Supplemental Material for Siam R-CNN: Visual Tracking by Re-Detection

Paul Voigtlaender[1] Jonathon Luiten[1,2†] Philip H.S. Torr[2] Bastian Leibe[1]
[1]RWTH Aachen University [2]University of Oxford
{voigtlaender,luiten,leibe}@vision.rwth-aachen.de phst@robots.ox.ac.uk

Abstract

We provide more details of Siam RCNN's training procedure, a more detailed description of the video hard example mining procedure, of the short-term tracking algorithm, and of the estimation of rotated bounding boxes for VOT2018. Additionally, we explain and analyze different methods to speed up Siam R-CNN in more detail and we analyze the amount of training data used by previous methods compared to our method. Moreover, we conduct a per-attribute analysis of variants of Siam R-CNN on OTB2015 and also analyze the effect of fine-tuning Box2Seg. Finally, we present additional quantitative and qualitative results for short-term tracking, long-term tracking and video object segmentation.

1. Further Method Details

In the following, we provide further details on the training procedure, video hard example mining, the short-term-tracking algorithm, and on rotated bounding box estimation.

1.1. Training

We train Siam R-CNN with random image scale sampling by scaling the small edge of the image to a random value between 640 and 800 pixels, while keeping the aspect ratio. In cases where this resizing would result in a larger edge size of more than 1333 pixels, it is resized to a larger edge size of 1333 pixels instead. During test time we resize the image to a smaller edge length of 800 pixels, again keeping the longer edge size no larger than 1333 pixels. Note that these settings are the default of the Mask R-CNN implementation we used.

We train our network with two NVIDIA GTX 1080 Ti GPUs for 1 million steps (5.8 days) with a learning rate of 0.01, and afterwards for 120,000 steps and 80,000 steps with learning rates of 0.001 and 0.0001, respectively. The hard example training is done afterwards with a single GPU for 160,000 steps and a learning rate of 0.001. A batch size of one pair of images (reference and target) per GPU is used.

1.2. Video Hard Example Mining

Index Structure. For the indexing structure, we use the "Approximate Nearest Neighbors Oh Yeah" library for approximate nearest neighbor queries[1].

Feature Pre-computation. During normal training without hard negative examples, the RoIs given to the second stage are generated automatically by the RPN and are thus not always perfectly aligned to the object boundaries. In the three cascade stages, the RoI will then be successively refined by bounding box regression. However, when naively pre-computing the features for the ground truth bounding boxes, the network might overfit to these perfect boxes.

In order for the network to learn to handle imperfect bounding boxes, we add random Gaussian noise to the ground truth bounding boxes before pre-computing the features, and afterwards run these jittered RoIs through the cascade stages and also pre-compute the re-aligned features after every cascade stage.

In particular, we take the ground truth bounding box (x_0, y_0, x_1, y_1) and independently add to each component noise sampled from a clipped Gaussian with mean 0 and standard deviation 0.25, *i.e.*, for $i \in \{0, 1, 2, 3\}$, we have

$$\tilde{r}_i \sim \mathcal{N}(0, 0.25), \tag{1}$$

$$r_i = \text{clip}(\tilde{r}_i, -0.25, 0.25). \tag{2}$$

The jittered box is then given by

$$(x_0 + r_0, y_0 + r_1, x_1 + r_2, y_1 + r_3). \tag{3}$$

Training Procedure. Having pre-computed the RoI-aligned features for each object and each cascade stage, the training procedure now works as follows. For each training step, as usual, a random video and object in this video

[†]Work performed while both at the RWTH Aachen and on a research visit at the University of Oxford.

[1]https://github.com/spotify/annoy

is selected and then a random reference and a random target frame. Afterwards, we use the indexing structure to retrieve the 10,000 nearest neighbor bounding boxes from other videos (50,000 boxes for LaSOT because of the long sequences). Note that the nearest neighbors are searched for over all training datasets, regardless where the reference comes from. Since the nearest neighbors are found per frame, often a few videos will dominate the set of nearest neighbors. To get more diverse negative examples, we create a list of all videos (excluding the reference) in which nearest neighbor boxes were found and randomly select 100 of these videos. For each of the 100 videos, we then randomly select 1 of the boxes which were retrieved as nearest neighbors from this video and add them as negative examples for the re-detection head.

Adding only additional negative examples creates an imbalance in the training data. Hence, we also retrieve the features of the ground truth bounding boxes of 30 randomly selected frames of the current reference video as additional positive examples.

1.3. Short-term Tracking Algorithm

For the VOT2018 dataset [27], it is standard to use a reset-based evaluation, where once the object is lost (0 IoU between predicted and ground truth bounding box), the tracker is restarted with the ground truth box five frames later and receives a penalty. This extreme short-term tracking scenario is not what Siam R-CNN with the Tracklet Dynamic Programming Algorithm (TDPA) was designed for. It often triggers resets, which normally (without reset-based evaluation) Siam R-CNN could have automatically recovered from.

Since VOT2018 is an important tracking benchmark, we created a short-term version of the Siam R-CNN tracking algorithm (see Alg. 1). Given the RoI Aligned features $\mathrm{ff\_gt\_feats}$ of the first-frame bounding box and the previous-frame tracking result det_{t-1}, we first extract the backbone features of the current image and afterwards generate regions of interest (RoIs) using the region proposal network (RPN, lines 1–3).

Note, that we know for sure that the previous-frame predicted box det_{t-1} has a positive IoU with the ground truth box, as otherwise a reset would have been triggered. Hence, the object to be tracked should be located close to the previous-frame predicted box. In order to exploit this, and to compensate for potential false negatives of the RPN, we add shifted versions of the previous-frame prediction as additional RoIs (lines 5–9). Here, the function $\mathrm{shift}(\cdot)$ shifts the previous-frame box det_{t-1} by factors of its width and height, *e.g.*, if $\mathrm{shift}_x = 0.5$ and $\mathrm{shift}_y = 1.0$, the box is shifted by half its width in x-direction and by its full height in y-direction.

Afterwards, we use the $\mathrm{redetection\_head}$ to produce

Algorithm 1 Perform Short-term Tracking for time-step t

1: **Inputs** $\mathrm{ff\_gt\_feats}, \mathrm{image_t}, \mathrm{det_{t-1}}$
2: $\mathrm{backbone\_feats} \leftarrow \mathrm{backbone}(\mathrm{image}_t)$
3: $\mathrm{RoIs} \leftarrow \mathrm{RPN}(\mathrm{backbone\_feats})$
4: ▷ Add shifted versions of det_{t-1} to RoIs
5: **for** $\mathrm{shift}_x \in \{-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5\}$ **do**
6: **for** $\mathrm{shift}_y \in \{-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5\}$ **do**
7: $\mathrm{RoIs} \leftarrow \mathrm{RoIs} \cup \{\mathrm{shift}(\mathrm{det}_{t-1}, \mathrm{shift}_x, \mathrm{shift}_y)\}$
8: **end for**
9: **end for**
10: $\mathrm{dets}_t, \mathrm{det\_scores}_t \leftarrow \mathrm{redetection\_head}(\mathrm{RoIs}, \mathrm{ff\_gt\_feats})$
11: $\mathrm{prev\_scores}_t \leftarrow \mathrm{score\_redetection}(\mathrm{dets}_t, \mathrm{det}_{t-1})$
12: $\mathrm{loc\_scores}_t \leftarrow |\mathrm{bbox}(\mathrm{dets}_t) - \mathrm{bbox}(\mathrm{det}_{t-1})|_1$
13: $\mathrm{scores}_t \leftarrow \mathrm{det\_scores}_t + \mathrm{prev\_scores}_t + \delta \cdot \mathrm{loc\_scores}_t$
14: ▷ Filter out detections with too large spatial distance
15: **for** $\mathrm{det}_t \in \mathrm{dets}_t$ **do**
16: **if** $\|\mathrm{det}_t - \mathrm{det}_{t-1}\|_\infty > \xi$ **then**
17: $\mathrm{scores}[\mathrm{det}_t] \leftarrow -\infty$
18: **end if**
19: **end for**
20: **return** $\arg\max_{\mathrm{det}_t \in \mathrm{dets}_t} \mathrm{scores}_t[\mathrm{det}_t]$

detections dets_t with detection scores $\mathrm{det\_scores}_t$ for the current frame t (line 10). We then additionally compute previous-frame scores $\mathrm{prev\_scores}_t$ by using the previous-frame box as a reference to score the current detections (line 11). To exploit spatial consistency, we also compute location scores ($\mathrm{loc\_scores}_t$, line 12), given by the L_1-norm of the pairwise differences between the current detection boxes and the previous-frame predicted box. All three scores are then combined (line 13) by a linear combination, where the current-frame and previous-frame scores have equal weight.

Even when using a location score, it can happen, that a distractor object appears far away from the object to be tracked and gets a high combined score because it looks very similar to that object. However, since we know that the previous-frame box has positive overlap with the ground truth box, a far-away detection cannot be the object to be tracked. Hence, we explicitly filter out detections which have a large spatial distance (measured by the L_∞-norm) to the previous-frame predicted box (lines 15–19).

Finally, we report the detection with the highest combined score as the result for the current frame (line 20), or in case there is no valid detection, we repeat the previous-frame result as result for the current frame.

1.4. Rotated Bounding Box Estimation

For VOT2018, the ground truth is given as rotated bounding boxes which were automatically estimated by an optimization procedure based on hand-annotated segmentation masks [26]. Nevertheless, most methods produce axis-aligned bounding boxes and then evaluate against the ro-

Method	EAO ↑	Accuracy ↑	Robustn. ↓
DiMP-50 [3]	**0.440**	0.597	0.153
SiamRPN++ [29]	0.414	0.440	0.234
ATOM [13]	0.401	0.590	0.204
Ours (short-term)	0.408	0.609	0.220
+Rotated Boxes	0.409	**0.686**	0.272
+Rotated Boxes +Mask Dens. Filt.	**0.423**	0.684	**0.248**

Table 1: Results using rotated bounding boxes on VOT2018. Mask Dens. Filt. denotes using a mask density filter.

Dataset Eval measure	Speed FPS	OTB2015 AUC	LaSOT AUC	LTB35 F
Siam R-CNN	4.7	70.1	**64.8**	66.8
ResNet-50	5.1	68.0	62.3	64.4
$\frac{1}{2}$ res.	5.7	**70.2**	62.9	65.6
100 RoIs	8.7	68.7	64.1	**67.3**
100 RoIs + $\frac{1}{2}$ res.	13.6	69.1	63.2	66.0
ResNet-50 + 100 RoIs	10.3	66.9	61.5	65.9
ResNet-50 + $\frac{1}{2}$ res.	6.1	68.6	61.2	64.0
ResNet-50 + 100 RoIs + $\frac{1}{2}$ res.	**15.2**	67.7	61.1	63.7

Table 2: Extended timing analysis of Siam R-CNN using a V100 GPU.

tated bounding box ground truth.

As an extension of Siam R-CNN, we used Box2Seg to produce segmentation masks and then also ran the optimization procedure which was used to create the ground truth to generate rotated bounding boxes. Note that this optimization procedure (implemented in MATLAB) is very slow, slowing down our whole method to around 0.23 FPS. Note that the speed of the optimization might strongly depend on the hardware/software setup. However, here we do not aim for a good run-time but instead want to analyze the achievable performance using rotated bounding boxes.

Table 1 shows our results on VOT2018 with rotated bounding boxes. When creating a rotated bounding box for each frame, the overall EAO stays almost the same and only increases from 0.408 to 0.409. However, the accuracy which measures the average intersection-over-union of the bounding box with the ground truth while disregarding resets, increases strongly from 0.609 to 0.686. At the same time, the number of resets strongly increases which can be seen by the robustness degrading from 0.220 to 0.272.

A manual inspection of the results revealed that in some cases the estimated segmentation mask from Box2Seg was almost empty and the resulting rotated bounding box is hence of very poor quality and can easily trigger a reset.

To avoid these cases, we apply a mask density filter, which means that in cases where the estimated segmentation mask fills less than 10% of the bounding box which was used to generate it, we stick to the original axis-aligned bounding box in this frame instead of reporting the rotated bounding box. In this setup, the EAO significantly increases to 0.423, while keeping a high accuracy of 0.684. With the mask density filter, Siam R-CNN achieves a robustness of 0.248 which is still worse than the robustness of the axis-aligned version, but significantly better than the version without the filter.

2. Further Analyses

In the following, we conduct further analyses of the speed-accuracy trade-off of Siam R-CNN and of the dependence of Siam R-CNN and other methods on the used training data. Additionally, we conduct a per-attribute analysis on OTB2015, and an analysis of the fine-tuning of Box2Seg.

2.1. Speed-Accuracy Trade-off

Tab. 2 extends the timing analysis of the main paper. Here, we evaluate three changes aimed at increasing the speed of Siam R-CNN (smaller backbone, smaller input resolution, and fewer RoI proposals) in more detail.

When evaluating with a ResNet-50 backbone, Siam R-CNN performs slightly faster and still achieves state-of-the-art (SOTA) results (62.3 on LaSOT, compared to 56.8 for DiMP-50 with the same backbone). This shows that the strong results are not only due to a larger backbone, but due to our tracking by re-detection approach.

We also evaluate reducing the image input size to a smaller image edge length of 400 pixels instead of 800 (row "$\frac{1}{2}$ res"). This also results in only a slight decrease in performance in two benchmarks, and a slight increase in performance on OTB2015.

The row "100 RoIs" shows the results of using 100 RoIs from the RPN, instead of 1000. This almost doubles the speed as most compute occurs in the re-detection head. This results in only a small score decrease on two benchmarks, while improving results on LTB35. This shows that Siam R-CNN can run very quickly, even though it is based on a two-stage detection architecture, as very few RoIs are required.

The fastest setup with all three of these speed improvements (ResNet-50 + 100 RoIs + $\frac{1}{2}$ res.) achieves 15.2 frames per second with a V100 GPU, but still achieves strong results, especially for long-term tracking. The same setup with a ResNet-101 backbone instead of ResNet-50 (100 RoIs + $\frac{1}{2}$ res.) runs at 13.6 frames per second and achieves excellent results and loses at most 1.6 percentage points over these three datasets compared to the standard Siam R-CNN, while running almost three times as fast.

Eval.	Method	Videos						Additional Images			Total
		GOT-10k 9k	ImageNet-Vid 4k	LaSOT 1k	YT-VOS 3k	TrackingNet 30k	YT-BB 380k	COCO 119k	ImageNet 1281k	ImageNet-Det 457k	Videos + Add. Imgs
ALL	Siam R-CNN	✓	✓	✓	✓			✓			18k+119k
	DiMP & ATOM	✓		✓		✓		✓	✓		41k+1400k
	SiamRPN++		✓				✓	✓	✓	✓	384k+1867k
GOT	Siam R-CNN	✓						✓			9k+119k
	DiMP & ATOM	✓							✓		9k+1281k
	SiamRPN++	✓						✓	✓		9k+1400k

Table 3: Training data used compared to some important recent methods [3, 13, 29]. Videos + Add. Imgs: Number of videos plus number of additional images not in videos. Eval.: Evaluation benchmark setup. ALL: All benchmarks except GOT-10k. GOT: Evaluate on GOT-10k, use only GOT-10k training data in addition to static images. ImageNet-Vid: ImageNet Video, YT-VOS: YouTube-VOS, YT-BB: YouTube BoundingBoxes.

	ALL	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
Siam R-CNN (ours)	**70.1**	**69.1**	64.5	**71.0**	**69.9**	71.6	71.1	**74.2**	**66.6**	**68.6**	**67.9**	**72.1**
No hard ex. min.	-1.7	-3.0	-1.3	-2.2	-2.2	-2.3	-5.3	-2.1	-1.7	-2.1	-1.2	-1.7
No TDPA (Argmax)	-6.3	-7.9	-1.5	-5.6	-9.5	-3.3	-14.0	-5.8	-4.6	-8.4	-6.8	-7.2
No TDPA (Short-term)	-2.8	-6.5	-2.8	-3.5	-1.6	-2.5	-6.2	-8.4	-4.8	-3.7	-14.5	-5.3
0 RoIs	-12.9	-14.0	-10.5	-20.5	-10.5	-12.2	-20.3	-25.8	-14.9	-12.1	-21.8	-13.8
100 RoIs	-1.4	-1.8	**+1.2**	-3.6	-3.1	-0.8	-7.8	-4.5	-1.6	-2.2	-4.2	-2.1
10000 RoIs	-0.5	-0.8	0.0	-0.1	-0.6	**+0.4**	**+0.8**	0.0	-0.7	-0.7	-0.3	-0.6
DiMP-50 [3]	-1.3	-5.3	**+3.7**	-2.6	-1.4	-2.2	-8.1	-4.3	-0.2	-0.8	-5.4	-2.8
SiamRPN++ [29]	-0.4	-0.0	+1.7	-2.0	-0.5	-0.3	-5.1	-3.0	-0.3	-0.3	-3.1	-2.4

Table 4: Per attribute ablation for Success (AUC) on OTB2015. The first row shows the performance of the full model, and all other rows show the absolute difference to the full model. All: All Videos, BC: Background Clutters, DEF: Deformation, FM: Fast Motion, IPR: In-Plane Rotation, IV: Illumination Variation, LR: Low Resolution, MB: Motion Blur, OCC: Occlusion, OPR: Out-of-Plane Rotation, OV: Out-of-View, SV: Scale Variation.

2.2. Training Data Dependence

We show how our training data compares to the data used by some important recent methods in Table 3. We use more video 'datasets', but actually use far less data. DiMP-50 [3] and ATOM [13] both use 2.28 times more videos. SiamRPN++ [29] uses 21.3 times more. All three use ImageNet and train on COCO by creating artificial videos using augmentations (we use COCO without this complex kind of augmentation). The only dataset we use which is not used by any of the other considered methods is YouTube-VOS [65], as we also evaluate on VOS benchmarks. For GOT-10k [21] evaluation, where only GOT-10k video training data is allowed, all other methods also use static images from ImageNet. For SiamRPN++, we use COCO but not ImageNet. This shows that our strong results are not due to the amount of training data.

2.3. Per-Attribute Analysis

Table 4 shows a per-attribute ablation on OTB2015. Hard example mining improves results over all attributes and is particularly helpful for low resolution (LR) and background clutter (BC).

Our Tracklet Dynamic Programming Algorithm (TDPA) models spatio-temporal consistency cues only where it is likely that there are consistent predictions, by building up tracklets. It corrects itself immediately after disappearance by tracking all objects simultaneously, and determining the most likely set of previous tracklets for the object online using dynamic programming. For the Out-of-View (OV) attribute (the target disappears in the video), TDPA significantly outperforms Short-term, which is unable to rely on spatio-temporal consistency cues during disappearance and thus often fails and performs worse than Argmax. TDPA tackles this problem of object disappearance, by using a dynamic and robust type of spatio-temporal consistency cues and also increases robustness against distractors, improving results across all attributes.

2.4. Fine-tuning Analysis

Figure 1 shows the result of Siam R-CNN with a different number of fine-tuning steps for Box2Seg. For the fine-tuned Box2Seg variant in the main paper, we used 300 steps which yields a speed-accuracy trade-off of 74.8 $\mathcal{J}\&\mathcal{F}$ with a run-time of 1 FPS (compared to 70.6 $\mathcal{J}\&\mathcal{F}$ and 3.1 FPS without fine-tuning). Note that here the timing is per frame, and not per object, so that the run-time without fine-tuning

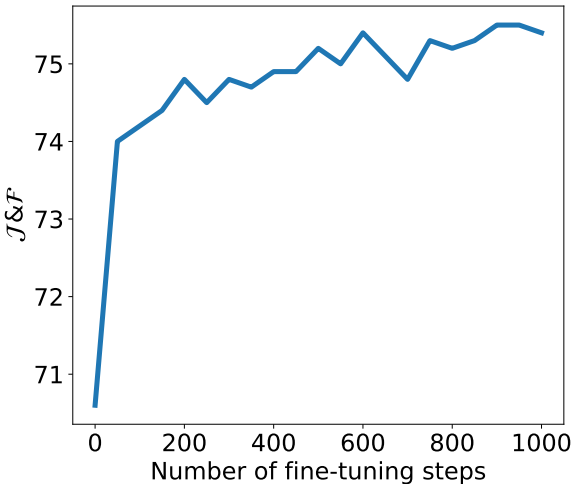

Figure 1: Segmentation quality on the DAVIS 2017 validation set depending on the number of fine-tuning steps.

is longer than for single-object tracking scenarios. When increasing the number of fine-tuning steps to 1000, Siam R-CNN achieves a $\mathcal{J}\&\mathcal{F}$ score of 75.4 with a run-time of 0.38 FPS.

3. Further Experimental Results

In addition to the 11 benchmarks that we presented in the main paper, we present here results on eight further benchmarks, of which five are short-term tracking benchmarks, one is a long-term tracking benchmark and two are video object segmentation benchmarks. For all benchmarks (except for the three VOT benchmarks) we use exactly the same tracking parameters. This is in contrast to many other methods which have parameters explicitly tuned for each dataset. This shows the generalization ability of our tracker to many different scenarios. For the three VOT datasets we use the short-term variant of the tracking parameters (with the same parameters across these three benchmarks).

3.1. Further Short-Term Tracking Evaluation

In the main paper we presented short-term tracking results on OTB2015 [63], UAV123 [41], NfS [25], TrackingNet [42], VOT2018 (the same as VOT2017) [27], and GOT-10k [21]. Here in the supplemental material we present results on five further short-term tracking benchmarks. These further benchmarks are OTB-50 [63], OTB2013 [62], VOT2015 [28], VOT2016 [26], and TempleColor128 (TC128) [35].

OTB-50. We evaluate on the OTB-50 benchmark [63] (50 videos, 539 frames average length). This dataset is a subset of OTB2015 using exactly half of the sequences. It is evaluated with the same evaluation measures as OTB2015. Tab. 5 compares our results to five state-of-the-art trackers.

	SA-Siam [19]	SINT++ [61]	RTINet [70]	SPM [56]	ACT [6]	Siam R-CNN
Success AUC	61.0	62.4	63.7	65.3	65.7	**66.3**

Table 5: Results on OTB-50 [63].

	RPCF [51]	SACF [72]	MCCT [58]	DRT [49]	GFS-DCF [66]	Siam R-CNN
Success AUC	71.3	71.3	71.4	72.0	**72.2**	70.4

Table 6: Results on OTB2013 [62].

	MCCT [58]	STRCF [31]	RTINet [70]	ASRCF [11]	UPDT [4]	Siam R-CNN
Success AUC	59.6	60.1	60.2	60.3	**62.2**	60.1

Table 7: Results on TC128 [35].

	FlowTrack [77]	SACF [72]	SiamRPN [30]	SiamDW [74]	DaSiamRPN [76]	Ours (short-t.)
EAO	34.1	34.3	35.8	38.0	44.6	**45.4**

Table 8: Results on VOT2015 [28].

	DaSiamRPN [76]	SPM [56]	DRT [49]	SiamMask [60]	UpdateNet [71]	Ours (short-t.)
EAO	41.1	43.4	44.2	44.2	**48.1**	46.5

Table 9: Results on VOT2016 [26].

Siam R-CNN achieves 66.3 AUC, which outperforms the previous best published results by ACT [6] by 0.6 percentage points.

OTB2013. We evaluate on the OTB2013 benchmark [62] (51 videos, 578 frames average length). This dataset is a predecessor to OTB2015 and is evaluated with the same evaluation measures. Tab. 6 compares our results to five state-of-the-art trackers. Siam R-CNN achieves 70.4 AUC, which is comparable to state-of-the-art trackers while being 1.8 percentage points behind the best published results by GFS-DCF [66].

TC128. We evaluate on the TempleColor128 (TC128) benchmark [35] (128 videos, 429 frames average length). This dataset is also evaluated using the OTB2015 evaluation measures. Tab. 7 compares our results to five state-of-the-art trackers. Siam R-CNN achieves 60.1 AUC, which is comparable to state-of-the-art trackers while being slightly inferior to the best published results by UPDT [4] by 2.1 percentage points.

VOT2015. We evaluate on the VOT2015 benchmark [28] (60 videos, 358 frames average length). This is evaluated with the same evaluation measures as VOT2018. Tab. 8 compares our results to five state-of-the-art trackers. The

	SiamFC [2]	PTAV [16]	ECO [12]	SiamRPN [30]	DaSiamRPN [76]	Siam R-CNN
Success AUC	39.9	42.3	43.5	45.4	61.7	**67.2**

Table 10: Results on UAV20L [41].

short-term version of Siam R-CNN achieves 45.4 EAO, which outperforms the previous best published results by DaSiamRPN [76] by 0.8 percentage points.

VOT2016. We evaluate on the VOT2016 benchmark [26] (60 videos, 358 frames average length). This is also evaluated with the same evaluation measures as VOT2018. Tab. 9 compares our results to five state-of-the-art trackers. The short-term version of Siam R-CNN achieves 46.5 EAO, which outperforms all previous published results except those of UpdateNet [71] which outperforms our results by 1.6 percentage points.

3.2. Further Long-Term Tracking Evaluation

We evaluate on one further long-term tracking dataset, in addition to the three benchmarks presented in the main paper.

UAV20L. We evaluate on the UAV20L benchmark [41] (20 videos, 2934 frames average length). This dataset contains 20 of the 123 sequences of UAV123, however each of these sequences extends for many more frames than the equivalent sequence in the UAV123 version. It is also evaluated with the same evaluation measures as OTB2015. Tab. 10 compares our results to five state-of-the-art trackers. Siam R-CNN achieves 67.2 AUC, which outperforms the previous best published results by DaSiamRPN [76] by 5.5 percentage points, which further highlights the ability of Siam R-CNN to perform long-term tracking.

3.3. Further Video Object Segmentation Evaluation

Table 11 is an extended version of the results on the DAVIS 2017 [46] validation set shown in the main paper.

Table 12 shows results on the DAVIS 2016 validation set [45] (20 videos, 68.8 frames average length, 1 object per video) compared to 14 state-of-the-art methods. Methods are ranked by the mean of $\mathcal{J}$ and $\mathcal{F}$. Among methods which only use the first-frame bounding box (without the mask), Siam R-CNN achieves the strongest result with 78.6% $\mathcal{J}\&\mathcal{F}$, which is 8.8 percentage points higher than SiamMask [60]. When fine-tuning Box2Seg, our method achieves 87.1% $\mathcal{J}\&\mathcal{F}$, which is close to the best result on DAVIS 2016 by STM-VOS [43] with 89.3%.

Table 13 shows results on the DAVIS 2017 [46] test-dev set (30 videos, 67.9 frames average length, 2.97 objects per video on average) compared to six state-of-the-art methods. Siam R-CNN achieves 53.3% $\mathcal{J}\&\mathcal{F}$, which is more than 10 percentage points higher than the result of SiamMask [60].

Init	Method	FT	M	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}_{box}$	t(s)
bbox	**Siam R-CNN (ours)**	✗	✗	**70.6**	66.1	**75.0**	**78.3**	0.32
	Siam R-CNN (fastest)	✗	✗	70.5	**66.4**	74.6	76.9	0.12
	SiamMask [60]	✗	✗	55.8	54.3	58.5	64.3	**0.06**†
	SiamMask [60] (Box2Seg)	✗	✗	63.3	59.5	67.3	64.3	0.11
	SiamRPN++ [29] (Box2Seg)	✗	✗	61.6	56.8	66.3	64.0	0.11
	DiMP-50 [3] (Box2Seg)	✗	✗	63.7	60.1	67.3	65.6	0.10
mask	STM-VOS [43]	✗	✓	**81.8**	**79.2**	**84.3**	–	0.32†
	FEELVOS [53]	✗	✓	71.5	69.1	74.0	71.4	0.51
	RGMP [64]	✗	✓	66.7	64.8	68.6	66.5	**0.28**†
	VideoMatch [20]	✗	✓	62.4	56.5	68.2	–	0.35
	FAVOS [8]	✗	✓	58.2	54.6	61.8	68.0	1.2†
	OSMN [68]	✗	✓	54.8	52.5	57.1	60.1	**0.28**†
mask+ft	PReMVOS [38]	✓	✓	**77.8**	**73.9**	**81.7**	**81.4**	37.6
	Ours (Fine-t. Box2Seg)	✓	✓	74.8	69.3	80.2	78.3	**1.0**
	DyeNet [33]	✓	✓	74.1	–	–	–	9.32†
	OSVOS-S [40]	✓	✓	68.0	64.7	71.3	68.4	9†
	CINM [1]	✓	✓	67.5	64.5	70.5	72.9	>120
	OnAVOS [55]	✓	✓	63.6	61.0	66.1	66.3	26
	OSVOS [5]	✓	✓	60.3	56.6	63.9	57.0	18†
	GT boxes (Box2Seg)	✗	✗	82.6	79.3	85.8	100.0	–
	GT boxes (Fine-t. Box2Seg)	✓	✓	86.2	81.8	90.5	100.0	–

Table 11: Results on the DAVIS 2017 validation set. FT: fine-tuning, M: using the first-frame masks, t(s): time per frame in seconds. †: timing extrapolated from DAVIS 2016 assuming linear scaling in the number of objects. Siam R-CNN (fastest) denotes Siam R-CNN with ResNet-50 backbone, half input resolution, and 100 RoIs from the RPN.

STM-VOS [43] performs significantly better with 72.3%, however it relies on the first-frame mask which makes it less usable in practice.

Fig. 2 shows the speed-accuracy tradeoff of different methods for the YouTube-VOS 2018 validation set. Again, Siam R-CNN achieves a good speed/accuracy trade-off which is only beaten by STM-VOS [43] (which relies on the first-frame mask).

3.4. Further Qualitative Evaluation

In Figure 3 we present further qualitative results of our method on the OTB2015, LTB35 and DAVIS 2017 benchmarks. We present results of our method compared to the best competing method. We show sequences for which Siam R-CNN has the best and worst relative performance compared to the competing method, as well as the sequence with the median relative performance.

3.5. Thorough Comparison to Previous Methods

Throughout the main paper and supplemental material we presented results on 11 short-term tracking benchmarks and four long-term tracking benchmarks, however these

Init	Method	FT	M	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}_{box}$	t(s)
bbox	**Siam R-CNN (ours)**	✗	✗	78.6	76.8	80.4	**86.6**	0.24
	Siam R-CNN (fastest)	✗	✗	**79.0**	**77.4**	**80.6**	85.0	0.08
	SiamMask [60]	✗	✗	69.8	71.7	67.8	73.3	**0.03**
	SiamMask [60] (Box2Seg)	✗	✗	75.9	75.6	76.3	73.3	0.06
mask	STM-VOS [43]	✗	✓	**89.3**	**88.7**	**89.9**	–	0.16
	RGMP [64]	✗	✓	81.8	81.5	82.0	79.3	**0.14**
	FEELVOS [53]	✗	✓	81.7	81.1	82.2	80.2	0.45
	FAVOS [8]	✗	✓	81.0	82.4	79.5	**83.1**	0.6
	VideoMatch [20]	✗	✓	80.9	81.0	80.8	–	0.32
	PML [7]	✗	✓	77.4	75.5	79.3	75.9	0.28
	OSMN [68]	✗	✓	73.5	74.0	72.9	71.8	**0.14**
mask+ft	**Ours (Fine-t. Box2Seg)**	✓	✓	**87.1**	85.3	**88.8**	86.6	**0.56**
	PReMVOS [38]	✓	✓	86.8	84.9	88.6	**89.9**	32.8
	DyeNet [33]	✓	✓	–	**86.2**	–	–	4.66
	OSVOS-S [40]	✓	✓	86.5	85.6	87.5	84.4	4.5
	OnAVOS [55]	✓	✓	85.0	85.7	84.2	84.1	13
	CINM [1]	✓	✓	84.2	83.4	85.0	83.6	> 120
	OSVOS [5]	✓	✓	80.2	79.8	80.6	76.0	9
	GT boxes (Box2Seg)	✗	✗	80.5	79.1	81.9	100.0	–
	GT boxes (Fine-t. Box2Seg)	✓	✓	89.0	87.6	90.5	100.0	–

Table 12: Quantitative results on the DAVIS 2016 validation set. FT denotes fine-tuning, M denotes using the first-frame mask, and t(s) denotes time per frame in seconds. Siam R-CNN (fastest) denotes Siam R-CNN with ResNet-50 backbone, half input resolution, and 100 RoIs from the RPN.

Init	Method	FT	M	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	t(s)
bbox	**Siam R-CNN (ours)**	✗	✗	**53.3**	**48.1**	**58.6**	0.44
	Siam R-CNN (fastest)	✗	✗	51.6	46.3	56.8	0.16
	SiamMask [60]	✗	✗	43.2	40.6	45.8	**0.09**†
mask	STM-VOS [43]	✗	✓	**72.3**	**69.3**	**75.2**	**0.48**†
	RGMP [64]	✗	✓	52.9	51.4	54.4	0.42†
	FEELVOS [53]	✗	✓	57.8	55.2	60.5	0.54
mask+ft	PReMVOS [38]	✓	✓	**71.6**	**67.5**	**75.7**	41.3
	Ours (Fine-t. Box2Seg)	✓	✓	62.1	57.3	66.9	**1.48**
	OnAVOS [55]	✓	✓	56.5	53.4	59.6	39

Table 13: Quantitative results on the DAVIS 2017 test-dev set. FT denotes fine-tuning, M denotes using the first-frame masks, and t(s) denotes time per frame in seconds. †: timing extrapolated from DAVIS 2016 assuming linear scaling in the number of objects. Ours (fastest) denotes Siam R-CNN with ResNet-50 backbone, half input resolution, and 100 RoIs from the RPN.

comparisons are spread throughout a number of tables and figures. We provide a unified and thorough comparison of our results to previous methods across all of these benchmarks in Table 14. We compare to the results of every paper

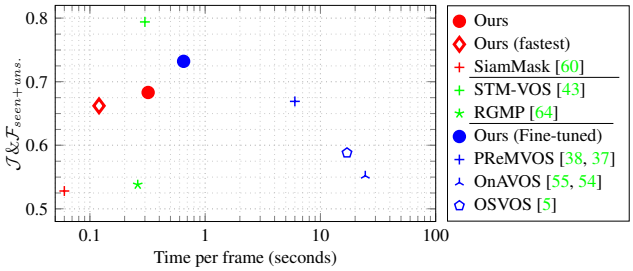

Figure 2: Quality versus timing on the YouTube-VOS 2018 [65] validation set. Only SiamMask [60] and our method (red) are able to work without the ground truth mask of the first frame and require just the bounding box. Methods shown in blue fine-tune on the first-frame mask. Ours (fastest) denotes Siam R-CNN with ResNet-50 backbone, half input resolution, and 100 RoIs from the RPN.

that presents comparable tracking results from major vision conferences in 2018 and 2019. As well as including all results from all of these papers we also present additional results from some methods that were either taken from later papers or that we obtained by evaluating open-source code. Sometimes these additional results were different to those presented in the original papers, in which case both results are shown. By evaluating on all of these datasets, and comparing to all methods from these two years, we are able to present a complete and holistic evaluation of our method compared to previous works.

Our method outperforms all previous methods on six out of the 11 evaluated short-term tracking benchmarks, sometimes by up to 7.2 percentage points. On the remaining five benchmarks we achieve close to the best results, with only a few previous methods obtaining better results, and by not too large a margin.

For long-term tracking, Siam R-CNN performs extremely well. Siam R-CNN outperforms all previous methods over all four benchmarks by between 3.9 and 10.1 percentage points.

Figure 3: Qualitative results on OTB2015 [63], LTB35 [39], and DAVIS 2017 [46] (validation set). We compare the results of Siam R-CNN to the best competing methods, which are UPDT [4] for OTB 2015, SiamRPN++ [29] for LTB35, and SiamMask [60] for DAVIS 2017. Siam R-CNN's result is shown in red (magenta for DAVIS 2017), the competing methods' results are shown in blue, and the ground truth in yellow. For each benchmark, the sequences with the best, worst and median relative performance (Δ) between Siam R-CNN and the competing method are shown. Six frames spaced equally throughout each video are shown.

		Short-Term Tracking											Long-Term Tracking			
		Track.Net AUC	GOT10k AUC	NfS AUC	VOT15 EAO	OTB50 AUC	OTB15 AUC	UAV123 AUC	VOT16 EAO	OTB13 AUC	TC128 AUC	VOT17/18 EAO	OxUVA maxGM	LaSOT AUC	UAV20L AUC	LTB35 F
Ours	Δ to SOTA	+7.2	+3.8	+1.9	+0.8	+0.6	+0.0	-0.5	-1.6	-1.8	-2.1	-3.2	+10.1	+7.9	+5.5	+3.9
	Siam R-CNN	81.2	64.9*	63.9	45.4†	66.3	70.1	64.9	46.5†	70.4	60.1	40.8†	72.3	64.8	67.2	66.8
ICCV 2019	DiMP [3]	74.0	61.1	62.0			68.4	65.4				44.0		56.9 / 56.8§		
	UpdateNet [71]	67.7							48.1			39.3		47.5		
	GFS-DCF [66]	60.9					69.3			72.2						
	SPLT [67]												62.2			61.6
	fdKCF [75]						67.5		34.7	70.5		26.5				
	Bridging [22]			51.5			64.7	58.6		65.6						
	GradNet [32]						63.9				55.6	24.7		36.5		
	MLT [10]						61.1			62.1				36.8		
	ARCF [23]							47.3								
CVPR 2019	SiamRPN++ [29]	73.3	45.4§				69.6	61.3 / 64.2§				41.4		49.6		62.9
	ATOM [13]	70.3		59 / 58.4§			67.1§	65 / 64.3§				40.1		51.5 / 51.4§		
	ASRCF [11]						69.2		39.1		60.3	32.8		35.9		
	SPM [56]		51.3§			65.3	68.7		43.4	69.3		33.8				
	SiamMask [60]								44.2			38.7				
	SiamDW [74]				38.0		67.3		37.0	66.6		30.1				
	RPCF [51]						69.6			71.3		31.6				
	C-RPN [17]	66.9					66.3		36.3	67.5		28.9		45.5		
	TADT [34]				32.7		66.0		29.9	68.0	56.2					
	GCT [18]						64.8	50.8		67.0		27.4				
	UDT [57]						63.2		30.1		54.1					
ECCV 2018	UPDT [4]	61.1§		54.1 / 53.7§			70.1§	55 / 54.7§			62.2	37.8				
	DaSiamRPN [76]	63.8§			44.6		65.8§	58.6 / 58.5§	41.1			32.6	41.5§		61.7	60.7§
	ACT [6]					65.7	64.3		27.5	66.3						
	RTINet [70]					63.7	68.2		29.8		60.2					
	SACF [72]				34.3		69.3		38.0	71.3						
	DRL-IS [47]						67.1				59.0					
	DSLT [36]						66.0	53.0	33.2	68.3	58.7					
	Meta-Tracker [44]						66.2		31.7							
	RT-MDNet [24]						65.0	53.5			56.3					
	MemTrack [69]						62.6			64.2						
	StructSiam [73]						62.1		26.4	63.8				33.5§		
	SiamFC-tri [14]					53.5	59.2			62.9		21.3				
CVPR 2018	DRT [49]						69.9		44.2	72.0						
	MCCT [58]						69.5		39.3	71.4	59.6					
	SiamRPN [30]				35.8		63.7		34.4						45.4§	
	STRCF [31]						68.3		31.3		60.1					
	VITAL [48]						68.2		32.3	71.0				39.0§		
	LSART [50]						67.2					32.3				
	FlowTrack [77]				34.1		65.5		33.4	68.9						
	RASNet [59]				32.7		64.1			67.0		28.1				
	SA-Siam [19]				31.0	61.0	65.7		29.1	67.7		23.6				
	TRACA [9]						60.3			65.2						
	MKCF [52]			45.5						64.1						
	HP [15]					55.4	60.1			62.9						
	SINT++ [61]					62.4	57.4									

Table 14: Comparison to all trackers published in CVPR, ICCV and ECCV in 2018 and 2019. Results from original papers, except when marked with § which are from later papers, or from running open-source code. Results in {Red, Green, Blue} are the {Best, Second, Third}, respectively. Benchmarks are ordered by performance relative to the best method other than ours (Δ to SOTA). Methods are ordered first by conference date, then by most 'bests', most 'seconds', most 'thirds' and finally by approximate 'head-to-head' performance. *On all benchmarks Siam R-CNN uses exactly the same network weights and tracking hyper-parameters*, except for those marked with † which use the 'short-term' tracking parameters, and those marked with * which use weights trained only on GOT-10k.

References

[1] L. Bao, B. Wu, and W. Liu. CNN in MRF: video object segmentation via inference in a cnn-based higher-order spatio-temporal MRF. In *CVPR*, 2018. 6, 7

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, 2016. 6

[3] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 3, 4, 6, 9

[4] G. Bhat, J. Johnander, M. Danelljan, F. S. Khan, and M. Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 5, 8, 9

[5] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017. 6, 7

[6] B. Chen, D. Wang, P. Li, S. Wang, and H. Lu. Real-time 'actor-critic' tracking. In *ECCV*, 2018. 5, 9

[7] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, 2018. 7

[8] J. Cheng, Y-H. Tsai, W-C. Hung, S. Wang, and M-H. Yang. Fast and accurate online video object segmentation via tracking parts. In *CVPR*, 2018. 6, 7

[9] J. Choi, H. Jin Chang, T. Fischer, S. Yun, K. Lee, J. Jeong, Y. Demiris, and J. Young Choi. Context-aware deep feature compression for high-speed visual tracking. In *CVPR*, 2018. 9

[10] J. Choi, J. Kwon, and K. M. Lee. Deep meta learning for real-time target-aware visual tracking. In *ICCV*, 2019. 9

[11] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li. Visual tracking via adaptive spatially-regularized correlation filters. In *CVPR*, 2019. 5, 9

[12] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017. 6

[13] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ATOM: accurate tracking by overlap maximization. In *CVPR*, 2019. 3, 4, 9

[14] X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *ECCV*, 2018. 9

[15] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *CVPR*, 2018. 9

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

[17] H. Fan and H. Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019. 9

[18] J. Gao, T. Zhang, and C. Xu. Graph convolutional tracking. In *CVPR*, 2019. 9

[19] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018. 5, 9

[20] Y.-T. Hu, J.-B. Huang, and A. G. Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, 2018. 6, 7

[21] L. Huang, X. Zhao, and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018. 4, 5

[22] L. Huang, X. Zhao, and K. Huang. Bridging the gap between detection and tracking: A unified approach. In *ICCV*, 2019. 9

[23] Z. Huang, C. Fu, Y. Li, F. Lin, and P. Lu. Learning aberrance repressed correlation filters for real-time uav tracking. In *ICCV*, 2019. 9

[24] I. Jung, J. Son, M. Baek, and B. Han. Real-time mdnet. In *ECCV*, 2018. 9

[25] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017. 5

[26] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Č. Zajc, T. Vojir, G. Häger, A. Lukežič, and G. Fernandez. The visual object tracking vot2016 challenge results. In *ECCVW*, 2016. 2, 5, 6

[27] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. C. Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, G. Fernandez, and et al. The sixth visual object tracking VOT2018 challenge results. In *ECCVW*, 2018. 2, 5

[28] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojir, G. Häger, G. Nebehay, R. Pflugfelder, A. Gupta, and et al. The visual object tracking VOT2015 challenge results. In *ICCVW*, 2015. 5

[29] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 3, 4, 6, 8, 9

[30] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 5, 6, 9

[31] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, 2018. 5, 9

[32] P. Li, B. Chen, W. Ouyang, D. Wang, X. Yang, and H. Lu. Gradnet: Gradient-guided network for visual object tracking. In *ICCV*, 2019. 9

[33] X. Li and C. Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *ECCV*, 2018. 6, 7

[34] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang. Target-aware deep tracking. In *CVPR*, 2019. 9

[35] P. Liang, E. Blasch, and H. Ling. Encoding color information for visual tracking: Algorithms and benchmark. *Trans. Image Proc.*, 2015. 5

[36] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang. Deep regression tracking with shrinkage loss. In *ECCV*, 2018. 9

[37] J. Luiten, P. Voigtlaender, and B. Leibe. PReMVOS: Proposal-generation, refinement and merging for the YouTube-VOS challenge on video object segmentation 2018. *ECCVW*, 2018. 7

[38] J. Luiten, P. Voigtlaender, and B. Leibe. PReMVOS: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, 2018. 6, 7

[39] A. Lukežič, L. Č. Zajc, T. Vojíř, J. Matas, and M. Kristan. Now you see me: evaluating performance in long-term visual tracking. *arXiv preprint arXiv:1804.07056*, 2018. 8

[40] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal Taixé, and L. Van Gool. Video object segmentation without temporal information. *PAMI*, 2018. 6, 7

[41] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 5, 6

[42] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 5

[43] S. Wug Oh, J.-Y. Lee, N. Xu, and S. Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019. 6, 7

[44] E. Park and A. C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, 2018. 9

[45] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 6

[46] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 6, 8

[47] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou. Deep reinforcement learning with iterative shift for visual tracking. In *ECCV*, 2018. 9

[48] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. Lau, and M.-H. Yang. VITAL: Visual tracking via adversarial learning. In *CVPR*, 2018. 9

[49] C. Sun, D. Wang, H. Lu, and M.-H. Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, 2018. 5, 9

[50] C. Sun, D. Wang, H. Lu, and M.-H. Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018. 9

[51] Y. Sun, C. Sun, D. Wang, Y. He, and H. Lu. Roi pooled correlation filters for visual tracking. In *CVPR*, 2019. 5, 9

[52] M. Tang, B. Yu, F. Zhang, and J. Wang. High-speed tracking with multi-kernel correlation filters. In *CVPR*, 2018. 9

[53] P. Voigtlaender, Y. Chai, F. Schroff, H. Adam, B. Leibe, and L.-C. Chen. FEELVOS: Fast end-to-end embedding learning for video object segmentation. In *CVPR*, 2019. 6, 7

[54] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for the 2017 DAVIS challenge on video object segmentation. In *CVPRW*, 2017. 7

[55] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 6, 7

[56] G. Wang, C. Luo, Z. Xiong, and W. Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. In *CVPR*, 2019. 5, 9

[57] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li. Unsupervised deep tracking. In *CVPR*, 2019. 9

[58] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, 2018. 5, 9

[59] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *CVPR*, 2018. 9

[60] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 5, 6, 7, 8, 9

[61] X. Wang, C. Li, B. Luo, and J. Tang. Sint++: Robust visual tracking via adversarial positive instance generation. In *CVPR*, 2018. 5, 9

[62] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 5

[63] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *PAMI*, 2015. 5, 8

[64] S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, 2018. 6, 7

[65] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang. YouTube-VOS: Sequence-to-sequence video object segmentation. In *ECCV*, 2018. 4, 7

[66] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler. Joint group feature selection and discriminative filter learning for robust visual object tracking. In *ICCV*, 2019. 5, 9

[67] B. Yan, H. Zhao, D. Wang, H. Lu, and X. Yang. 'Skimming-Perusal' Tracking: A framework for real-time and robust long-term tracking. In *ICCV*, 2019. 9

[68] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, 2018. 6, 7

[69] T. Yang and A. B. Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018. 9

[70] Y. Yao, X. Wu, L. Zhang, S. Shan, and W. Zuo. Joint representation and truncated inference learning for correlation filter based tracking. In *ECCV*, 2018. 5, 9

[71] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, and F. S. Khan. Learning the model update for siamese trackers. In *ICCV*, 2019. 5, 6, 9

[72] M. Zhang, Q. Wang, J. Xing, J. Gao, P. Peng, W. Hu, and S. Maybank. Visual tracking via spatially aligned correlation filters network. In *ECCV*, 2018. 5, 9

[73] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu. Structured siamese network for real-time visual tracking. In *ECCV*, 2018. 9

[74] Z. Zhang, H. Peng, and Q. Wang. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, 2019. 5, 9

[75] L. Zheng, M. Tang, Y. Chen, J. Wang, and H. Lu. Fast-deepkcf without boundary effect. In *ICCV*, 2019. 9

[76] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 5, 6, 9

[77] Z. Zhu, W. Wu, W. Zou, and J. Yan. End-to-end flow correlation tracking with spatial-temporal attention. In *CVPR*, 2018. 5, 9