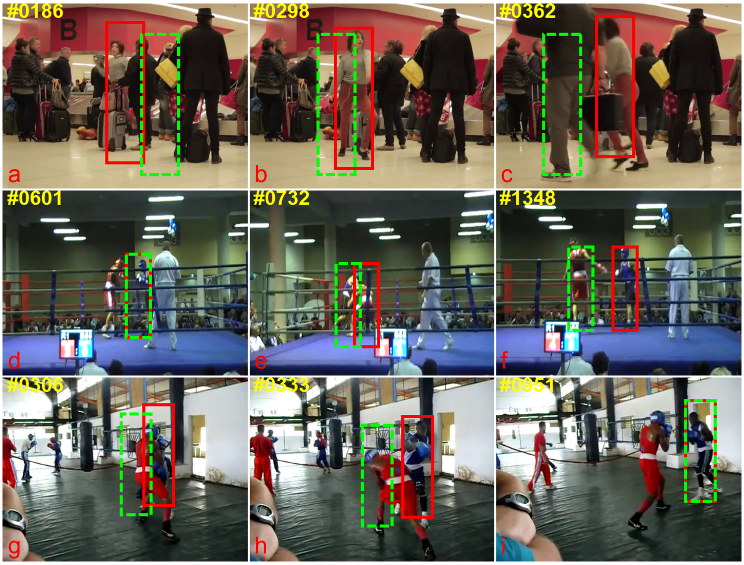


Fig. 6. Example frames where the tracker restoration occurs. (a)-(c) are from "dance" , where the target is the girl; (d)-(f) are from "boxing1", where the target is the boxer in blue; (g)-(h) are from "boxing2", where the target is the boxer in blue. The dashed green rectangles are the predictions of the current tracker before restoration, and the red ones are its predictions after restoration.

target becomes consistent with some previous snapshots again, our entropy-regularized multi-expert scheme can often detect the model drift and restore the tracker.

Fig. 6 shows examples of tracker restoration. In many cases, model drift is corrected by the restoration, resulting in a better localization of the target. It can also happen that the restored tracker gives the same prediction as the original one (*e.g.* Fig. 6(d)(i)), but restoration removes the effects of some recent model updates, which may have made the current tracker more ambiguous. Sometimes a restoration may lead to worse predictions (*e.g.* Fig. 6(e)). However, mistakes made in expert selection do not affect the snapshots already saved, but only the current tracker. Therefore, the effects of an undesirable tracker restoration may also be undone later on, when the target's appearance becomes consistent with some previous snapshots again.

8 Conclusions

In this paper, we propose a multi-expert tracking framework, where the base tracker can evolve backwards to correct undesirable effects of bad model updates using an entorpy-regularized restoration scheme. Our base tracker exploits an online linear SVM algorithm, which uses a prototype set to manage the training samples, and an explicit feature mapping technique for efficient model update. The experimental results demonstrated the superior performance of our method, and the utility of the multi-expert scheme for drift correction.

Acknowledgments. This work was supported in part through grants from the US National Science Foundation #1029430 and #0910908.

References

1. Avidan, S.: Support vector tracking. PAMI 26(8), 1064–1072 (2004)
2. Avidan, S.: Ensemble tracking. PAMI 29(2), 261–271 (2007)
3. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. PAMI 33(8), 1619–1632 (2011)
4. Bai, Q., Wu, Z., Sclaroff, S., Betke, M., Monnier, C.: Randomized ensemble tracking. In: ICCV (2013)
5. Chu, D.M., Smeulders, A.W.: Thirteen hard cases in visual tracking. In: AVSS (2010)
6. Dinh, T.B., Vo, N., Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: CVPR (2011)
7. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: BMVC (2006)
8. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
9. Grabner, M., Grabner, H., Bischof, H.: Learning features for tracking. In: CVPR (2007)
10. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: NIPS (2005)
11. Hare, S., Saffari, A., Torr, P.H.: Struck: Structured output tracking with kernels. In: ICCV (2011)
12. He, S., Yang, Q., Lau, R.W., Wang, J., Yang, M.H.: Visual tracking via locality sensitive histograms. In: CVPR (2013)
13. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: CVPR (2012)
14. Kalal, Z., Matas, J., Mikolajczyk, K.: Pn learning: Bootstrapping binary classifiers by structural constraints. In: CVPR (2010)
15. Kim, M., Kumar, S., Pavlovic, V., Rowley, H.: Face tracking and recognition with visual constraints in real-world videos. In: CVPR (2008)
16. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: CVPR (2010)
17. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: ICCV (2011)
18. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., van den Hengel, A.: A survey of appearance models in visual object tracking. arXiv preprint arXiv:1303.4803 (2013)
19. Maji, S., Berg, A.C.: Max-margin additive classifiers for detection. In: CVPR (2009)
20. Matthews, L., Ishikawa, T., Baker, S.: The template update problem. PAMI 26(6), 810–815 (2004)
21. Mei, X., Ling, H.: Robust visual tracking and vehicle classification via sparse representation. PAMI 33(11), 2259–2272 (2011)
22. Mei, X., Ling, H., Wu, Y., Blasch, E., Bai, L.: Minimum error bounded efficient l1 tracker with occlusion detection. In: CVPR (2011)
23. Supancic III, J.S., Ramanan, D.: Self-paced learning for long-term tracking. In: CVPR (2013)
24. Tang, F., Brennan, S., Zhao, Q., Tao, H.: Co-tracking using semi-supervised support vector machines. In: ICCV (2007)
25. Wang, D., Lu, H., Yang, M.H.: Least soft-thresold squares tracking. In: CVPR (2013)
26. Wang, Z., Vucetic, S.: Online training on a budget of support vector machines using twin prototypes. Statistical Analysis and Data Mining 3(3), 149–169 (2010)

27. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR (2013)
28. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys (CSUR) 38(4), 13 (2006)
29. Yu, Q., Dinh, T.B., Medioni, G.: Online tracking and reacquisition using co-trained generative and discriminative trackers. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 678–691. Springer, Heidelberg (2008)
30. Zabih, R., Woodfill, J.: Non-parametric local transforms for computing visual correspondence. In: Eklundh, J.-O. (ed.) ECCV 1994. LNCS, vol. 801, Springer, Heidelberg (1994)
31. Zhang, T., Ghanem, B., Liu, S., Ahuja, N.: Low-rank sparse learning for robust visual tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VI. LNCS, vol. 7577, pp. 470–484. Springer, Heidelberg (2012)
32. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparsity-based collaborative model. In: CVPR (2012)