'Skimming-Perusal' Tracking: A Framework for Real-Time and Robust Long-term Tracking

Bin Yan[1†], Haojie Zhao[1†], Dong Wang[1*], Huchuan Lu[1] and Xiaoyun Yang[2]
[1]School of Information and Communication Engineering, Dalian University of Technology, China
[2]China Science IntelliCloud Technology Co., Ltd., † Equal Contribution
{yan_bin, haojie_zhao}@mail.dlut.edu.cn, {wdice, lhchuan}@dlut.edu.cn, xiaoyun.yang@intellicloud.ai

Abstract

Compared with traditional short-term tracking, long-term tracking poses more challenges and is much closer to realistic applications. However, few works have been done and their performance have also been limited. In this work, we present a novel robust and real-time long-term tracking framework based on the proposed skimming and perusal modules. The perusal module consists of an effective bounding box regressor to generate a series of candidate proposals and a robust target verifier to infer the optimal candidate with its confidence score. Based on this score, our tracker determines whether the tracked object being present or absent, and then chooses the tracking strategies of local search or global search respectively in the next frame. To speed up the image-wide global search, a novel skimming module is designed to efficiently choose the most possible regions from a large number of sliding windows. Numerous experimental results on the VOT-2018 long-term and OxUvA long-term benchmarks demonstrate that the proposed method achieves the best performance and runs in real-time. The source codes are available at https://github.com/iiau-tracker/SPLT.

1. Introduction

Online visual tracking is one of most important problems in computer vision, and has many practical applications including video surveillance, behavior analysis, visual navigation, augmented reality and so on. It is a very tough task to design a robust and efficient tracker caused by the challenges from both foreground and background variations. These challenges include occlusion, illumination variation, viewpoint change, rotation, motion blur, to name a few. Due to the breakthrough of deep learning and the construction of large-scale benchmarks, numerous trackers [5, 25, 31, 30, 18, 41, 14, 20, 6, 34, 17, 4, 8, 19, 36, 37] have recently achieved very promising performance.

However, most of existing trackers and datasets focus on the short-term tracking task, the setting of which is that the tracked object is almost always in the camera filed of view but not necessarily fully visible. There still exists a big gap between the short-term tracking setting and the realistic tracking applications. In recent, some researchers have focused on the long-term tracking task and attempted to create related large-scale benchmarks (such as VOT2018LT [23] and OxUvA [33]).

Compared with short-term tracking, the long-term tracking task additionally requires the tracker having the capability to capture the tracked object in long-term videos and to handle the frequent target disappearance and reappearance. Thus, it poses more challenges than short-term tracking mainly from two aspects. First, the frame length in long-term datasets is much greater than that in short-term scenarios. For examples, the average numbers of frame length in the VOT2018LT [23], OxUvA [33], OTB2015 [35] and VOT2018 [16] benchmarks are 4196, 4235, 590 and 350, respectively (the former two are long-term datasets and the latter two are well-known short-term datasets). Second, there exist a large number of *absent* labels in long-term tracking datasets. Thus, it is critical for long-term trackers to capture the tracked object in long-term sequences, determine whether the target is *present* or *absent*, and have the capability of image-wide re-detection.

Until now, some long-term trackers have been developed based on hand-crafted features, including TLD [15], LCT [24], FCLT [22], MUSTer [11], CMT [26] and EBT [40]. Although these methods obtain some achievements on a small number of long-term videos, they cannot achieve satisfactory performance on recent long-term benchmarks (see experiments in [16, 33]). Recently, some deep-learning-based algorithms [7, 16, 38] are proposed for long-term tracking and significantly improved the tracking performance. But there still lacks of a robust and real-time framework to address the long-term tracking task.

In this work, we propose a novel 'Skimming-Perusal'

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

tracking framework for long-term tracking. The perusal module aims to precisely capture the tracked object in a local search region; while the skimming module focuses on efficiently selecting the most possible candidate regions and significantly speeding up the image-wide re-detection process. Our main contributions can be summarized as follows.

- *A novel 'Skimming-Perusal' framework based on deep networks is proposed to address the long-term tracking task. Both skimming and persual modules are offline trained and directly used during the tracking process. Our framework is simple yet effective, which could be served as a new baseline for long-term tracking.*
- *A novel perusal module is developed to precisely capture the tracked object in a local search region, which is comprised of an effective bounding box regressor based on SiameseRPN and a robust offline-trained verifier based on deep feature embedding.*
- *A novel skimming module is designed to efficiently select the most possible local regions from densely sampled sliding windows, which could speed up the image-wide re-detection process when the target is absent.*
- *Numerous experimental results on the VOT2018LT and OxUvA datasets show that our tracker achieves the best accuracies with real-time performance.*

2. Related Work

Traditional Long-term Tracking. In [15], Kalal *et al.* propose a tracking-learning-detection (TLD) algorithm for long-term tracking, which exploits an optical-flow-based matcher for local search and an ensemble of weak classifiers for global re-detection. Following the idea of TLD, Ma *et al.* [24] develop a long-term correlation tracker (LCT) using a KCF method as a local tracker and a random ferns classifier as a detector. The fully correlational long-term tracker (FCLT) [22] maintains several correlation filters trained on different time scales as a detector and exploits the correlation response to guide the dynamic interaction between the short-term tracker and long-term detector.

Besides, some researchers have addressed the long-term tracking task using the keypoint matching or global proposal scheme. The CMT [26] method utilizes a keypoint-based model to conduct long-term tracking, and the MUSTer [11] tracker exploits an integrated correlation filter for short-term localization and a keypoint-based matcher for long-term tracking. But the keypoint extractors and descriptors are often not stable in complicated scenes. In [40], Zhu *et al.* develop an EdgeBox Tracking (EBT) method to generate a series of candidate proposals using EdgeBox [42] and verify these proposals using structured SVM [10] with multi-scale color histograms. However, the edge-based object proposal is inevitably susceptible to illumination variation and motion blur. The above-mentioned trackers have attempted to address long-term tracking from different perspectives, but their performance are not satisfactory since they merely exploit hand-crafted low-level features. In this work, we develop a simple yet effective long-term tracking framework based on deep learning, whose goal is to achieve high accuracy with real-time performance.

Deep Long-term Tracking. Recently, some researchers have attempted to exploit deep-learning-based models for long-term tracking. Fan *et al.* [7] propose a parallel tracking and verifying (PTAV) framework, which effectively integrates a real-time tracker and a high accurate verifier for robust tracking. The PTAV method performs much better than other compared trackers on the UAV20L dataset. Valmadre *et al.* [33] implement a long-term tracker, named as SiamFC+R. This method equips SiamFC [3] with a simple re-detection scheme, and finds the tracked object within a random search region when the maximum score of the SiamFC's response is lower than a given threshold. The experimental results demonstrate that the SiamFC+R tracker achieves significantly better performance than the original SiamFC method on the OxUvA dataset. But the SiamFC's score map is not always reliable, which limits the performance of the SiamFC+R tracker.

In [38], Zhang *et al.* combine an offline-trained bounding box regression network and an online-learned verification network to design a long-term tracking framework. The regression network determines the bounding boxes of the tracked object in a search region; while the verification network verifies whether the tracking result is reliable or not and makes the tracker dynamically switch between local search and global search states. The global search explicitly conducts image-wide re-detection by exploiting a sliding window strategy in the entire image. This method provides a complete long-term tracking framework, and also is the winner of the VOT2018 long-term challenge. However, the adopted straightforward sliding window strategy and online-learned verification model make it be very slow and far from the real-time applications (merely 2.7fps reported in [38] and 4.4fps in our experiment setting). In [16], Zhu *et al.* propose a DaSiam_LT method for long-term tracking, which extends the original SiameseRPN tracker [18] by introducing a local-to-global search region strategy. During the tracking process, the size of the search region will be iteratively increasing when the tracking failure is indicated. The distractor-aware training and inference [41] make the output tracking scores suitable to determine whether the tracker fails or not. The DaSiam_LT tracker achieves the second best performance in the VOT2018 long-term challenge, however, it requires very large numbers of image sequences for offline training. Besides, the DaSiam_LT method does not explicitly include the image-wide re-detection scheme, thereby failing in the re-detection experiment. Compared with aforementioned methods, our goal is to develop a simple but effective long-term tracking framework with high accuracy and real-time performance.

3. ‘Skimming-Perusal’ Tracking Framework

In this work, we propose a novel ‘Skimming-Perusal’ framework to address the long-term tracking problem. There exist two fundamental modules: skimming and perusal. The perusal module aims to conduct robust object regression and verification in a local search region; while the skimming module focuses on quickly selecting the most possible candidate regions within a large number of sliding windows when the tracker runs in the global search state.

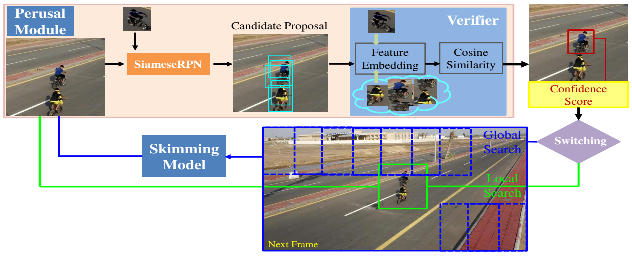


Figure 1. Our ‘Skimming-Perusal’ long-term tracking framework. Better viewed in color with zoom-in.

The overall framework is presented in Figure 1. Our tracker first searches the target in a local search region (four times of the target size) using the perusal module. After obtaining the best candidate in each frame, our tracker treats the tracked object as *present* or *absent* based on its confidence score and then determines the search state (local search or global search) in the next frame. If the confidence score is higher than a pre-defined threshold, the tracker treats the target as *present* and continues to track the target in the local search region centered by the object location. Otherwise, the tracker regards the target as *absent* and conducts global search in the next frame. To be specific, our global search scheme crops a series of local search regions using sliding windows and then deals with these regions using the perusal module. To speed it up, we develop a novel skimming module to efficiently select the most likely local regions and effectively handle these regions using the perusal module. The detailed descriptions of our skimming and perusal modules are presented as follows.

3.1. Robust Local Perusal with Offline-learned Regression and Verification Networks

Our perusal module is composed of an offline-learned SiameseRPN model and an offline-learned verification model (shown in Figure 1). The former one generates a series of candidate proposals within a local search region, and the latter one verifies them and determines the best candidate.

SiameseRPN. The SiameseRPN method [18] improves the the classical SiamFC method [3] by introducing a region proposal network, which allows the tracker to estimate the bounding box of variable aspect ratio effectively. In this work, we choose SiameseRPN as our basic regressor due to its robustness and efficiency. To be specific, we adopt a variant of the original SiameseRPN method proposed in [38] since its training and testing codes are all publicly available. The flowchart of the adopted SiameseRPN model is illustrated in Figure 2. Given a target template $\mathcal{Z}$ and a local search region $\mathcal{X}$, the SiameseRPN model generates a set of bounding boxes $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_N]$ with their corresponding similarity scores $\mathbf{s} = [s_1, s_2, ..., s_N]$ (N denotes the total number of candidates).

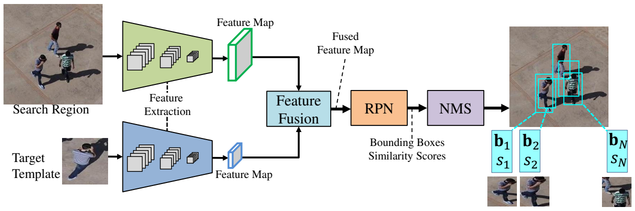


Figure 2. The adopted SiameseRPN module in our framework. Better viewed in color with zoom-in.

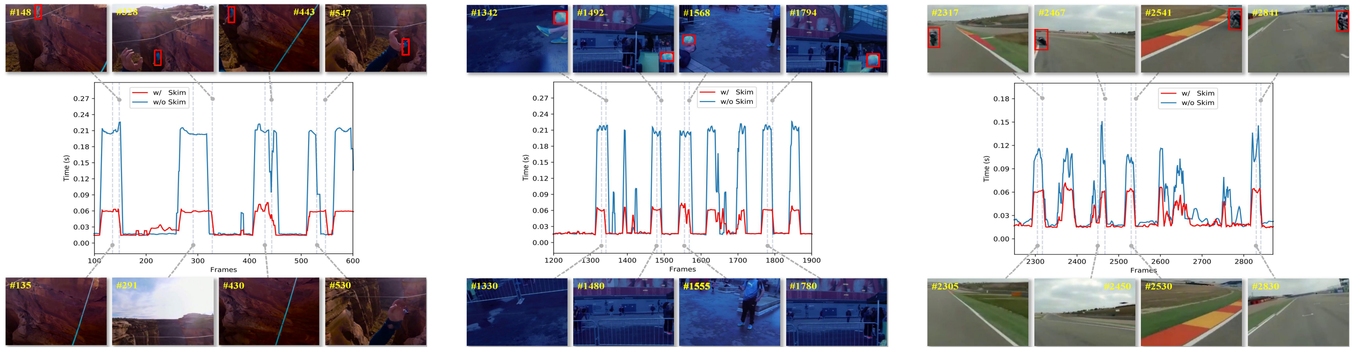
Simply, we can directly locate the tracked object based on the bounding box with the highest score in the current frame (i.e., obtain the optimal bounding box as $\mathbf{b}_{i^*}, i^* = \arg\max_i \{s_i\}$. However, this manner makes the tracker unstable and easily drift to some distractors. This is mainly attributed to the multi-task learning manner in SiameseRPN. The joint learning bounding box predictions and classification scores usually generate accurate box proposals but unreliable scores for the tracking task. Thus, we merely exploit SiameseRPN to generate candidate proposals, and then infer their confidence score using an additional verifier.

Offline-learned Verification Network. To ensure the efficiency of our tracking framework, we attempt to exploit an offline-trained verifier based on deep feature embedding [29]. Specifically, we learn an embedding function $f(.)$ to embed the target template and the candidate proposals into a discriminative Euclidean space. The discriminative ability is ensured by the following triplet loss,

$$\sum_{i=1}^{M} \left[\| f(\mathcal{Y}_i^a) - f(\mathcal{Y}_i^p) \|_2^2 - \| f(\mathcal{Y}_i^a) - f(\mathcal{Y}_i^n) \|_2^2 + \alpha \right]_+, \tag{1}$$

where $\mathcal{Y}_i^a$ denotes the i-th anchor of a specific target, $\mathcal{Y}_i^p$ is a positive sample (i.e., one of other images of the target), and $\mathcal{Y}_i^n$ is a negative sample of any other target or background. α is a margin value (simply set to 0.2 in this work). $\mathcal{T}$ is the set of all possible triplet pairs in the training set and has cardinality M. The construction of training triplets and the hyper-parameters are presented in Section 3.3.

During tracking, we can determine the confidence scores $[c_1, c_2, ..., c_N]$ of the candidate proposals $[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_N]$



(a) tightrope (b) freestyle (c) yamaha

Figure 3. Illustration of the effectiveness of the proposed skimming module. The abbreviations 'w/ skim' and 'w/o skim' denote our trackers with and without the skimming module, respectively.

using a thresholding cosine similarity metric as

$$c_i = \max\left(\frac{f^\top(\mathcal{Z})\, f(\phi(\mathbf{b}_i))}{\|f(\mathcal{Z})\|_2 \|f(\phi(\mathbf{b}_i))\|_2}, 0\right), \qquad (2)$$

where $\mathcal{Z}$ is the target template in the first frame and $\phi(\mathbf{b}_i)$ denotes the image cropped within the i-th bounding box $\mathbf{b}_i$.

After that, the optimal candidate can be determined as $\mathbf{b}_{i^*}$ ($i^* = \arg\max_i \{c_i\}$), whose corresponding confidence score is c_{i^*}. Finally, we exploit a threshold-based switching strategy to make interactions between local search and global search dynamically. If the score c_{i^*} is larger than a pre-defined threshold θ, our tracker treats the target being *present* and continues to conduct local search in the next frame. Otherwise, it considers the tracked object as *absent* and then invokes the global search scheme in the next frame. θ is set to 0.65 in this work and discussed in Section 4.2.

Cascaded Training. To enhance the performance of our perusal module, we exploit a cascaded training strategy to make our verifier more robust. First, we train the SiameseRPN and verification network individually. Then, we apply the perusal module to the training set and collect the mis-classified samples as hard examples. Finally, we fine-tune the verification network using the collected hard examples. The training details are presented in Section 3.3.

3.2. Efficient Global Search with Offline-learned Skimming Module

The long-term tracker usually combines a local tracker and a global re-detector, and invokes the global re-detector when the object is *absent*. The sidling window technique is widely used to conduct global search [24, 38], by which the local region in each window is utilized to determine whether the object is *present* or not. However, this manner is very time-consuming especially when deep-learning-based models are used. For example, the recent MBMD tracker [38] (the winner of VOT2018 long-term challenge) merely runs less than 5fps (very far from real-time performance).

To address this issue, we propose a skimming module to conduct fast global search by efficiently selecting the most possible candidate regions from a large number of sliding windows. Figure 3 demonstrates some representative examples, from which we can see that our skimming scheme could significantly reduce the running time when the tracker conducts image-wide re-detection (i.e., the time interval between target disappearance and reappearance).

Given a target template $\mathcal{Z}$ and a search region $\mathcal{X}$, the skimming module aims to learn a function $p = g(\mathcal{Z}, \mathcal{X})$, where p indicates whether the target appears in this region or not. The function $g(.,.)$ is implemented using deep convolutional neural networks (CNN), whose network architecture is presented in Figure 4. Both target template and search region are fed into CNN feature extractors, and then their feature maps are fused and concatenated into a long vector. Finally, the fully connected (FC) layer with the sigmoid function is added to conduct a binary classification. The cross entropy loss is adopted to train this network.

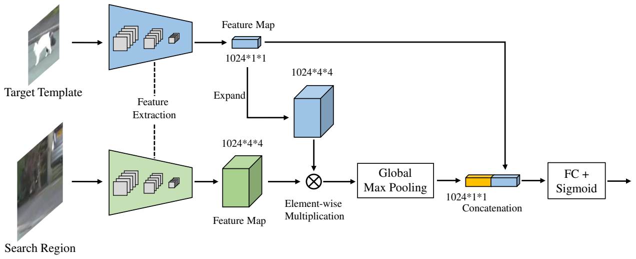


Figure 4. The network architecture of our skimming module. Better viewed in color with zoom-in.

When the tracker runs on global search, a series of sliding windows are densely sampled. We first apply our skimming module on these regions, and then select the top-K candidates based on their classification margins and discard the remaining ones as distractors. Only selected regions will be further handled using the perusal module (SiameseRPN+Verifier), which makes our tracker very efficient in image-wide re-detection. In addition, our skimming module could improve the tracker's robustness since it filters out some distractors and alleviates tracking drift. The parameter K is set as 3 in this work.

3.3. Implementation Details

In this subsection, we present some key implementation details due to space limitation. The detailed parameter settings can be found in our source codes.

Network Architectures and Training Dataset: In this work, our regression and skimming models have similar structures. First, we use MobileNet(V1) [12] as our feature extractor. Second, we downsample the spatial resolution of the template feature to 1×1 using average pooling. Besides, the Siamese branches do not share parameters. For the verification model, we adopt ResNet50 as the backbone of our verifier. The resolution of the template image and candidate proposals is resized to 127×127, and the resolution of the search regions is resized to 300×300. The parameters of aforementioned networks are initialized with the ImageNet pre-trained models and then fine-tuned on the ImageNet DET and VID datasets [28].

Training Data Preparation: In order to make our regression module obtain capability to regress any kind of object for a given object template as well as learn a generic matching function for tracking to tolerate the common appearance variations, we combine the ImageNet DET and VID datasets, and introduce some data augmentations like horizontally flipping and random erasing [39]. For the verification network, theoretically we should choose hard examples as triplet pairs to speed up convergence and boost the discriminative power. To achieve this goal, we adopt the following sampling strategy: (1) randomly choosing one video from training set and picking its initial target patch as the anchor; (2) stochastically choosing one frame within this video as the positive one; and (3) randomly choosing another frame from the video belonging to a different object class as the negative one. Besides, we exploit the cascaded training strategy (see Section 3.1) to further mine hard samples and fine-tune the verification network. For the skimming network, we randomly crop search regions around the target as positive samples. In order to obtain negative samples more conveniently, we directly mask the target with mean pixel value, then crop negative samples using the same strategy as cropping positive ones. The template generation manner is consistent with the SiamFC method [3].

Training Strategy: The regression, verification and skimming modules are trained independently in an end-to-end fashion using the batch gradient descent optimizer with the momentum of 0.9. The batch sizes of regression and skimming networks are set to 32, and the batch size of the verification network is chosen as 64. We train regression and skimming networks for 500,000 iterations and 20 epochs with the same learning rate of $1e^{-3}$, respectively. For the verification network, we train it for 70 epochs and exploit 60,000 triplet pairs in each epoch. The learning rate is initially set to $1e^{-2}$ and gradually decayed to its $1/10$ every 20 epochs.

4. Experiments

In this work, we implement our tracker using Python with the Tensorflow [1] and Keras deep learning libraries. The proposed method is tested on a PC machine with an Inter i7 CPU (32G RAM) and a NVIDIA GTX1080Ti GPU (11G memory), which runs in real-time with **25.7** frames per second (fps). Our tracker is denoted as **SPLT**. Both training and testing codes are available at https://github.com/iiau-tracker/SPLT.

We compare our tracker with other competing algorithms on the VOT-2018 long-term (VOT2018LT) dataset [23] and OxUvA long-term dataset [33]. The quantitative evaluations and ablation studies are reported as follows.

4.1. Results on VOT2018LT

The VOT2018LT [23] dataset is first presented in Visual Object Tracking (VOT) challenge 2018 to evaluate the performance of different long-term trackers. This dataset includes 35 sequences of various objects (e.g., persons, cars, bicycles and animals), with the total frame length of 146847 frames and the resolution ranges between 1280×720 and 290×217. The construction of this dataset fully takes object disappearance into account, where each sequence contains on average 12 long-term object disappearances and each disappearance lasts on average 40 frames. The ground truths of the tracked objects are annotated by bounding boxes, and sequences are annotated by nine visual attributes (including full occlusion, out-of-view, partial occlusion, camera motion, fast motion, scale change, aspect ratio change, viewpoint change and similar objects).

The evaluation protocol of the VOT2018LT [23] dataset includes two aspects: accuracy evaluation and re-detection evaluation. First, the accuracy evaluation measures the performance of a given long-term tracker using tracking precision (**Pr**), tracking recall (**Re**) and tracking F-measure. The F-measure criterion is defined based on equation (3).

$$\mathbf{F}(\tau_\theta) = 2\mathbf{Pr}(\tau_\theta)\mathbf{Re}(\tau_\theta)/(\mathbf{Pr}(\tau_\theta) + \mathbf{Re}(\tau_\theta)), \quad (3)$$

where τ_θ is a given threshold. $\mathbf{Pr}(\tau_\theta)$, $\mathbf{Re}(\tau_\theta)$ and $\mathbf{F}(\tau_\theta)$ denote the thresholding precision, recall and F-measure, respectively. Thus, long-term tracking performance can be visualized by the tracking precision, recall, F-measure plots by computing these scores for all thresholds τ_θ. In [23], the **F-score** is defined as the highest score on the F-measure plot (i.e., taken at the tracker-specific optimal threshold), which acts as the primary role for ranking different tracker-

Table 1. Comparison of our tracker and 15 competing algorithms on the VOT2018LT dataset [23]. The best three results are marked in **red**, **blue** and **green** bold fonts respectively. The trackers are ranked from top to bottom using the **F-score** measure.

Tracker	F-score	Pr	Re	Frames (Success)
SPLT(Ours)	**0.616**	**0.633**	**0.600**	1 (100%)
MBMD	**0.610**	**0.634**	**0.588**	1 (100%)
DaSiam_LT	**0.607**	0.627	**0.588**	- (0%)
MMLT	0.546	0.574	**0.521**	0 (100%)
LTSINT	0.536	0.566	0.510	2 (100%)
SYT	0.509	0.520	0.499	0 (43%)
PTAVplus	0.481	0.595	0.404	0 (11%)
FuCoLoT	0.480	0.539	0.432	78 (97%)
SiamVGG	0.459	0.552	0.393	- (0%)
SLT	0.456	0.502	0.417	0 (100%)
SiamFC	0.433	**0.636**	0.328	- (0%)
SiamFCDet	0.401	0.488	0.341	0 (83%)
HMMTxD	0.335	0.330	0.339	3 (91%)
SAPKLTF	0.323	0.348	0.300	- (0%)
ASMS	0.306	0.373	0.259	- (0%)
FoT	0.119	0.298	0.074	- (6%)

s. Second, the re-detection evaluation aims to test the tracker's re-detection capability based on two criteria including the average number of frames required for re-detection (**Frames**) and the percentage of sequences with successful re-detection (**Success**). The detailed evaluation protocol can be found in the VOT2018 official report and toolbox [23].

Table 1 summarizes the comparison results of different tracking algorithms, from which we can see that our tracker achieves the best performance in terms of **F-score** and **Re** criteria while maintaining the highest re-detection success rate. The average precision-recall curves of our tracker and other competing ones are presented in Figure 5. Besides the proposed tracker, the MBMD and DaSiam_LT methods also achieve top-ranked performance.

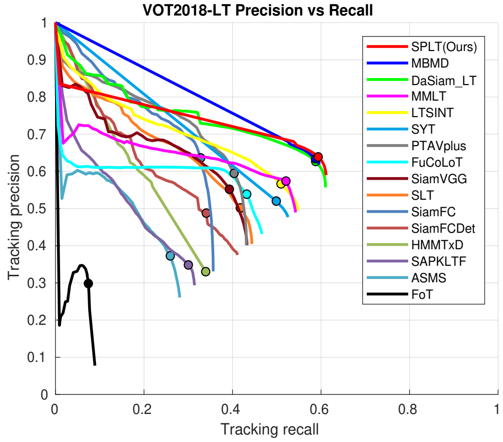


Figure 5. Average precision-recall curves of our tracker and other state-of-the-art methods on the VOT2018LT benchmark [23]. Different trackers are ranked based on the maximum of the F-Score.

Ours *vs* **MBMD**: Compared with the MBMD method (the VOT2018 long-term challenge winner), our tracker achieves better performance in terms of accuracy and keeps the same re-detection capability. The most important advantage is that our tracker runs much faster than MBMD (25.7fps for ours and 4.4fps for MBMD in our experiment platform). This is mainly attributed to the proposed skimming module and the offline-trained verification module. Thus, we believe that our tracker could act as a new baseline algorithm in the VOT2018LT benchmark.

Ours *vs* **DaSiam_LT**: Compared with DaSiam_LT (the second best method in the VOT2018 long-term task), the proposed tracking algorithm has three major strengths. First, our tracker performs better than DaSiam_LT in terms of all three accuracy criteria. Second, our tracker has successfully passed the re-detection experiment (i.e., achieves a success rate of 100%), while the DaSiam_LT method completely fails since its success rate is almost zero. Third, our tracker merely uses the ImageNet DET [28] and VID[28] datasets for training, while the DaSiam_LT method utilizes more than ten times training data for developing their algorithm (including the ImageNet DET [28], ImageNet VID [28], YTBB [27] and COCO detection [21] datasets).

Quantitative analysis on different attributes. We conduct this analysis on VOT2018LT dataset and report the results in Table 2. Our tracker achieves top three performance in most cases. MBMD also performs well, but it runs much slower than ours.

Table 2. Quantitative analysis with respect to different attributes. Visual attributes: (O) Full occlusion, (V) Out-of-view, (P) Partial occlusion, (C) Camera motion, (F) Fast motion, (S) Scale change, (A) Aspect ratio change, (W) Viewpoint change, (I) Similar objects, (D) Deformable object.

	O	V	P	C	F	S	A	W	I	D
Ours	0.5666	0.521	0.5734	0.6439	0.4520	0.5894	0.5708	0.6123	0.5495	0.4991
MBMD	0.5693	0.4985	0.5730	0.6437	0.4637	0.5719	0.5432	0.5780	0.5663	0.4752
DaSiam_LT	0.5525	0.5052	0.5379	0.6515	0.4785	0.5713	0.5453	0.6202	0.5701	0.4570
MMLT	0.5158	0.4823	0.4771	0.5929	0.4351	0.5188	0.504	0.5307	0.5213	0.4546
LTSINT	0.5718	0.4704	0.5358	0.5384	0.4582	0.5005	0.5123	0.4714	0.6026	0.4670
SYT	0.4729	0.4154	0.4508	0.5236	0.4234	0.4851	0.4255	0.5182	0.4995	0.3909
PTAVplus	0.4079	0.3143	0.4800	0.4547	0.1857	0.3599	0.3498	0.3677	0.3974	0.3738
FuCoLoT	0.4542	0.3378	0.4637	0.4667	0.2446	0.3959	0.3950	0.3745	0.4433	0.3828
SiamVGG	0.3265	0.3242	0.4480	0.4559	0.2195	0.4173	0.3358	0.4777	0.4445	0.2712
SLT	0.4067	0.3855	0.3982	0.5019	0.2871	0.4082	0.3938	0.4528	0.4110	0.3740
SiamFC	0.2339	0.2949	0.3523	0.4117	0.1519	0.3631	0.2759	0.4663	0.2873	0.2483
SiamFCDet	0.3852	0.3184	0.3170	0.4231	0.2920	0.3625	0.3640	0.4349	0.3135	0.3479
HMMTxD	0.2880	0.2584	0.3098	0.3702	0.2854	0.3362	0.2854	0.3423	0.3490	0.2705
SAPKLTF	0.2010	0.2230	0.2734	0.3534	0.1410	0.3243	0.2399	0.3502	0.3427	0.1598
ASMS	0.1809	0.2276	0.2437	0.3373	0.1497	0.2804	0.2294	0.3339	0.3017	0.1709
FoT	0.0761	0.1102	0.1280	0.1092	0.0358	0.1305	0.0989	0.1503	0.1066	0.0958

4.2. Ablation Study

We conduct ablation analysis to evaluate different components of our tracker using the VOT2018LT dataset.

Effectiveness of Different Components: The proposed long-term tracking framework includes skimming (S) and perusal modules, and the perusal module consists of a RPN-based regressor (R) and a robust verifier (V). To evaluate the contributions of different components, we implement the following variants: (1) Ours (R) denotes our tracker merely using the SiameseRPN model to conduct local search in every frame; (2) Ours (S+R) stands for our tracker combining the skimming module and SiameseRPN to conduct image-wide re-detection in every frame; (3) Ours (R+V) represents our final tracker without the skimming module; and (4) Our (S+R+V) is our final skimming-perusal tracker.

Table 3. Effectiveness of different components for our tracker.

Tracker	F-score	Pr	Re	fps
Ours (R)	0.553	0.561	0.545	34.7
Ours (S+R)	0.583	0.605	0.563	30.6
Ours (R+V)	0.606	0.635	0.579	20.0
Ours (S+R+V)	0.616	0.633	0.600	25.7

Table 3 reports the results of the above-mentioned variants, and illustrates that all components could improve the long-term tracking performance. First, the comparison between Ours (R) and Ours (R+V) demonstrates that our designed verifier improves the long-term tracking performance by a large margin but reduces the tracking speed significantly. Second, the comparison between Ours (R+V) and Ours (S+R+V) illustrates that our skimming module could efficiently speed up long-term tracking and slightly improve the tracking performance. This is because the skimming module effectively selects few possible regions from a large number of sliding windows when the tracker conducts image-wide re-detection, which can also filter out some distractors and avoid sending them to the regressor.

Threshold θ for Dynamically Switching: The threshold θ determines that the tracker will run on local search or global search in the next frame. A small threshold makes the tracker run more on the local search state. Extremely, the tracker will always track the object by local searching when $\theta = 0$. A large threshold treats the tracking result less reliable and makes the tracker exploit global search more frequently. Supposing $\theta = 1$, the tracker will always locate the target by image-wide re-detection. Figure 6 illustrates the tracking accuracies and speeds with different θ values, which shows that our tracker achieves the best performance with a satisfactory speed when $\theta = 0.65$.

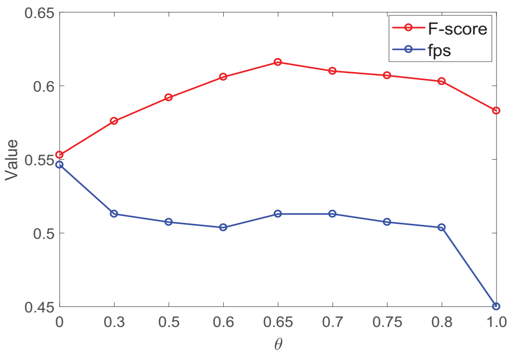


Figure 6. Effects of different θ values for dynamically switching. The fps values are divided by 50 for better illustrations.

Parameter K for Skimming: The parameter K determines the number of possible regions being selected and sent to PRN. Figure 7 illustrates that our tracker achieves the best accuracy and a satisfactory speed when $K = 3$.

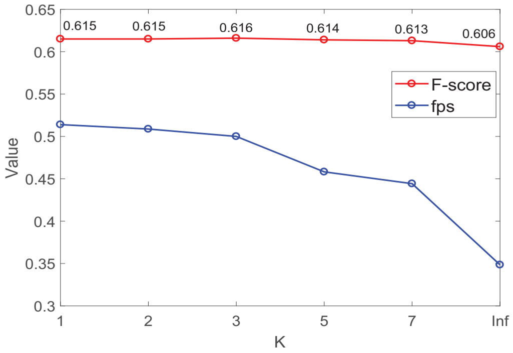


Figure 7. Effects of different K values for the skimming module. The fps values are divided by 50 for better illustrations.

Table 4. Comparison of different verification networks.

Tracker	F-score	Pr	Re	fps
Ours(MoblieNet)	0.597	0.626	0.571	31.7
Ours(VGG16)	0.596	0.622	0.571	19.6
Ours(ResNet50)	0.616	0.633	0.600	25.7
Ours(ResNet101)	0.601	0.630	0.571	18.3

Different Verification Networks: The offline-trained verification network aims to robust verify candidate proposals generated by SiameseRPN. We also investigate different kinds of networks, including MoblieNet, VGG16, ResNet50 and ResNet101. Table 4 shows that our tracker with ResNet50 as verifier takes a good trade-off between accuracy and speed. Besides, our tracker with MoblieNet also achieves good performance with a faster speed. Although the ResNet101 network is more powerful than ResNe50 in many vision tasks, it has not achieved better results but run much slower. This may be attributed to the complicated network structure and limited number of training samples.

4.3. Results on OxUvA

The OxUvA [33] long-term dataset consists of 366 object tracks in 337 videos, which are carefully selected from the YTBB [27] dataset and sparsely labled at a frequency of 1Hz. Compared with the popular short-term tracking dataset (such as OTB2015), this dataset has many long-term videos (each video lasts for average 2.4 minutes) and includes severe out-of-view and full occlusion challenges. In [33], the authors divide the OxUvA long-term dataset into two disjoint subsets, i.e., *dev* (with 200 tracks) and *test* (with 166 tracks) sets. Based on these two subsets, the OxUvA benchmark poses two challenges: constrained and open. For the former one, trackers can be developed merely using the video sequences from the OxUvA *dev* set. For the open challenge, trackers can use any public dataset except for the YTBB *validation* set since the OxUvA dataset is constructed upon it. In [33], there exist three major criteria to evaluate the performance of different trackers, namely, true positive rate (**TPR**), true negative rate (**TNR**) and maximum geometric mean (**MaxGM**). **TPR** gives the fraction of *present* objects that are reported *present* and correctly located, while **TNR** measures the fraction of *absent* objects that are reported *absent*. Then, the **MaxGM** rule (4) is defined to synthetically consider both **TPR** and **TNR**, which is adopted for ranking different trackers. For a given tracker, a larger **MaxGM** value means a better performance.

$$\mathbf{MaxGM} = \max_{0 \leq p \leq 1} \sqrt{((1-p)\cdot\mathbf{TPR})((1-p)\cdot\mathbf{TNR}+p)} \cdot \quad (4)$$

Until now, all trackers reported in OxUvA [33] and the recent MBMD method are tested using the open challenge. Thus, we also compare the proposed method with these trackers on the OxUvA open challenge for fair comparison. In this subsection, we compare the proposed tracker with the state-of-the-art MBMD method and ten competing algorithms reported in [33], including LCT [24], EBT [40], TLD [15], ECO-HC [5], BACF [9], Staple [2], MDNet [25], SINT [32], SiamFC [3] and SiamFC+R [33]. The comparison results are presented in Table 5.We can see that the proposed method achieves the top-ranked performance in terms of **MaxGM**, **TPR** and **TNR**. Especially, our tracker performs the best in comparison with other competing algorithms in terms of **MaxGM**, which is the most important metric on OxUvA. Compared with the VOT2018LT winner (MBMD) and the original best tracker (SiamFC+R), our method achieves a substantial improvement, with relative gains of 37.0% and 14.3% over **MaxGM**.

Table 5. Comparisons of different tracking algorithms on the OxUvA [33] long-term dataset. The best three results are marked in **red**, **blue** and **green** bold fonts respectively. The trackers are ranked from top to bottom using the **MaxGM** measure.

Tracker	MaxGM	TPR	TNR
SPLT(Ours)	**0.622**	0.498	**0.776**
MBMD	**0.544**	**0.609**	0.485
SiamFC+R	**0.454**	0.427	0.481
TLD	0.431	0.208	**0.895**
DaSiam_LT	0.415	**0.689**	0
LCT	0.396	0.292	**0.537**
SYT	0.381	**0.581**	0
LTSINT	0.363	0.526	0
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

Ours *vs* **Short-term Trackers**: We first compare our tracker with some popular short-term trackers. ECO-HC [5], BACF [9] and Staple [2] are three correlation-filter-based short-term trackers with high accuracies and fast speeds. MDNet [25], SINT [32] and SiamFC [3] are three popular deep-learning-based short-term trackers. Compared with these methods, our tracker achieves a very significant improvement in terms of all three quantitative criteria. Table 5 shows that the **TNR** values of the aforementioned short-term trackers are all zeros, which means that these trackers are not able to identify the *absent* of the tracked object when it moves out of view or is fully occluded. In principle, these short-term methods cannot meet the requirement of the long-term tracking task. In contrast, our tracker includes an efficient image-wide re-detection scheme and exploits an effective deep-learning appearance model.

Ours *vs* **Traditional Long-term Trackers**: LCT [24], EBT [40] and TLD [15] are three traditional long-term trackers with different hand-crafted features and re-detection schemes. Table 5 indicates that all deep long-term trackers perform better than traditional ones, which means the learned deep features and models are also effective in the long-term tracking task. Especially, our method outperforms the best traditional method (TLD) by a very large margin (0.622 *vs* 0.431 over **MaxGM**).

Ours *vs* **Deep Long-term Trackers**: SiamFC+R [33] equips the original SiamFC tracker with a simple re-detection scheme similar to [13]. Compared with it, our tracker exploits a more robust perusal model to precisely locate the tracked object, and therefore performs better than SiamFC+R. Both MBMD and our methods utilize the same SiameseRPN-based regressor but different verifiers. In addition, the proposed skimming module could efficiently not only speed up the image-wide re-detection but also filter out some distractors. Thus, our tracker achieves more accuracy and runs much faster than the MBMD method. Our tracker also outperforms three VOT2018LT trackers (DaSiam_LT, LTSINT, SYT) by a very large margin.

5. Conclusion

This work presents a novel ‘Skimming-Perusal’ tracking framework for long-term visual tracking. The perusal module aims to precisely locate the tracked object in a local search region using the offline-trained regression and verification networks. Based on the confidence score output by the perusal module, the tracker determines the tracked object being present or absent and invokes image-wide re-detection when absent. The skimming module focuses on efficiently selecting the most possible regions from densely sampled sliding windows, thereby speeding up the global search process. Numerous experimental results on two recent benchmarks show that our tracker achieves the best performance and runs at a real-time speed. It is worth noticing that our ‘Skimming-Perusal’ model is a simple yet effective real-time long-term tracking framework. We believe that it can be acted as a new baseline for further researches.

Acknowledgements. This paper is supported in part by National Natural Science Foundation of China (Nos. 61872056, 61725202, 61829102, 61751212), in part by the Fundamental Research Funds for the Central Universities (Nos. DUT19GJ201, DUT18JC30), and in part by Natural Science Foundation of Liaoning Province (No. 20180550673).

References

[1] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. TensorFlow: Large scale machine learning on heterogeneous distributed systems. In *CoRR abs/1603.04467*, 2016.

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

[3] 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.

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

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

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

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

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

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

[10] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L. Hicks, and Philip H. S. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

[11] 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.

[12] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. In *CoRR abs/1704.04861*, 2017.

[13] James Steven Supancic III and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *ICCV*, 2017.

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

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

[16] 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. 2018.

[17] 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.

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

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

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

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

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

[23] Alan Lukei, Luka ehovin Zajc, Tom Voj, Ji Matas, and Matej Kristan. Now you see me: Evaluating performance in long-term visual tracking. In *ECCV*, 2018.

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

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

[26] Georg Nebehay and Roman Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *CVPR*, 2015.

[27] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

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

[31] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018.

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

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *CoRR abs/1708.04896*, 2017.

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

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

[42] C. Lawrence Zitnick and Piotr Dollár. Edge Boxes: Locating object proposals from edges. In *ECCV*, 2014.