High-Performance Long-Term Tracking with Meta-Updater

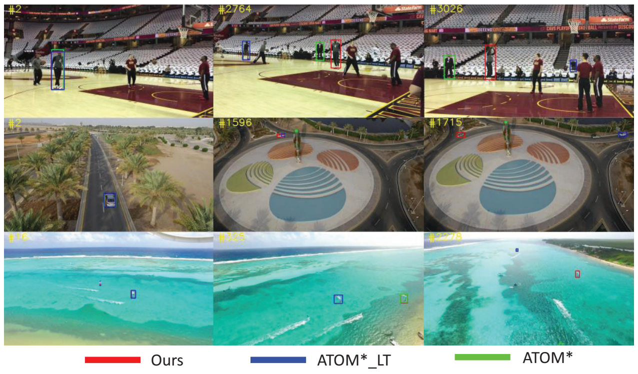
Kenan Dai[1], Yunhua Zhang[2], Dong Wang[1*], Jianhua Li[1], Huchuan Lu[1,4], Xiaoyun Yang[3]
[1]School of Information and Communication Engineering, Dalian University of Technology, China
[2]University of Amsterdam [3]China Science IntelliCloud Technology Co., Ltd [4]Peng Cheng Laboratory
dkn2014@mail.dlut.edu.cn, y.zhang9@uva.nl, wdice@dlut.edu.cn
jianhual@dlut.edu.cn, lhchuan@dlut.edu.cn, xiaoyun.yang@intellicloud.ai

Abstract

Long-term visual tracking has drawn increasing attention because it is much closer to practical applications than short-term tracking. Most top-ranked long-term trackers adopt the offline-trained Siamese architectures, thus, they cannot benefit from great progress of short-term trackers with online update. However, it is quite risky to straightforwardly introduce online-update-based trackers to solve the long-term problem, due to long-term uncertain and noisy observations. In this work, we propose a novel offline-trained ***Meta-Updater*** *to address an important but unsolved problem: Is the tracker ready for updating in the current frame? The proposed meta-updater can effectively integrate geometric, discriminative, and appearance cues in a sequential manner, and then mine the sequential information with a designed cascaded LSTM module. Our meta-updater learns a binary output to guide the tracker's update and can be easily embedded into different trackers. This work also introduces a long-term tracking framework consisting of an online local tracker, an online verifier, a SiamRPN-based re-detector, and our meta-updater. Numerous experimental results on the VOT2018LT, VOT2019LT, OxUvALT, TLP, and LaSOT benchmarks show that our tracker performs remarkably better than other competing algorithms. Our project is available on the website:* https://github.com/Daikenan/LTMU.

1. Introduction

The study of visual tracking has begun to shift from short-term tracking to large-scale long-term tracking, roughly due to two reasons. First, long-term tracking is much closer to practical applications than short-term tracking. The average length of sequences in short-term tracking benchmarks (OTB [46], VOT2018 [23], TC128 [31], to name a few) is often at the second level, whereas the av-



	ATOM*	ATOM*_LT	Ours	CLGS	SiamDW_LT
F-score	0.527	0.651	**0.697**	0.674	0.665
Pr	0.589	0.685	0.721	**0.739**	0.697
Re	0.477	0.621	**0.674**	0.619	0.636

Figure 1. Visualization and comparisons of representative long-term tracking results on VOT2019LT. "ATOM*" is our local tracker based on ATOM [9], "Ours" denotes our long-term tracker with meta-update. "ATOM*_LT" means "Ours" without meta-updater. "CLGS" and "SiamDW_LT" are the second and third best trackers on VOT2019LT. Please see Sections 3 and 4 for more details.

erage frame length in long-term tracking datasets (such as VOT2018LT [23], VOT2019LT [24], and OxUvALT [42]) is at least at the minute level. Second, the long-term tracking task additionally requires the tracker having the capability to handle frequent disappearance and reappearance (i.e., having a strong re-detection capability)[1].

Deep-learning-based methods have dominated the short-term tracking field [30, 47, 35], from the perspective of either one-shot learning [41, 2, 15, 28, 26, 12, 53, 29] or online learning [37, 10, 8, 21, 40, 7, 49, 50, 9]. Usually, the latter methods (e.g., ECO [8], ATOM [9]) are more accurate (with less training data) but slower than the former ones (e.g., SiamFC [2], SiamRPN [28]). A curious phenomenon is that few leading long-term trackers exploit online-updated short-term trackers to conduct local track-

*Corresponding Author: Dr. Dong Wang, wdice@dlut.edu.cn

[1]More resources about long-term tracking can be found in https://github.com/wangdongdut/Long-term-Visual-Tracking.

ing. MBMD [51], the winner of VOT2018LT, exploits an offline-trained regression network to directly regress the target's bounding box in a local region, and uses an online-learned verifier to make the tracker switch between local tracking and global re-detection. The recent SPLT [48] method utilizes the same SiamRPN model in [51] for local tracking. SiamFC+R [42], the best method in the OxUvALT report, equips the original SiamFC [2] with a simple re-detection scheme. An important reason is that online update is a double-edged sword for tracking. Online update captures appearance variations from both target and background, but inevitably pollutes the model with noisy samples. The risk of online update is amplified for long-term tracking, due to long-term uncertain observations.

Motivated by the aforementioned analysis, this work attempts to improve the long-term tracking performance from two aspects. First, we design a long-term tracking framework that exploits an online-updated tracker for local tracking. As seen in Figure 1, the tracking performance is remarkably improved by extending ATOM* to a long-term tracker (ATOM*_LT), but it remains worse than the CLGS and SiamDW_LT methods. Second, we propose a novel meta-updater to effectively guide the tracker's update. Figure 1 shows that after adding our meta-updater, the proposed tracker achieves very promising tracking results.

Our main contributions can be summarized as follows.

- *A novel offline-trained meta-updater is proposed to address an important but unsolved problem: Is the tracker ready for updating in the current frame? The proposed meta-updater effectively guide the update of the online tracker, not only facilitating the proposed tracker but also having good generalization ability.*
- *A long-term tracking framework is introduced on the basis of a SiamRPN-based re-detector, an online verifier, and an online local tracker with our meta-updater. Compared with other methods, our long-term tracking framework can benefit from the strength of online-updated short-term tracker at low risk.*
- *Numerous experimental results on the VOT2018LT, VOT2019LT, OxUvALT, TLP and LaSOT long-term benchmarks show that the proposed method outperforms the state-of-the-art trackers by a large margin.*

2. Related Work

2.1. Long-term Visual Tracking

Although large-scale long-term tracking benchmarks [23, 42] began to emerge since 2018, researchers have attached importance to the long-term tracking task for a long time (such as keypoint-based [17], proposal-based [54], detector-based [22, 32], and other methods). A classical algorithm is the tracking-learning-detection (TLD) method [22], which addresses long-term tracking as a combination of a local tracker (with forward-backward optical flow) and a global re-detector (with an ensemble of weak classifiers). Following this idea, many researchers [34, 32, 42] attempt to handle the long-term tracking problem with different local trackers and different global re-detectors. Among them, the local tracker and global re-detectors can also adopt the same powerful model [32, 26, 51, 48], being equipped with a re-detection scheme (e.g., random search and sliding window). A crucial problem of these trackers is how to switch the tracker between the local tracker and the global re-detector. Usually, they use the outputs of local trackers to conduct self-evaluation, i.e., to determine whether the tracker losses the target or not. This manner has a high risk since the outputs of local trackers are not always reliable and unexpectedly mislead the switcher sometimes. The MBMD method [51], the winner of VOT2018LT, conducts local and global switching with an additional online-updated deep classifier. This tracker exploits a SiamPRN-based network to regress the target in a local search region or every sliding window when re-detection. The recent SPLT method [48] utilizes the same SiamPRN in [51] for tracking and re-detection, replaces the online verifier in [51] with an offline trained matching network, and speeds up the tracker by using their proposed skimming module. A curious phenomenon is that most top-ranked long-term trackers (such as MBMD [51], SPLT [48], and SiamRPN++ [26]), have not adopted excellent online-updated trackers (e.g., ECO [8], ATOM [9]) to conduct local tracking. One of the underlying reasons is that the risk of online update is amplified for long-term tracking, caused by long-term uncertain observations. In this work, we attempt to address this dilemma by designing a high-performance long-term tracker with a meta-updater.

2.2. Online Update for Visual Tracking

For visual tracking, online update acts as a vital role to capture appearance variations from both target and its surrounding background during the tracking process. Numerous schemes have been designed to achieve this goal by using template update [6, 55, 29], incremental subspace learning [39, 43], online learning classifiers [16, 37, 8, 9], to name a few. However, online update is a double-edged sword in balancing the dynamical information description and unexpected noise introduction. Accumulating errors over a long time, collecting inappropriate samples or overfitting to available data when the target disappears can easily degrade the tracker and lead to tracking drift, especially for long-term tracking. To deal with this dilemma, many efforts have been done at least from two aspects. The first one aims to distill the online collected samples by recovering or clustering noisy observations [43, 8]. Another effective attempt is to design some criteria for evaluating the reliability of the current tracking result, to remove the unreliable samples or reject the inappropriate update. These criteria include the confidence score [37], the maximum (MAX) response [9], peak-to-sidelobe rate (PSR) [9], average peak-to-correlation energy [44], and MAX-PSR [32]. These methods usually utilize the tracker's output to self-evaluate this reliability. But the self-evaluation of the trackers' reliability with its outputs has inevitable risks, especially when the tracker experiences the long-term uncertain and noisy observations. In this work, we propose a novel offline-trained meta-updater to integrate multiple cues in a sequential manner. The meta-updater outputs a binary score to indicate whether the tracker should be updated or not in the current frame, which not only remarkably improves the performance of our long-term tracker but also is easy to be embedded into other online-updated trackers. Recently, some meta-learning-based methods [25, 38, 27, 18, 5, 29] have been presented. All these methods focus on addressing the "how to update" problem (i.e., efficiently and/or effectively updating the trackers' appearance models). By contrast, our meta-updater is designed to deal with the "when to update" problem, and it can be combined with many "how to update" algorithms to further improve the tracking performance.

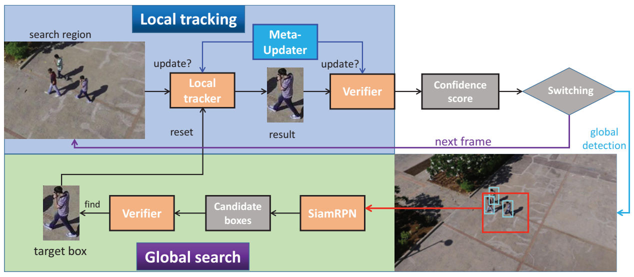


Figure 2. Proposed long-term tracking framework. Better viewed in color with zoom-in.

3. Long-term Tracking with Meta-Updater

3.1. Long-term Tracking Framework

The overall framework is presented in Figure 2. In each frame, the local tracker takes the local search region as input, and outputs the bounding box of the tracked object. Then, the verifier evaluates the correctness of the current tracking result. If the output verification score is larger than a predefined threshold, the tracker will continue to conduct local tracking in the next frame. If the score is smaller than the threshold, we use the faster R-CNN detector [4] to detect all possible candidates in the next frame and crop the local search region regarding each candidate. Then, a SiamPRN model [51] takes each region as input and outputs corresponding candidate boxes. These bounding boxes are sent to the verifier for identifying whether there exists the target or not. When the verifier finds the target, the local tracker will be reset to adapt to the current target appearance. Before entering into the next frame, all historic information is collected and sent into the proposed meta-updater. Finally, the meta-updater guides the online trackers' update.

In this work, we implement an improved ATOM tracker (denoted as ATOM*) as our local tracker, which applies the classification branch of the ATOM method [9] for localization and exploits the SiamMask method [45] for scale estimation[2]. We use the RTMDNet method [21] as our verifier, and its verification threshold is set to 0.

Strength and Imperfection. Compared with recent top-ranked long-term trackers (such as MBMD [51] and SPLT [48]), the major strength of our framework lies in embedding an online-updated local tracker into the long-term tracking framework. This idea makes the long-term tracking solution benefit from the progress of short-term trackers, and unifies the short-term and long-term tracking problems as much as possible. One imperfection is that the risk of online update is amplified due to the long-term uncertain observations (since the results of any frame except for the first one have no absolute accuracy during tracking). Thus, we propose a novel **Meta-Updater** to handle this problem and obtain more robust tracking performance.

3.2. Meta-Updater

It is essential to update the tracker for capturing appearance variations from both target and its surrounding background. However, the inappropriate update will inevitably make the tracker degrade and cause tracking drift. To address this dilemma, we attempt to answer an important but unsolved question: ***Is the tracker ready for updating in the current frame?*** To be specific, we propose a **Meta-Updater** to determine whether the tracker should be updated or not in the present moment, by integrating historical tracking results. These historical results include geometric, discriminative, and appearance cues in a sequential manner. We introduce our meta-updater on the basis of an online tracker outputting a response map in each frame (e.g., ECO [8], ATOM [9]). It is easy to generalize our meta-updater for other types of trackers (such as MDNet [37]).

3.2.1 Sequential Information for Meta-Updater

Given an online tracker $\mathcal{T}$, in the t-th frame, we denote the output response map as $\mathbf{R}_t$, the output bounding box as $\mathbf{b}_t$, and the result image (cropped according to $\mathbf{b}_t$) as $\mathbf{I}_t$, respectively. The target template in the first frame is denoted as $\mathbf{I}_0$. An intuitive explanation is illustrated in Figure 3.

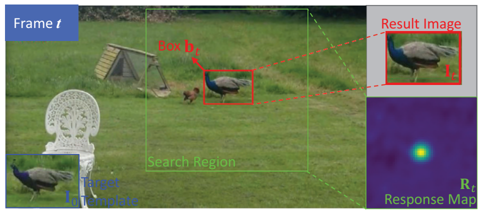


Figure 3. Intuitive explanations of some notions in this work.

We develop our meta-updater by mining the sequential

[2]In the original ATOM method [9], the scale estimation is conducted via an offline trained instance-aware IoUNet [20]. In practice, we have found the SiamMask method [45] can provide a more accurate scale estimation partly due to the strong supervision of pixel-wise annotations.

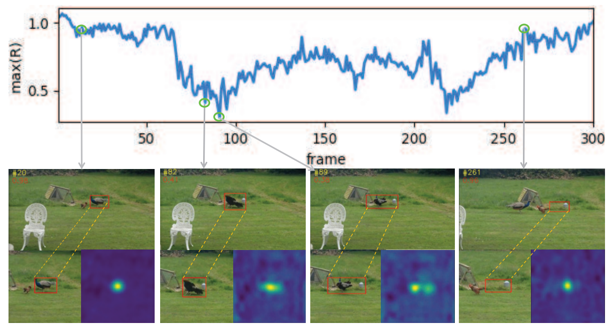


Figure 4. Illustration of varied confidence scores with representative frames. Better viewed in color with zoom-in.

information, integrating geometric, discriminative, and appearance cues within a given time slice.

Geometric Cue. In the t-th frame, the tracker outputs a bounding box $\mathbf{b}_t = [x_t, y_t, w_t, h_t]$ as the tracking state, where (x, y) denote the horizontal and vertical coordinates of the up-left corner and (w, h) are the width and height of the target. This bounding box itself merely reflects the geometric shape of the tracked object in the current frame. However, a series of bounding boxes from consecutive frames contain the important motion information regarding the target, such as velocity, acceleration, and scale change.

Discriminative Cue. Visual tracking can be considered as a classification task to distinguish the target from its surrounding background, thus, an online tracker should have good discriminative ability itself. We define a confidence score s_t^C as the maximum value of the response map $\mathbf{R}_t$ (1). For some trackers that do not output any response map (e.g., MDNet [37]), it is also not difficult to obtain this confidence score based on the classification probability or margin.

$$s_t^C = \max\left(\mathbf{R}_t\right). \tag{1}$$

Figure 4 indicates that the confidence score is not stable during the tracking process (see 89-and 261-th frames). In this work, we also exploit a convolutional neural network (CNN) to thoroughly mine the information within the response map, and obtain a response vector $\mathbf{v}_t^R$ as

$$\mathbf{v}_t^R = f^R\left(\mathbf{R}_t; \mathbf{W}^R\right), \tag{2}$$

where $f^R(.;.)$ denotes the CNN model with the parameter $\mathbf{W}^R$. The output vector $\mathbf{v}_t^R$ implicitly encodes the reliability information of the tracker in the current frame, and is further processed by the subsequent model.

Appearance Cue. The self-evaluation of the trackers' reliability with its outputs has inevitable risks, since online updating with noisy samples often makes the response not sensitive to appearance variations. Thus, we resort to a template matching method as a vital supplement, and define an appearance score as

$$s_t^A = \left\| f^A\left(\mathbf{I}_t, \mathbf{W}^A\right) - f^A\left(\mathbf{I}_0, \mathbf{W}^A\right)\right\|_2, \tag{3}$$

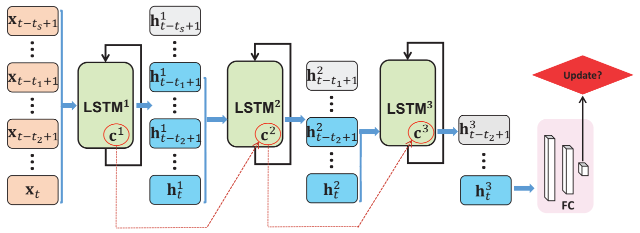


Figure 5. Proposed three-stage cascaded LSTM.

where $f^A\left(., \mathbf{W}^A\right)$ is the embedding function to embed the target and candidates into a discriminative Euclidean space, $\mathbf{W}^A$ stands for its offline trained network parameters. As presented in [33], the network $f^A\left(., \mathbf{W}^A\right)$ can be effectively trained with the combination of triplet and classification loss functions. The score s_t^A measures the distance between the tracked result $\mathbf{I}_t$ and target template $\mathbf{I}_0$. This template matching scheme is not affected by noisy observations.

Sequential Information. We integrate the aforementioned geometric, discriminative and appearance cues into a sequential matrix as $\mathbf{X}_t = [\mathbf{x}_{t-t_s+1}; ...; \mathbf{x}_{t-1}; \mathbf{x}_t] \in \mathbb{R}^{d \times t_s}$, where $\mathbf{x}_t \in \mathbb{R}^{d \times 1}$ is a column vector concentrated by s_t^C, $\mathbf{v}_t^R$, s_t^A, and $\mathbf{b}_t$. d is the dimension of concentrated cues, and t_s is a time step to balance the historical experience and current observation. This sequential information is further mined with the following cascaded LSTM scheme.

3.2.2 Cascaded LSTM

LSTM. Here, we briefly introduce the basic ideas and notions of LSTM [14] to make this paper self-contained. Its mathematical descriptions are presented as follows.

$$\begin{cases} \mathbf{f}_t = \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right) \\ \mathbf{i}_t = \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\ \mathbf{o}_t = \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\ \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right) \\ \mathbf{h}_t = \mathbf{o}_t \odot tanh\left(\mathbf{c}_t\right) \end{cases},$$

where $\sigma(.)$ denotes the element-wise sigmoid function, $tanh(.)$ stands for the element-wise tangent operation, and $\odot$ is the element-wise multiplication. $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ denote the weight matrices and bias vector requiring to be learned. The subscripts f, i, o, and c stand for the forget gate, input gate, output gate, and memory cell, respectively. Other variables are defined as follows. (a) $\mathbf{x}_t$: the input vector to the LSTM unit; (b) $\mathbf{f}_t$: the forget gate's activation vector; (c) $\mathbf{i}_t$: the input gate's activation vector; (d) $\mathbf{o}_t$: the output gate's activation vector; (e) $\mathbf{h}_t$: the hidden state vector; and (f) $\mathbf{h}_t$: the cell state vector.

Three-stage Cascaded LSTM. After obtaining the sequential features $\mathbf{X}_t$, presented in Section 3.2.1, we feed it into a three-stage cascaded LSTM model, shown in Figure 5. The time steps of three LSTMs gradually decrease to distill the sequential information and focus on the recent frames. The input-output relations are presented in Table 1. The superscript i denotes the i-th stage LSTM.

Finally, the output $\mathbf{h}_t^3$ is processed by two fully connect-

Table 1. Input-output relations of our cascaded LSTM model.

Input	$\mathbf{x}_{t-t_s+1}, ..., \mathbf{x}_{t-t_1+1}, ..., \mathbf{x}_{t-t_2+1}, ..., \mathbf{x}_t$
LSTM^1 $\rightarrow$ LSTM^2	$\mathbf{h}^1_{t-t_1+1}, ..., \mathbf{h}^1_{t-t_2+1}, ..., \mathbf{h}^1_t; \mathbf{c}^1_t$
LSTM^2 $\rightarrow$ LSTM^3	$\mathbf{h}^2_{t-t_2+1}, ..., \mathbf{h}^2_t; \mathbf{c}^2_t$
Output	$\mathbf{h}^3_t$

ed layers to generate a binary classification score, indicating whether the tracker should be updated or not.

3.2.3 Meta-Updater Training

Sample Collection. We run the local tracker on different training video sequences[3], and record the tracking results in all frames. Then, we divide these results into a series of time slices, denoted as $\mathcal{Y} = \left(\mathbf{Y}^v_t|^{t_v}_{t=t_s}\right)\Big|^V_{v=1}$. v is the video index, V is the number of training sequences, and t_v is the total frame length of the v-th video. $\mathbf{Y}^v_t = \left\{\mathbf{y}^v_{t-t_s+1}, \mathbf{y}^v_{t-t_s+2}, ..., \mathbf{y}^v_{t-1}, \mathbf{y}^v_t\right\}$, where t_s denotes the time step. Each time slice $\mathbf{y}^v_t$ includes the bounding box, response map, response score, and predicted target image in the t-th frame, along with the corresponding target template. See Section 3.2.1 for more detailed descriptions[4].

Then, we determine the label of $\mathbf{Y}^v_t$ as

$$l\left(\mathbf{Y}^v_t\right) = \left\{ \begin{array}{l} 1,\ if\ \mathbf{IoU}\left(\mathbf{b}^v_t, \mathbf{g}^v_t\right) > 0.5 \\ 0,\ if\ \mathbf{IoU}\left(\mathbf{b}^v_t, \mathbf{g}^v_t\right) = 0 \end{array} \right. , \tag{4}$$

where **IoU** stands for the Intersection-over-Union criterion. The slices whose IoUs are between 0 and 0.5 have been not adopted in the training phases to guarantee the training convergence. $\mathbf{b}^v_t$ is the output bounding box in the t-th frame in video v, and $\mathbf{g}^v_t$ is the corresponding groundtruth[5]. Equation (4) means that the label of a given time slice is determined based on whether the target is successfully located or not in the current (i.e., t-th) frame. Figure 6 visualizes some positive and negative samples for training our meta-updater.

Algorithm 1 Iterative Training Scheme

```
for $k = 0; k < K; k++$ do
    Run $\left\{\mathcal{T}, \mathcal{MU}^k(\mathcal{T})\right\}$, and record the tracking results
    Collect training samples $\mathcal{Y}^k$ with their labels $\mathcal{L}^k$
    Train the meta-updater $\mathcal{MU}^{k+1}(\mathcal{T})$
end for
```

Model Training. In this study, the local tracker and its meta-updater are tightly-coupled. The tracker affects the sample collection process for training its meta-updater. The meta-updater will change the tracker's performance, and further affect sample collection indirectly. Thus, we propose an iterative training algorithm, listed in **Algorithm 1**. The symbol $\{\mathcal{T}, \mathcal{MU}(\mathcal{T})\}$ is used to denote a local tracker equipped with its meta-updater $\mathcal{MU}(\mathcal{T})$. $\mathcal{MU}^k(\mathcal{T})$ is the learned meta-updater after the k-th iteration ($k = 0$ means no meta-updater). K is set to 3 in this work.

3.2.4 Generalization ability

The aforementioned introduction is with respect to the online-updated tracker outputting a response map. For the trackers without the response map (e.g., MDNet [37], RT-MDNet [21]), we can simply remove the subnetwork f^R, and train the meta-updater with the remaining information. For some trackers those are online updated with accumulated samples over time (such as ECO [8]), our meta-updater is able to purify the sample pool used for updating. For a given frame, if the output of the meta-updater is 0, then the current tracking results will not be added into the sample pool (i.e., not used for updating). If an ensemble of multiple online-updated trackers (such as our long-term trackers, ATOM* for local tracking and RTMDNet for verification), we can train only one meta-updater with the information from all trackers as the input, and then use it to guide all trackers' update. Section 4.3 shows our meta-updater's generalization ability for different trackers.

3.3. Implementation Details

All networks below are trained using the stochastic gradient decent optimizer, with the momentum of 0.9. The training samples are all from the LaSOT [11] training set.

Matching Network f^A. The matching network f^A adopts the ResNet-50 architecture and takes 107×107 image patches as inputs. For each target, we randomly sample bounding boxes around the groundtruth in each frame. We choose the patches with IoU above 0.7 as the positive data, and use the boxes with high confidence scores from the SiamRPN-based network [51] but not belonging to the target as the negative data. The batch size of the network f^A is 16 and we train it for 60000 iterations. The initial learning rate is 10^{-4} and divided by 10 every 200000 iterations. The matching network is individually trained and fixed when training the remaining networks of our meta-updater.

Subnetwork f^R. The input response map is first resized to 50×50, processed by two convolutional layers, and then followed by a global average pooling layer. The output is a $1 \times 1 \times 8$ vector. This subnetwork is jointly trained with the cascade LSTMs and the two fully connected layers.

LSTMs with fully connected layers. The three-stage cascaded LSTMs have 64 units in each LSTM cell. t_s, t_1 and t_2 are set to 20, 8 and 3, respectively. The forget bias is set to 1.0. The outputs are finally sent into two fully connected layers with 64 hidden units to get the final binary value. Each training stage of LSTM has a batch size of 16 and is trained by $100,000$ iterations with the learning rate of 10^{-4}.

4. Experiments

We implement our tracker using Tensorflow on a PC machine with an Intel-i9 CPU (64G RAM) and a NVIDIA

[3]For each sequence, we initialize the target in the first frame with the groundtruth, and then track it in the subsequent frames. This strictly follows the experiment setting of online single object tracking. The tracker is online updated on its own manner.

[4]The meaning of $\mathbf{y}^v_t$ is slightly different with that of $\mathbf{x}_t$ because the parameters of CNN models are also required to be trained.

[5]The training sequences have annotated groundtruth in every frame.

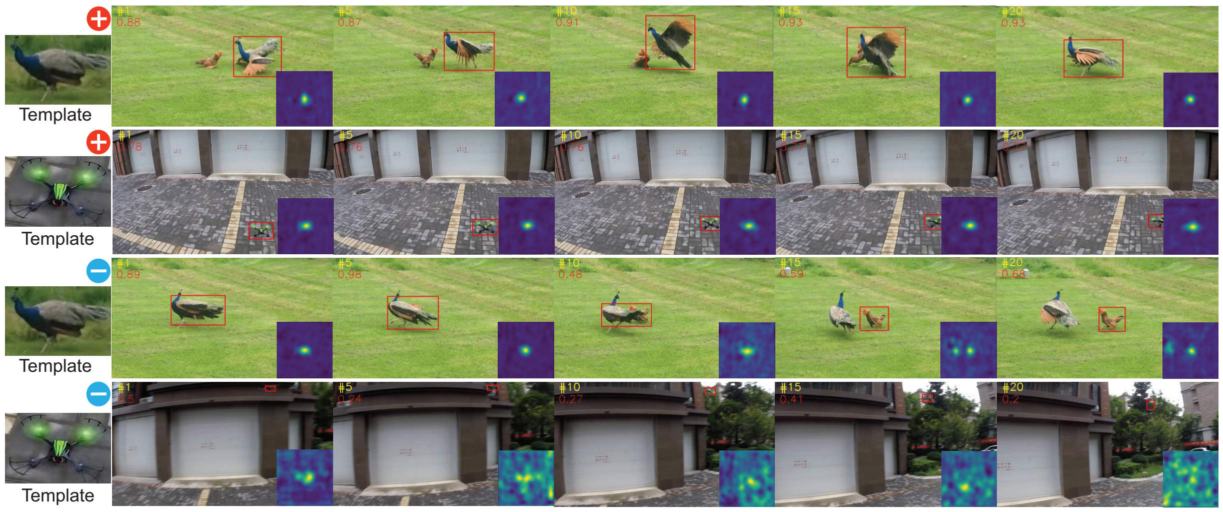


Figure 6. Illustration of positive and negative samples for meta-updater training. The first two rows illustrate two positive examples, whereas the last two rows display the negative ones. In fact, there is no interval among frames, the interval 5 is merely for clear visualization.

GTX2080Ti GPU (11G memory). The tracking speed is approximatively 13 *fps*. We evaluate our tracker on five benchmarks: VOT2018LT [23], VOT2019LT [24], OxUvALT [42], TLP [36], and LaSOT [11].

4.1. Quantitative Evaluation

Table 2. Comparisons of our tracker and 15 state-of-the-art methods on the VOT2018LT dataset [23]. The best three results are shown in **red**, **blue** and **green** colors, respectively. The trackers are ranked from top to bottom according to **F-score**.

Tracker	F-score	Pr	Re
LTMU(Ours)	**0.690**	**0.710**	**0.672**
SiamRPN++	**0.629**	**0.649**	**0.609**
SPLT	**0.616**	0.633	**0.600**
MBMD	0.610	0.634	0.588
DaSiam_LT	0.607	0.627	0.588
MMLT	0.546	0.574	0.521
LTSINT	0.536	0.566	0.510
SYT	0.509	0.520	0.499
PTAVplus	0.481	0.595	0.404
FuCoLoT	0.480	0.539	0.432
SiamVGG	0.459	0.552	0.393
SLT	0.456	0.502	0.417
SiamFC	0.433	**0.636**	0.328
SiamFCDet	0.401	0.488	0.341
HMMTxD	0.335	0.330	0.339
SAPKLTF	0.323	0.348	0.300
ASMS	0.306	0.373	0.259

VOT2018LT. We first compare our tracker with other state-of-the-art algorithms on the VOT2018LT dataset [23], which contains 35 challenging sequences of diverse objects (e.g., persons, cars, motorcycles, bicycles and animals) with the total length of 146817 frames. Each sequence contains on average 12 long-term target disappearances, each lasting on average 40 frames. The accuracy evaluation of the VOT2018LT dataset [23] mainly includes tracking precision (**Pr**), tracking recall (**Re**) and tracking **F-score**. Different trackers are ranked according to the tracking **F-score**. The detailed definitions of **Pr**, **Re** and **F-score** can be found in the VOT2018 challenge official report [23].

We compare our tracker with the VOT2018 official trackers and three recent methods (i.e., MBMD [51], SiamRPN++ [26], and SPLT [48]) and report the evaluation results in Table 2. The results show that the proposed tracker outperforms all other trackers by a very large margin.

VOT2019LT. The VOT2019LT [24] dataset, containing 50 videos with 215294 frames in total, is the most recent long-term tracking dataset. Each sequence contains on average 10 long-term target disappearances, each lasting on average 52 frames. Compared with VOT2018LT [23], VOT2019LT poses more challenges since it introduces 15 more difficult videos and some uncommon targets (e.g., boat, bull, and parachute). Its evaluation protocol is the same as that in VOT2018LT. Table 3 shows that our trackers achieves the first place on the VOT2019LT challenge.

Table 3. Performance evaluation of our tracker and eight competing algorithms on the VOT2019LT dataset. The best three results are shown in **red** , **blue** and **green** colors, respectively. The trackers are ranked from top to bottom using the **F-score** measure.

Tracker	F-score	Pr	Re
LTMU(Ours)	**0.697**	**0.721**	**0.674**
CLGS	**0.674**	**0.739**	**0.619**
SiamDW_LT	**0.665**	0.697	**0.636**
mbdet	0.567	0.609	0.530
SiamRPNsLT	0.556	**0.749**	0.443
Siamfcos-LT	0.520	0.493	0.549
CooSiam	0.508	0.482	0.537
ASINT	0.505	0.517	0.494
FuCoLoT	0.411	0.507	0.346

OxUvALT. The OxUvA long-term (denoted as OxUvALT) dataset [42] contains 366 object tracks in 337 videos, which are selected from YTBB. Each video in this dataset lasts for average 2.4 minutes, which is much longer than other commonly used short-term datasets (such as OTB2015 [46]). The targets are sparsely labeled at a frequency of 1 Hz. The dataset was divided into two disjoint subsets, *dev* and *test*. In this work, we follow the open challenge in OxUvALT, which means that trackers can use any dataset except for the YTBB validation set for training and use the OxUvALT *test* subset for testing. In the OxUvALT dataset, three criteria are adopted to evaluate different trackers, including true positive rate (**TPR**), true negative rate (**TNR**) and maximum geometric mean (**MaxGM**). **TPR** measures the fraction of present objects that are reported present as well as the location accuracy, and **TNR** gives the fraction of absent frames that are reported as absent. **MaxGM** makes a trade-off between **TPR** and **TNR** (i.e., **MaxGM** $= \max_{0\leq p\leq 1} \sqrt{((1-p)\cdot \mathbf{TPR})((1-p)\cdot \mathbf{TNR}+p)}$), which is used to rank different trackers. We compare our tracker with three recent algorithms (MBMD [51], SPLT [48] and GlobalTrack [19]) and ten algorithms reported in [42] (such as LCT [34], EBT [54], TLD [22], ECO-HC [8], BACF [13], Staple [1], MDNet [37], SINT [41], SiamFC [2], and SiamFC+R [42]). Table 4 shows that our tracker performs best in terms of **MaxGM** and **TPR** while maintaining a very competitive **TNR** value.

Table 4. Performance evaluation of our tracker and 13 competing algorithms on the OxUvALT dataset. The best three results are shown in **red**, **blue** and **green** colors, respectively. The trackers are ranked from top to bottom according to the **MaxGM** values.

Tracker	MaxGM	TPR	TNR
LTMU(Ours)	**0.751**	**0.749**	**0.754**
SPLT	**0.622**	0.498	**0.776**
GlobalTrack	**0.603**	**0.574**	0.633
MBMD	0.544	**0.609**	0.485
SiamFC+R	0.454	0.427	0.481
TLD	0.431	0.208	**0.895**
LCT	0.396	0.292	0.537
MDNet	0.343	0.472	0
SINT	0.326	0.426	0
ECO-HC	0.314	0.395	0
SiamFC	0.313	0.391	0
EBT	0.283	0.321	0
BACF	0.281	0.316	0
Staple	0.261	0.273	0

LaSOT. The LaSOT dataset [11] is one of the most recent large-scale datasets with high-quality annotations. It contains 1400 challenging sequences (1120 for training and 280 for testing) with 70 tracking categories, with an average of 2500 frames per sequence. In this work, we follow the one-pass evaluation (success and precision) to evaluate different trackers on the test set of LaSOT. Figure 7 illustrates both success and precision plots of our tracker and ten state-of-the-art algorithms, including Dimp50 [3], Dimp18 [3], GlobalTrack [19], SPLT [48], ATOM [9], SiamRPN++ [26], ECO(python) [8], StructSiam [52], DSiam [55], and MDNet [37]. Figure 7 shows that our tracker achieves the best results among all competing methods.

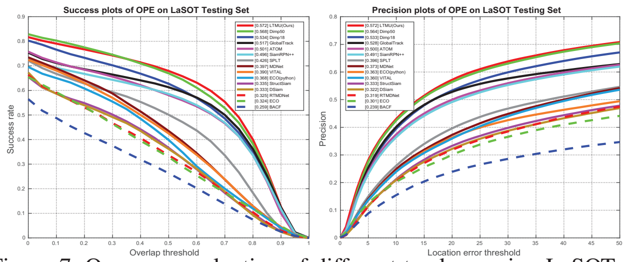


Figure 7. One-pass evaluation of different trackers using LaSOT. Better viewed in color with zoom-in.

TLP. The TLP dataset [36] contains 50 HD videos from real-world scenarios, with an average of 13500 frames per sequence. We follow the one-pass evaluation (success and precision) to evaluate different trackers on the TLP dataset. As shown in Figure 8, our tracker achieves the best results among all competing methods.

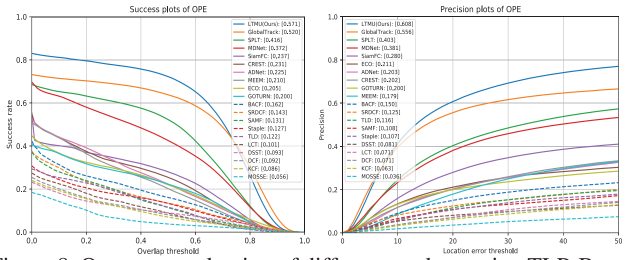


Figure 8. One-pass evaluation of different trackers using TLP. Better viewed in color with zoom-in.

4.2. Ablation Study

In this subsection, we conduct ablation analysis of our meta-updater using the LaSOT dataset [11].

Different time steps of meta-updater. First, we investigate the effects of different time steps. An appropriate time step could achieve a good trade-off between historical information and current observations. Table 5 shows that the best performance is obtained when the time step is set to 20.

Table 5. Effects of different time steps for our meta-updater.

time step	5	10	20	30	50
Success	0.553	0.564	**0.572**	0.570	0.567
Precision	0.548	0.561	**0.572**	0.569	0.565

Different inputs for our meta-updater. For our long-term trackers, the inputs of the meta-updater include bounding box (B), confidence score (C), response map (R), and appearance score (A). We verify their contributions by separately removing them from our meta-update. Detailed results are reported in Table 6, showing that each input contributes to our meta-updater (w/o means 'without').

Table 6. Effectiveness of different inputs of our meta-updater.

different input	w/o C	w/o R	w/o B	w/o A	Ours
Success	0.561	0.568	0.563	0.549	**0.572**
Precision	0.558	0.566	0.562	0.540	**0.572**

Evaluation of iterative steps. Table 7 shows that the performance is gradually improved with the increase of k.

Table 7. Evaluation of iterative steps for our cascaded LSTM.

k	0	1	2	3
Success	0.539	0.562	0.568	**0.572**
Precision	0.535	0.558	0.566	**0.572**

4.3. Discussions

Generalization ability and speed analysis. We note that our meta-updater is easy to be embedded into other trackers with online learning. To show this good generalization ability, we introduce our meta-updater into four tracking algorithms, including ATOM, ECO (the official python implementation), RTMDNet and our base tracker (using a threshold to control update). Figure 9 shows the tracking performance of different trackers without and with meta-updater on the LaSOT dataset, and it demonstrates that the proposed meta-updater can consistently improve the tracking accuracy of different trackers. Table 8 reports the running speeds of those trackers without and with the proposed meta-updater, which demonstrates that the tracking speeds decrease slightly with an additional meta-updater scheme. Thus, we can conclude that our meta-updater has a good generalization ability, which can consistently improve the tracking accuracy almost without sacrificing the efficiency.

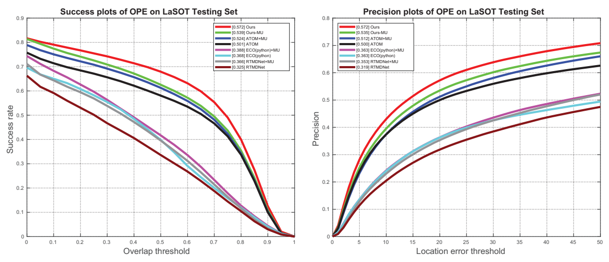


Figure 9. Generalization ability of our meta-updater (MU). Different trackers without and with meta-updater are evaluated using the LaSOT test dataset. Better viewed in color with zoom-in.

Table 8. Speed comparisons of different trackers without and with meta-updater (MU).

Trackers	ATOM	ECO	RTMDNet	Ours-MU
FPS	40	49	41	15
Trackers	ATOM+MU	ECO+MU	RTMDNet+MU	Ours
FPS	32	38	32	13

Why our meta-updater works? We run a tracker without and with its meta-updater, and record the trackers' update state ($u = 0, 1$) paired with its ground truth in each frame ($l = 0, 1$). $u = 1$ means that the tracker has been updated; otherwise, has not been updated. $l = 1$ means that the tracker can be updated; otherwise, cannot be updated. The definition of ground truth l is the same as equation (4). We have the following concepts: (1) true positive (TP): $l = 1, u = 1$; (2) false positive (FP): $l = 0, u = 1$; (3) true negative (TN): $l = 0, u = 0$; and (4) false negative (FN): $l = 1, u = 0$. Then, we can obtain the update precision (Pr), and update recall (Re) as Pr = TP/(TP+FP), and Re = TP/(TP+FN), respectively. A higher precision means that the tracker has been updated with less wrong observations. A higher recall means that the tracker more likely accepts to be updated with correct observations. We also define a true negative rate (TNR) to pay much attention to wrong observations as TNR = TN/(TN+FP). A higher TNR value means that the tracker rejects to be updated with wrong observations more strongly. Table 9 shows the statistic results of different trackers with and without their meta-updater modules. The usage of meta-updater slightly sacrifices the update recall, which means that a portion of correct observations have not been used to update the tracker in comparison with that without meta-updater. This phenomenon affects little on the trackers' performance because correct observations are all for the same target and have a large amount of redundant information. In contrast, the usage of meta-updater significantly improves the Pr and TNR values, indicating that the tracker is much less polluted by wrong observations. Thus, the risk of online update will be significantly decreased.

Table 9. Effectiveness of our meta-updater for different trackers.

Tracker	Pr	Re	TNR
RTMDNet	0.599	0.993	0.402
RTMDNet+MU	0.909	0.902	0.898
ECO	0.583	1.000	0.000
ECO+MU	0.852	0.895	0.803
ATOM	0.765	0.997	0.310
ATOM+MU	0.931	0.886	0.845
Ours-MU	0.867	0.994	0.479
Ours	0.952	0.874	0.862

5. Conclusions

This work presents a novel long-term tracking framework with the proposed meta-updater. Combined with other top-ranked trackers, our framework exploits an online-update-based tracker to conduct local tracking, which makes the long-term tracking performance benefit from the excellent short-term trackers with online update (such as ATOM). More importantly, a novel meta-updater is proposed by integrating geometric, discriminative, and appearance cues in a sequential manner to determine whether the tracker should be updated or not at the present moment. This method substantially reduces the risk of online update for long-term tracking, and effectively yet efficiently guides the tracker's update. Numerous experimental results on five recent long-term benchmarks demonstrate that our long-term tracker achieves significantly better performance than other state-of-the-art methods. The results also indicate that our meta-updater has good generalization ability.

Acknowledgement. The paper is supported in part by National Natural Science Foundation of China under Grant No. 61872056, 61771088, 61725202, U1903215, in part by the National Key RD Program of China under Grant No. 2018AAA0102001, and in part by the Fundamental Research Funds for the Central Universities under Grant No. DUT19GJ201.

References

[1] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[2] Luca Bertinetto, Jack Valmadre, Joo F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshop*, 2016.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019.

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[5] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time target-aware visual tracking. In *ICCV*, 2019.

[6] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[7] Kenan Dai, Dong Wang, Huchuan Lu, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *CVPR*, 2019.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017.

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *CVPR*, 2019.

[10] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[11] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019.

[12] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019.

[13] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017.

[14] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[15] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018.

[16] Joo F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. In *ICVS*, 2008.

[17] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. MUlti-Store Tracker (MUSTer): A cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[18] Jianglei Huang and Wengang Zhou. Re^{2}EMA: Regularized and reinitialized exponential moving average for target model update in object tracking. In *AAAI*, 2019.

[19] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GlobalTrack: A simple and strong baseline for long-term tracking. In *AAAI*, 2020.

[20] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[21] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time MDNet. In *ECCV*, 2018.

[22] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[23] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pfugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al. The sixth visual object tracking VOT2018 challenge results. In *ECCVW*, 2018.

[24] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Ondrej Drbohlav, Alan Lukežič, Amanda Berg, Abdelrahman Eldesokey, Jani Kapyla, and Gustavo Fernandez. The seventh visual object tracking VOT2019 challenge results. In *ICCVW*, 2019.

[25] Hankyeol Lee, Seokeon Choi, and Changick Kim. A memory model based on the siamese network for long-term tracking. In *ECCVW*, 2018.

[26] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[27] Bi Li, Wenxuan Xie, Wenjun Zeng, and Wenyu Liu. Learning to update for object tracking with recurrent meta-learner. *IEEE Transcations on Image Processing*, 28(7):3624–3635, 2019.

[28] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[29] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. GradNet: Gradient-guided network for visual object tracking. In *ICCV*, 2019.

[30] Peixia Li, Dong Wang, Lijun Wang, and Huchuan Lu. Deep visual tracking: Review and experimental comparison. *Pattern Recognition*, 76:323–338, 2018.

[31] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. *IEEE Transcations on Image Processing*, 24(12):5630–5644, 2015.

[32] Alan Lukei, Luka ehovin Zajc, Tom Voj, Ji Matas, and Matej Kristan. FCLT - A fully-correlational long-term tracker. In *ACCV*, 2018.

[33] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *CVPR*, 2019.

[34] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming Hsuan Yang. Long-term correlation tracking. In *CVPR*, 2015.

[35] Seyed Mojtaba Marvasti-Zadeh, Li Cheng, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep learning for visual tracking: A comprehensive survey. *CoRR*, abs/1912.00535, 2019.

[36] Abhinav Moudgil and Vineet Gandhi. Long-term visual object tracking benchmark. In *ACCV*, 2018.

[37] Hyeonseob Nam and Bohyung Han. Learning multi–domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[38] Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, 2018.

[39] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[40] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018.

[41] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[42] Jack Valmadre, Luca Bertinetto, Joao F. Henriques, Ran Tao, Andrea Vedaldi, Arnold W.M. Smeulders, Philip H.S. Torr, and Efstratios Gavves. Long-term tracking in the wild: a benchmark. In *ECCV*, 2018.

[43] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE Transcations on Image Processing*, 22(1):314–325, 2013.

[44] Mengmeng Wang, Yong Liu, and Zeyi Huang. Large margin object tracking with circulant feature maps. In *CVPR*, pages 4800–4808, 2017.

[45] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019.

[46] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[47] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Cooling-Shrinking Attack: Blinding the tracker with imperceptible noises. In *CVPR*, 2020.

[48] Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Skimming-Perusal Tracking: A framework for real-time and robust long-term tracking. In *ICCV*, 2019.

[49] Tianzhu Zhang, Si Liu, Changsheng Xu, Bin Liu, and Ming-Hsuan Yang. Correlation particle filter for visual tracking. *IEEE Transactions on Image Processing*, 27(6):2676–2687, 2018.

[50] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Learning multi-task correlation particle filters for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):365–378, 2019.

[51] Yunhua Zhang, Dong Wang, Lijun Wang, Jinqing Qi, and Huchuan Lu. Learning regression and verification networks for long-term visual tracking. *CoRR*, abs/1809.04320, 2018.

[52] Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *ECCV*, pages 355–370, 2018.

[53] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, 2019.

[54] Gao Zhu, Fatih Porikli, and Hongdong Li. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *CVPR*, 2016.

[55] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018.