Learning regression and verification networks for long-term visual tracking

Yunhua Zhang, Dong Wang, Lijun Wang, Jinqing Qi, Huchuan Lu
Dalian University of Technology
Dalian, China
zhangyunhua@mail.dlut.edu.cn, wdice@dlut.edu.cn, wlj@mail.dlut.edu.cn
{jinqin, lhchuan}@dlut.edu.cn

Abstract

In the long-term single object tracking task, the target moves out of view frequently. It is difficult to determine the presence of the target and re-search the target in the entire image. In this paper, we circumvent this issue by introducing a collaborative framework that exploits both matching mechanism and discriminative features to account for target identification and image-wide re-detection. Within the proposed collaborative framework, we develop a matching based regression module and a classification based verification module for long-term visual tracking. In the regression module, we present a regressor that conducts matching learning and copes with drastic appearance changes. In the verification module, we propose a classifier that filters out distractions efficiently. Compared to previous long-term trackers, the proposed tracker is able to track the target object more robustly in long-term sequences. Extensive experiments show that our algorithm achieves state-of-the-art results on several datasets. The code and pre-trained models are publicly available at `https://github.com/xiaobai1217/MBMD`.

1. Introduction

Visual object tracking is a fundamental problem in computer vision and can be applied in many practical systems like video surveillance, vehicle navigation, and human-machine interaction. We study the common setting that a bounding box of a target is provided in the first frame and the tracking objective is to predict locations of the target in subsequent frames.

Modern single object tracking tasks can be roughly divided into two branches. The first branch is the short-term tracking scenario, in which the target is always in the camera field of view, but not necessarily fully visible. The tracker thus just needs to report the position of a target in each frame. Another branch is the long-term scenario which not just has longer sequences, but also larger number of target disappearances. Thus, it is vital for long-term trackers to provide a confidence score of target presence and have the capability of image-wide re-detection.

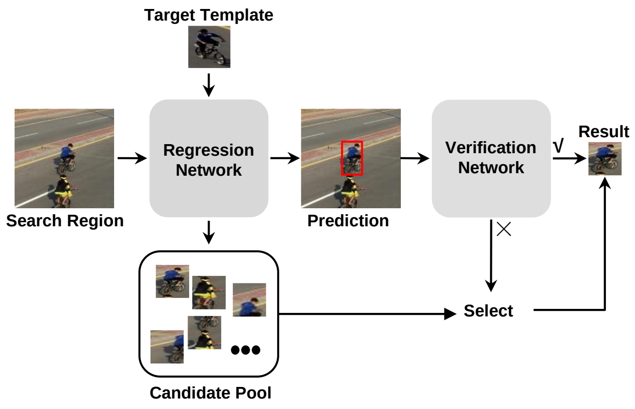

Figure 1. Overall scheme of our algorithm.

In recent years, numerous trackers [3, 5, 15, 22, 21, 10] have achieved state-of-the-art performance in short-term sequences. Only a few existing works try to tackle with long-term challenges [12, 14, 6, 8, 16]. Some long-term methods [6, 14] just find the target in a local search region and cannot re-detect the target successfully after the target moves out of view. Some others [16, 12] use a single model (matching or classification) for the whole tracking process, which is easy to drift due to online variations. There are also approaches [6, 8] that exploit ensemble models to treat short-term and long-term scenarios differently. However, according to recent experimental results reported in [13], they use hand-crafted features and fail to develop an efficient collaborative way for long-term difficulties.

To deal with challenges existed in the long-term scenario, in this work, we Learn Regression and Verification Networks off-line and on-line separately to ensemble a novel Tracking framework (LRVNT) for robust long-term tracking. LRVNT typically consists of two components: a matching based regressor denoted by $\mathcal{R}$ and a classification based verifier denoted by $\mathcal{V}$. In each frame, the regressor $\mathcal{R}$ regresses a series of candidate bounding boxes in a search region (cropped around the position of the target in the last frame), with scores measuring the similarities between candidates and the initial patch of the target in the first frame. The candidate bounding boxes that are similar to the target form a candidate pool. Then, the verifier $\mathcal{V}$ learns a classification boundary on-line to further decide whether the most similar candidate is the target or a distractor. It chooses a foreground box from the candidate pool when the most similar one is classified by it into background. As opposed to prior long-term ensemble trackers, our method does not draw a line between short-term and long-term stages. Instead, we design a solid way to determine the presence of the target and do image-wide re-detection if necessary. The judgment of target presence is deduced by the predictions from both $\mathcal{R}$ and $\mathcal{V}$. When $\mathcal{R}$ and $\mathcal{V}$ cannot find any candidate in the current search region that is similar to the target and judged as foreground as well, the tracker supposes the target is lost and then searches the target in the entire image. Unless the tracker has found one patch that is convincing for both $\mathcal{R}$ and $\mathcal{V}$, the tracker regards the target is absent in the current frame. The key idea is that as $\mathcal{R}$ learns a generic matching function off-line for tracking to robustly handle the common appearance variations an object can undergo in video sequences, $\mathcal{V}$ further equips the tracker with strong discriminative power by online learning. With the collaboration between generalized $\mathcal{R}$ and discriminative $\mathcal{V}$, the proposed tracker is capable of searching the target efficiently in long-term sequences.

The main contributions of this paper can be summarized as follows: i) We design a specific generalized regression network for long-term visual tracking to propose tight bounding boxes of candidates that are similar to the target. It learns a generic matching function for tracking, from external video data and image data, to robustly handle the common appearance variations an object can undergo in video sequences and regress the tight bounding box of any object. ii) We propose a verification mechanism, which enables our tracker to have not only metric evaluation ability but also discriminant property. It filters out distractions effectively which yields more accurate tracking results. iii) A new tracking framework for long-term scenario is proposed, which can determine whether the target is present or not and conduct an image-wide re-detection when the target is lost. Extensive evaluations performed on VOT2018 long-term benchmark show that the proposed method performs competitively in terms of tracking accuracy.

2. Related Work

Short-term tracking. With the emergence of powerful generic deep-learning representations in recent years, current state-of-the-art short-term trackers can cope well with significant appearance and motion changes and are robust to short-term occlusions. Some learn discriminative correlation filters [5, 3, 22, 21, 4] or CNN based classifiers online [24, 25, 15] to differentiate the true target from cluttered background. Some others [2, 23, 10] train generalized deep neural networks off-line to tolerate the appearance changes of the tracked object and treat object tracking as a similarity problem. However, such trackers only use a single model (discriminative or generalized) just to find a candidate patch that is most likely to be the foreground in each frame. They cannot determine whether the target is present or absent, therefore, they are not able to find the target in the entire image when the target is lost. Although some works [6, 7] have exploited the combination of both discriminative model and generalized model, they just focus on achieving high performance in the short-term scenario. Generally, short-term trackers perform poorly on long sequences.

Long-term tracking. Several existing works have been proposed for long-term tracking task. TLD [26] exploits motion information based key points matching for short-term tracking and an ensemble of classifiers for verification and rectification. MUSTer [8] utilizes a classifier for short-term classification and key points matching for long-term processing. CMT [16] only conducts key points matching during tracking. Although they are all able to search the target in the entire image, they use hand-crafted features and fail to handle complex variations during online tracking. Tow recent long-term trackers, LCT [14] and PTAV [6], can only find the target in a local search region instead of the entire image. When the target moves out of view, they are not able to detect the target any more. FCLT [12] learns correlation filter based classifiers online and gradually increases the target search range with time. It appears to be the best in [13]. However, its performance is still far from state-of-the-art.

RPN in detection and tracking. Region Proposal Network (RPN) is first proposed in Faster R-CNN [18]. The enumeration of multiple anchors [18] and sharing convolution features make the proposal extraction method time efficient while achieving high quality. RPN is capable of extracting more precise proposals due to the supervision of both foreground-background classification and bounding box regression. The improved versions of RPN, such as SSD [11] and YOLO9000 [17] are efficient detectors. Although RPN has many successful applications in detection because of its speed and great performance, it hasnt been fully exploited in tracking. SiameseRPN [10] is the first attempt to embed RPN technique into tracking and has achieved a highly competitive performance. However, it needs substantial labeled video data for training.

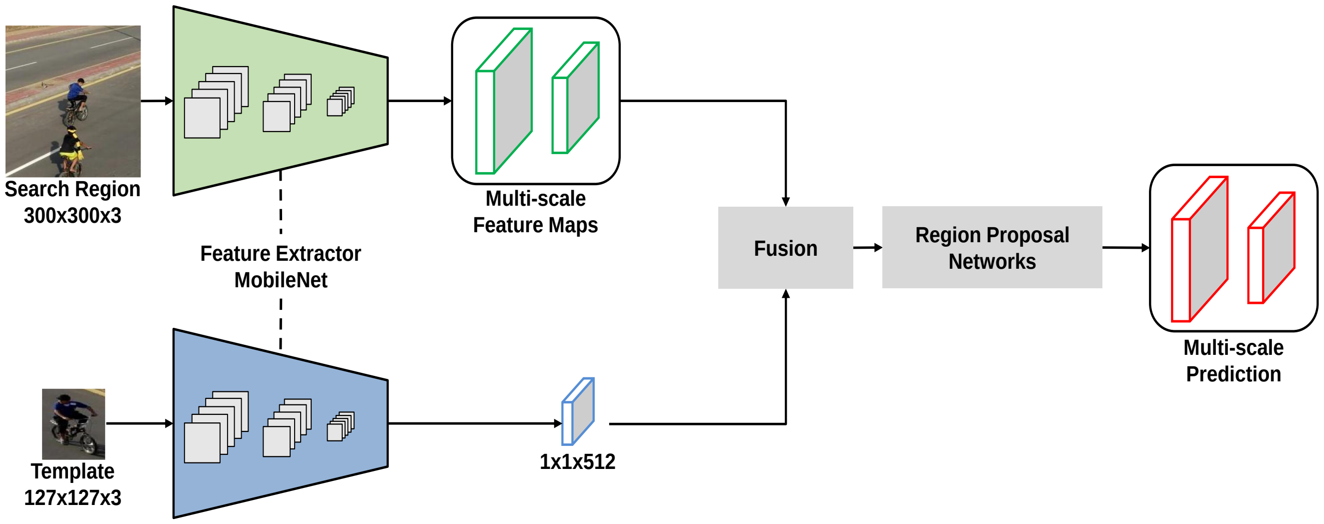

Figure 2. The pipeline of our regression network

3. Proposed Tracking Framework

3.1. Overview

In this work, we propose a collaborative framework, with regression network $\mathcal{R}$ regressing a series of bounding boxes that are similar to the target and verification network $\mathcal{V}$ verifying the selected candidate. Figure 1 overviews the scheme of our tracking algorithm. $\mathcal{R}$ is trained off-line to locate a 127×127 template image within a larger 300×300 search image. A matching and a regression functions are learned to densely compare the template image to each candidate region in the search image and regress bounding boxes of those are similar to the target. The most similar ones are collected to form a candidate pool for further verification. The candidate with the highest similarity score is first cropped out and resized to 107×107, and verified by $\mathcal{V}$. If it is classified into foreground, the proposed tracker will take it as the tracking result of the current frame. Otherwise, $\mathcal{V}$ chooses one candidate which is a foreground from the candidate pool. When both $\mathcal{R}$ and $\mathcal{V}$ cannot find any candidate box with high similarity and classification scores simultaneously, the proposed tracker regards the target is out of view and searches the target in the entire image. The details are presented in the following sections.

3.2. Regression Network

Overall pipeline. The pipeline of our regression network is shown in Figure 2. It adopts SSD [11] detection framework and MobileNets [9] as feature extractors. The two streams of the network share the same architecture but use different parameters. The upper branch takes the search region (cropped around the location of the target in the last frame) as input. It outputs two scales of feature maps, namely $19 \times 19 \times 512$ and $10 \times 10 \times 512$, to handle drastic size changes. The lower branch takes the target template (ground truth given in the first frame) as input and outputs a single $1 \times 1 \times 512$ feature vector. The feature maps of the target template and the search region are then fused and sent into the region proposal networks (RPNs). The outputs of RPNs are a series of feature maps encoding bounding box information and matching results. Non-maximum-suppression (NMS) is performed afterwards to get the candidate bounding boxes, with threshold of IoU being 0.6.

Fusion procedure. To merge with the feature maps of the search region, the single $1 \times 1 \times 512$ feature vector of the target template is duplicated to $19 \times 19 \times 512$ and $10 \times 10 \times 512$ feature maps. The obtained feature maps of the target template have the same sizes with these of the search region and are fused with them correspondingly. We take the fusion procedure of 19×19 scale for example, as shown in Figure 3. The $19 \times 19 \times 512$ feature maps of the search region are first multiplied with these of the target template, which provides similarity information and highlights the locations that are similar. The result feature maps are then concatenated with the $19 \times 19 \times 512$ feature maps of the target template to give the latter RPN information about the target. The final $19 \times 19 \times 1024$ feature maps are the inputs of the corresponding RPN. The fusion operation of the 10×10 scale is the same as that of the 19×19.

Region proposal networks. The proposed tracker has one region proposal network for each scale. Each subnetwork has two branches, one for similarity calculation and the other for proposal regression. Each branch consists of three convolutional layers with 3×3 and 1×1 kernels. If there are k anchors, for each scale, the network needs to output $2k$ channels for matching and $4k$ channels for regres-

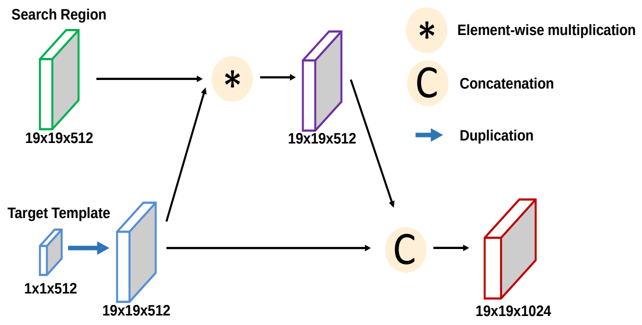

Figure 3. Illustration of the fusion procedure.

sion. When training the network with several anchors and several scales, we employ the loss function that is used in SSD [11]. Loss for matching is the cross-entropy loss and we adopt smooth L_1 loss with normalized coordinates for regression. The overall objective loss function is a sum of the matching loss and the localization loss.

3.3. Verification Network

Since $\mathcal{R}$ proposes candidates that are similar to the target template, the generated candidate pool may contain distractions. Therefore, we incorporate $\mathcal{V}$ to filter out backgrounds and reduce drifts.

The architecture of $\mathcal{V}$ is similar to that of MDNet [15]. It takes a 107×107 patch as input and outputs two neurons indicating the probabilities of foreground and background respectively. In each frame, $\mathcal{V}$ first checks the reliability of the candidate with the highest similarity score. When the most similar one is classified into background, $\mathcal{V}$ selects a foreground candidate from the candidate pool with higher similarity score than other foregrounds as the tracking result of the current frame.

As the original MDNet, we update the last three convolutional layers of the network online to train a strong softmax based classifier which can discriminate foreground from background efficiently. Through online updating, $\mathcal{V}$ helps the proposed tracker tackle with various cluttered background during tracking.

3.4. Tracking Strategy

The proposed tracker first searches the target in the search region, which is four times the size of the target. The regression network $\mathcal{R}$ proposes N_r sorted candidate bounding boxes which form the candidate pool. These candidates are similar to the target template, with similarity threshold θ_r. We set $N_r = 20$ in the implementation. Then the verification network $\mathcal{V}$ selects a foreground from the candidate pool as described above. When the tracker cannot find any candidate with scores from both $\mathcal{R}$ and $\mathcal{V}$ being higher than θ_r and θ_v respectively, the tracker regards the target has moved out of the search region.

When the tracker cannot find the target in the search region, it then searches the target in the entire image. Image-wide search begins from the top left corner of the image, and stride of each direction is half of the corresponding side of the search region. The tracker stops search after having found one candidate with credible scores or evaluated all regions of the image. Unless the proposed tracker has found one reliable candidate, it regards the target is absent in the current frame.

The confidence score of the selected candidate in each frame is estimated by both classification score S_c and similarity score S_r. The criterion can be written as below,

$$confidence = \begin{cases} 0.99, & S_r > \theta_{r'}, S_c > \theta_v \ or \ S_c > \theta_{v'} \\ NaN, & S_r < \theta_r, S_c < \theta_v \\ S_r, & otherwise \end{cases} \tag{1}$$

where θ_r is 0.3, $\theta_{r'}$ is 0.5, θ_v is 0 and $\theta_{v'}$ is 20.0. When both $\mathcal{R}$ and $\mathcal{V}$ are confident about the selected candidate or $\mathcal{V}$ is very confident, the proposed tracker regards the target is present in the current frame and outputs a confidence score of 0.99. When both $\mathcal{R}$ and $\mathcal{V}$ give negative feedbacks, the tracker regards the target has moved out of view and returns a confidence score of NaN. Otherwise, the confidence score is the similarity score from $\mathcal{R}$.

4. Experiments

4.1. Implementation Details

Our experiments are implemented using Tensorflow [1] on a PC with an Inter i7, 32G RAM, NVIDIA GTX TITAN X. The speed of our tracker is around 2fps.

Regression Network. We use MobileNet pretrained from ImageNet [20] for both branches of the feature extractor section. During the training phase, sample pairs are picked from ILSVRC [19] object localization dataset and video object localization dataset with a random interval. For image object localization dataset, we intend to train the regression network to have the capability of regressing any kind of object given the template. We choose an object as the target from an image randomly and cropped a detection region around the object. For video object localization dataset, the similarity calculation branch of the regression network learns a generic matching function for tracking to tolerate the common appearance variations. The regression network is trained in an end-to-end manner using Stochastic Gradient Descent (SGD) after the feature extractor subnetworks (MobileNets) being pre-trained using ImageNet. Because of the need of training regression branch, some data augmentations are adopted including affine transformation and random erasing [27]. We fix its parameters during online tracking. In long-term sequences, since the target moves out of view frequently and its size often has changed dramatically when it re-appears, we adopt two scales with different ratios of anchor. The anchor ratios we adopt are $[0.33, 0.5, 1, 2, 3]$.

The strategy to pick positive and negative training samples is also important in our proposed framework. The criterion used in object detection task is adopted here that we use IoU together with two thresholds th_{hi} and th_{lo} as the measurement. Positive samples are defined as the anchors which have $IoU > th_{hi}$ with their corresponding ground truth. Negative ones are defined as the anchors which satisfy $IoU < th_{lo}$. We set th_{lo} to 0.5 and th_{hi} to 0.7. We also limit at most 16 positive samples and totally 64 samples from one training pair. There are totally $500,000$ iterations performed during training phase and the batch size is 32. Each batch consists of 16 pairs from image object localization dataset and 16 pairs from video object localization dataset. We use the 10^{-2} and 10^{-3} learning rates both for $200,000$ iterations, then continue training for $100,000$ iterations with 10^{-4}. During inference phase, there is no online adaptation. The features of the target template are only extracted in the first frame and keep fixed in the whole tracking sequence.

Verification Network $\mathcal{V}$ adopts VGGM pretrained from ImageNet [20] with the parameters of the first three convolution layers fixed and fine-tunes the last three convolution layers online. We train the forth and fifth layers with the the learning rate 10^{-3} and the last layer with 10^{-2}. We crop out 500 positive samples and 5000 negative samples to train 30 times during initialization, 50 positive samples and 200 negative samples for the 15 iterations during update. We define the score of the first frame as the initial score. If the current classification score is higher than half of the initial score and the similarity score from $\mathcal{R}$ is higher than $\theta_{r'}$, we will storage the current frame as reliable training samples, the max size of which is 5.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin. Tensorflow: Large scale machine learning on heterogeneous distributed systems. In *arXiv preprint arXiv:1603.04467*, 2016.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshop*, 2016.

[3] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

[4] M. Danelljan, G. Hger, F. S. Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshop*, 2015.

[5] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[6] H. Fan and H. Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *ICCV*, 2017.

[7] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018.

[8] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*, 2015.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[10] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[12] A. Lukei, L. . Zajc, T. Voj, J. Matas, and M. Kristan. Fclt - a fully-correlational long-term tracker. In *arXiv preprint arXiv:1711.09594*, 2017.

[13] A. Lukei, L. . Zajc, T. Voj, J. Matas, and M. Kristan. Now you see me: evaluating performance in long-term visual tracking. In *ECCV*, 2018.

[14] C. Ma, X. Yang, C. Zhang, and M. H. Yang. Long-term correlation tracking. In *CVPR*, 2015.

[15] H. Nam and B. Han. Learning multi–domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[16] G. Nebehay and R. Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *CVPR*, 2015.

[17] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster r–cnn: towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2014.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] C. Sun, D. Wang, H. Lu, and M. H. Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018.

[22] C. Sun, D. Wang, H. Lu, and M. H. Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018.

[23] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[24] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[25] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016.

[26] K. Z, M. K, and M. J. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[27] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *arXiv preprint arXiv:1708.04896*, 2017.