Robust Visual Tracking by Segmentation

Matthieu Paul, Martin Danelljan, Christoph Mayer, and Luc Van Gool

Computer Vision Lab, ETH Zürich, Switzerland
{paulma, damartin, chmayer, vangool}@vision.ee.ethz.ch

Abstract. Estimating the target extent poses a fundamental challenge in visual object tracking. Typically, trackers are *box-centric* and fully rely on a bounding box to define the target in the scene. In practice, objects often have complex shapes and are not aligned with the image axis. In these cases, bounding boxes do not provide an accurate description of the target and often contain a majority of background pixels.
We propose a *segmentation-centric* tracking pipeline that not only produces a highly accurate segmentation mask, but also works internally with segmentation masks instead of bounding boxes. Thus, our tracker is able to better learn a target representation that clearly differentiates the target in the scene from background content. In order to achieve the necessary robustness for the challenging tracking scenario, we propose a separate instance localization component that is used to condition the segmentation decoder when producing the output mask. We infer a bounding box from the segmentation mask and validate our tracker on challenging tracking datasets and achieve the new state of the art on LaSOT [16] with a success AUC score of 69.7%. Since fully evaluating the predicted masks on tracking datasets is not possible due to the missing mask annotations, we further validate our segmentation quality on two popular video object segmentation datasets. The code and trained models are available at https://github.com/visionml/pytracking.

1 Introduction

Visual object tracking is the task of estimating the state of a target object for each frame in a video sequence. The target is solely characterized by its initial state in the video. Current approaches predominately characterize the state itself with a bounding box. However, this only gives a very coarse representation of the target's state in the image. In practice, objects often have complex shapes, undergo substantial deformations, can be highly elongated, or simply not align well with the image axes. In such cases, the majority of the image content inside the target's bounding box often consists of background regions, thus providing very limited information about the object itself. In contrast, a segmentation mask gives a precise characterization of the object's extent in the image (see Fig. 1 frames #1600 and #3200). Such information is vital in a variety of applications, including video analysis, video editing, and robotics. In this work, we therefore aim to develop an approach capable of accurately and robustly segmenting the target object, even in the highly challenging tracking datasets [16,35].

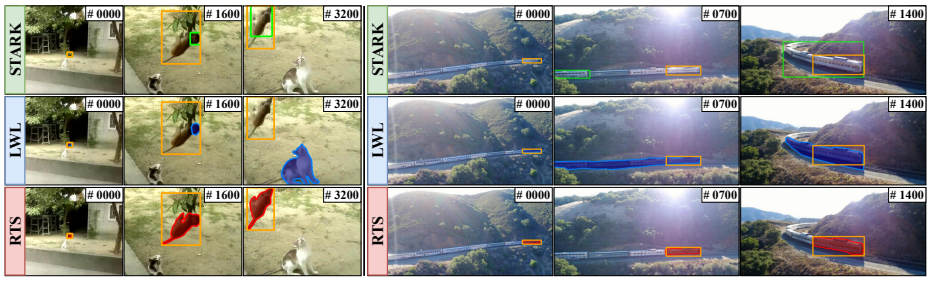

Fig. 1. Comparison between the VOT method Stark [54], the VOS method LWL [5] and our proposed method on two tracking sequences from the LaSOT [16] dataset. The ground-truth annotation (□) is shown in each frame for reference. Our approach is more robust and predicts a more accurate target representation.

While severely limiting the information achieved about the target's state in the video, the aforementioned issues with the bounding box representation can itself lead to inaccurate bounding box predictions, or even tracking failure. Fig. 1 shows two typical tracking sequences. The tracking method STARK [54] (first row) fails to regress bounding boxes that contain the entire object (#1600, #1400) or even starts tracking the wrong object (#0700). Conversely, segmentation masks are a better fit to define the target object in a scene because only pixels corresponding to the target are marked as foreground. Thus, a *segmentation-centric* tracking architecture that is designed to work internally with a segmentation mask of the target instead of a bounding box has the potential to learn better target representations because it can clearly differentiate background from foreground regions in the scene.

A few recent tracking methods [46,53] have recognized the advantage of producing a segmentation mask instead of a bounding box as tracking output. However, these trackers are typically *bounding box-centric* and the final segmentation mask is obtained by a separate *box-to-mask* post-processing network. Thus, these methods miss the opportunity to leverage the accurate target definition of segmentation masks to learn a more accurate and robust internal representation of the target. In contrast, most video object segmentation method [37,5] follow a *segmentation-centric* paradigm. However, these methods are not designed for the challenging tracking scenarios. Typical VOS sequences consist only of a few hundred frames [40] whereas multiple tracking sequence of more than ten thousand frames exist in tracking datasets [16]. Due to this setup, VOS methods focus on producing highly accurate segmentation masks but are sensitive to distractors, substantial deformations and occlusions of the target object. Fig. 1 shows two typical tracking sequences where the VOS method LWL [5] (second row) produces a fine-grained segmentation mask of the wrong object (#3200) or is unable to detect only the target within a crowd (#0700, #1400).

We propose *RTS*, a unified tracking architecture capable of predicting accurate segmentation masks. To design a *segmentation-centric* approach, we take inspiration from the VOS method LWL [5]. However, to achieve robust and accurate segmentation on tracking datasets, we propose several new components.

In particular, we propose an instance localization branch that is trained to predict a target appearance model, which allows the detection of occlusions and to identify the correct target even in cluttered scenes. The output of the instance localization branch is further used to condition the high dimensional mask encoding. This enables the segmentation decoder to focus on the localized target, leading to a more robust mask prediction. Since, our proposed method contains a segmentation and instance memory that needs to be updated with previous tracking results, we design a memory management module. This module first assesses the prediction quality, decides whether the sample should enter into the memory and triggers the tracking model if it should be updated.

Contributions Our contributions are the following: **(i)** We propose a unified tracking architecture capable of predicting robust classification scores and accurate segmentation masks. We design separate feature spaces and memories to ensure optimal receptive fields and update-rates for segmentation and instance localization. **(ii)** To produce a segmentation mask which also agrees with the instance prediction, we design a fusion mechanism that further conditions the segmentation decoder on the instance localization output and leads to more robust tracking performance. **(iii)** We introduce an effective inference procedure capable of fusing the instance localization output and mask encoding to ensure both robust and accurate tracking. **(iv)** We perform comprehensive evaluation and ablation studies of the proposed tracking pipeline on multiple popular tracking benchmarks. Our approach achieves the new state of the art on LaSOT with an *area-under-the-curve* (AUC) score of 69.7%.

2 Related Work

In this section we will revisit recent tracking and video object segmentation methods and particularly focus on existing tracking methods that produce a segmentation mask as output.

Visual Object Tracking Over the years many new challenging tracking benchmarks such as LaSOT [16], GOT-10k [24], and TrackingNet [36] have been proposed that accelerated the tracking research. In particular, Discriminative Correlation Filter (DCF) based, Siamese and more recently transformer based trackers are the most dominant paradigms in visual object tracking.

Visual trackers based on DCFs [6,22,15,32,12,47,60,3,14] are very popular. These methods essentially solve an optimization problem to estimate the weights of the DCF that allow to distinguish foreground from background regions. The DCF is often referred to as the target appearance model and allows to localize the target in the video frame. More recent DCF approaches [3,14] enable end-to-end training by unrolling a fixed number of the optimization iterations during *offline* training. Siamese tracking methods have become more and more popular because they are typically end-to-end trainable, simple and fast [43,2,42,61,20,49,21,29,28]. These trackers aim at learning a similarity metric using only the initial video frame and its annotation that allows to clearly identify the target *offline*. Since no *online* learning component is involved, these trackers

achieve high frame rates at the cost of limited *online* adaptability to changes of the target's appearance. Nonetheless, several methods have been proposed to overcome these issues [43,2,29,28]. The very recently proposed transformer based trackers achieve state of the art performance on many datasets often outperforming DCF or Siamese based trackers. This group of trackers typically uses a Transformer component in order to fuse information extracted from training and test frames to produce discriminative features that allow to accurately localize and estimate the target in the scene [8,55,54,48].

Video Object Segmentation Semi-supervised VOS is the task of classifying all pixels belonging to the target in each video frame, given only the segmentation mask of the target in the initial frame. The cost of annotating accurate segmentation masks is limiting the sequence length and number of videos contained in available VOS datasets. Despite the relatively small size of VOS datasets compared to other computer vision problems, new benchmarks such as Youtube-VOS [52] and DAVIS [40] accelerated the research progress in the last years. A group of methods relies on a learnt target detector [7,45,33] whereas another learns how to propagate the segmentation mask across frames [51,39,30,25]. Other methods use feature matching techniques across one or multiple frames with or without using an explicit spatio-temporal memory [9,23,44,37]. Another method employs meta-learning to tackle VOS [5]. Specifically, Bhat *et al.* [5] introduce an end-to-end trainable VOS architecture employing a few-shot learner that predicts a learnable labels encoding. In particular, the few-shot learner generates and updates *online* the parameters of a segmentation target model that produces the mask encoding used to generate the final segmentation mask.

Joint Visual Tracking and Segmentation A group of tracking methods have already identified the advantages of predicting a segmentation mask instead of a bounding box [53,59,46,31,50,41]. Siam-RCNN is a box-centric tracker that used a pretrained *box2seg* network to predict the segmentation mask given a bounding box prediction. In contrast, AlphaRefine represents a novel *box2seg* method that has been evaluated with many recent trackers such as SuperDiMP [14] or SiamRPN++ [28]. Further, Zhao *et al.* [59] focus on generating segmentation masks from bounding box annotations in videos using a spatio-temporal aggregation module to mine consistencies of the scene across multiple frames. Conversely, SiamMask [50] and D3S [31] are segmentation-centric trackers that produce directly a segmentation output mask without employing a *box2seg* module. In particular, SiamMask [50] consists of a fully-convolutional Siamese network that employs a separate branch to predict binary segmentation masks supervised via a segmentation loss. From a high-level view the single-shot segmentation tracker D3S [31] is most related to our proposed method. Both methods employ two dedicated modules or branches; one for localization and one for segmentation. Whereas D3S adopts the target classification component of ATOM [12] that requires to optimize online the weights of a two-layer CNN, we simply learn online the weights of a DCF similar to DiMP [3]. For segmentation, they propose a feature matching technique that matches test frame features with background and foreground features corresponding to the initial frame. In contrast, we adopt

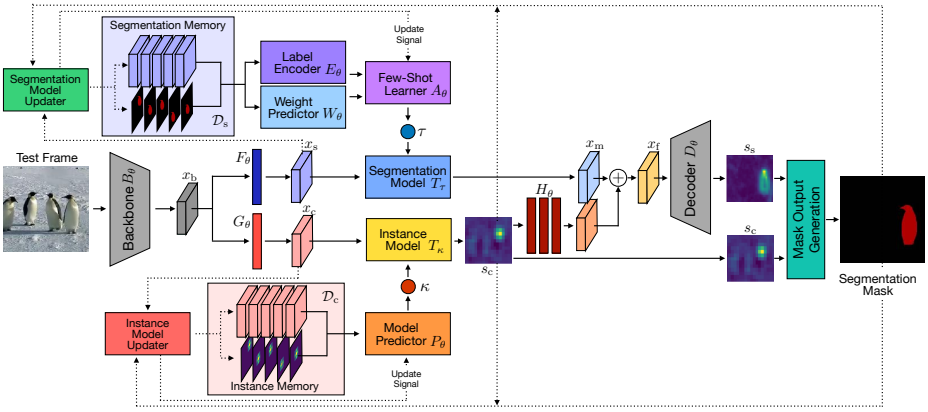

Fig. 2. Overview of our entire online tracking pipeline used for inference, see Sec 3.1.

the few-shot learning based model prediction proposed in LWL [5] to produce accurate segmentation masks. Furthermore, D3S proposes to simply concatenate the outputs of both modules whereas we learn a localization encoding to condition the segmentation mask decoding based on the localization information. Compared to D3S we update not only the instance localization but also the segmentation models and memories. Hence, our method integrates specific memory management components.

3 Method

3.1 Overview

Video object segmentation methods can produce high quality segmentation masks but are typically not robust enough for video object tracking. Robustness becomes vital for medium and long sequences, which are most prevalent in tracking datasets [16,35]. In such scenarios, the target object frequently undergoes substantial appearance changes, also occlusions and similarly looking objects are common. Hence, we propose to adapt a typical VOS approach with tracking components to increase its robustness. In particular, we base our approach on the Learning What to Learn (LWL) [5] method and design a novel and segmentation-centric tracking pipeline that estimates accurate object masks instead of bounding boxes. During inference a segmentation mask is typically not provided in visual object tracking. Hence, we use STA [59] to generate the required initial segmentation mask from the provided initial bounding box. An overview of our tracking method RTS is shown in Fig. 2. Our pipeline consists of a backbone network, a segmentation branch, an instance localization branch and a segmentation decoder. For each video frame, the backbone first extracts a feature map x_b. These features are further processed into segmentation features x_s and classification features x_c to serve as input of their respective branch.

The segmentation branch is designed to capture the details of the object with a high dimensional mask encoding whereas the instance localization branch aims at providing a coarser but robust score map representing the target location. Both branches contain online learned components that are trained on memories ($\mathcal{D}_\mathrm{s}$ and $\mathcal{D}_\mathrm{c}$) that store features and predictions of past frames. The instance localization branch has two purposes: it is used to control the updating of both models and memories and it is used to condition the segmentation mask decoding based on a learned score encoding produced by H_θ in order to inject instance localization information. The obtained segmentation scores and the raw instance model score map are then used to generate the final segmentation mask output.

3.2 Segmentation Branch

The architecture of the segmentation branch is adopted from LWL [5], and we briefly review it here. It consists of a segmentation sample memory $\mathcal{D}_\mathrm{s}$, a label generator E_θ, a weight predictor W_θ, a few-shot learner A_θ and a segmentation model T_τ. The goal of the few-shot learner A_θ is producing the parameters τ of the segmentation model T_τ such that the obtained mask encoding x_m contains the information needed to compute the final segmentation mask of the target object. The label mask encodings used by the few-shot learner are predicted by the label generator E_θ.

The few-shot learner is formulated through the following optimization problem, which is unrolled through steepest descent iterations in the network

$$L_\mathrm{s}(\tau) = \frac{1}{2} \sum_{(x_\mathrm{s}, y_\mathrm{s}) \in \mathcal{D}_\mathrm{s}} \left\| W_\theta(y_\mathrm{s}) \cdot \big(T_\tau(x_\mathrm{s}) - E_\theta(y_\mathrm{s})\big) \right\|^2 + \frac{\lambda_\mathrm{s}}{2} \|\tau\|^2, \tag{1}$$

where $\mathcal{D}_\mathrm{s}$ corresponds to the segmentation memory, x_s denotes the segmentation features, y_s the segmentation masks and λ_s is a learnable scalar regularization parameter. The weight predictor W_θ produces sample confidence weights for each spatial location in each memory sample. Applying the optimized model parameters τ^* within the segmentation model produces the mask encoding $x_\mathrm{m} = T_{\tau^*}(x_\mathrm{s})$ for the segmentation features x_s.

LWL [5] feeds the mask encoding directly into the segmentation decoder to produce the segmentation mask. For long and challenging tracking sequences, only relying on the mask encoding may lead to an accurate segmentation mask, but often for the wrong object in the scene (see Fig 1). Since LWL [5] is only able to identify the target to a certain degree in challenging tracking sequences, we propose to condition the mask encoding based on an instance localization representation, described next.

3.3 Instance Localization Branch

The previously described segmentation branch can produce accurate segmentation masks but typically lacks the necessary robustness for tracking in medium

or long-term sequences. Especially challenging are sequences where objects similar to the target appear, where the target object is occluded or vanishes from the scene for a short time. Therefore, we propose a dedicated branch for target instance localization, in order to robustly identify the target among distractors or to detect occlusions. A powerful tracking paradigm that learns a target specific appearance model on both foreground and background information are discriminative correlation filters (DCF) [6,22,13,3]. These methods learn the weights of a filter that differentiates foreground from background pixels represented by a score map, where the maximal value corresponds to the target's center.

Similar to the segmentation branch, we propose an instance localization branch that consists of a sample memory $\mathcal{D}_\mathrm{c}$ and a model predictor P_θ. The latter predicts the parameters κ of the instance model T_κ. The instance model is trained online to produce the target score map used to localize the target object. To obtain the instance model parameters κ we minimize the following loss function

$$L_\mathrm{c}(\kappa) = \sum_{(x_\mathrm{c}, y_\mathrm{c}) \in \mathcal{D}_\mathrm{c}} \big\| R\big(T_\kappa(x_\mathrm{c}), y_\mathrm{c}\big) \big\|^2 + \frac{\lambda_\mathrm{c}}{2} \|\kappa\|^2, \tag{2}$$

where $\mathcal{D}_\mathrm{c}$ corresponds to the instance memory containing the classification features x_c and the Gaussian labels y_c. R denotes the robust hinge-like loss [3] and λ_c is a fixed regularization parameter. To solve the optimization problem we apply the method from [3], which unrolls steepest descent iterations of the Gauss-Newton approximation of (2) to obtain the final model parameters κ^*. The score map can then be obtained with $s_\mathrm{c} = T_{\kappa^*}(x_\mathrm{c})$ by evaluating the target model on the classification features x_c.

3.4 Instance-Conditional Segmentation Decoder

In video object segmentation the produced mask encoding is directly fed into the segmentation decoder to generate the segmentation mask. However, solely relying on the mask encoding is not robust enough for the challenging tracking scenario, see Fig 1. Thus, we propose to integrate the instance localization information into the segmentation decoding procedure. In particular, we condition the mask encoding on a learned encoding of the instance localization score map.

First, we encode the raw score maps using a multi-layer Convolutional Neural Network (CNN) to learn a suitable representation. Secondly, we simply condition the mask encoding with the learned representation via an element-wise addition. The entire conditioning procedure can be defined as $x_\mathrm{f} = x_\mathrm{m} + H_\theta(s_\mathrm{c})$, where H_θ denotes the CNN encoding the scores s_c, and x_m the mask encoding. The resulting features are then fed into the segmentation decoder that produces the segmentation scores of the target object.

3.5 Jointly Learning Instance Localization and Segmentation

In this section, we describe our general training strategy and parameters. In particular, we further detail the segmentation and classification losses that we use for offline training.

Segmentation Loss First, we randomly sample J frames from an annotated video sequence and sort them according to their frame ids in increasing order to construct the training sequence $\mathcal{V} = \{(x_{\mathrm{b}}^j, y_{\mathrm{s}}^j, y_{\mathrm{c}}^j)\}_{j=0}^{J-1}$, where $x_{\mathrm{b}}^j = B_\theta(I^j)$ are the extracted features of the video frame I^j using the backbone B_θ, y_{s}^j is the corresponding segmentation mask and y_{s}^j denotes the Gaussian label at the target's center location. We start with entry $v_0 \in \mathcal{V}$ and store it in the segmentation $\mathcal{D}_{\mathrm{s}}$ and instance memory $\mathcal{D}_{\mathrm{c}}$ and obtain parameters τ^0 and κ^0 of the segmentation and instance model. We use these parameters to compute the segmentation loss for $v_1 \in \mathcal{V}$. Using the predicted segmentation mask we update the segmentation model parameters to τ^1 but keep the instance model parameters fixed. We use this procedure since the segmentation parameters typically need to be updated frequently to enable accurate segmentation. Conversely, we train the model predictor to produce instance model parameters only on a single frame that generalize to multiple unseen future frames in order to ensure robust target localization. The resulting segmentation loss for the entire sequence $\mathcal{V}$ can thus be described as follows

$$\mathcal{L}_{\mathrm{s}}^{\mathrm{seq}}(\theta; \mathcal{V}) = \sum_{j=1}^{J-1} \mathcal{L}_{\mathrm{s}}\Big(D_\theta\Big(T_{\tau^{j-1}}(x_{\mathrm{s}}^j) + H_\theta\big(T_{\kappa^0}(x_{\mathrm{c}}^j)\big)\Big),\ y_{\mathrm{s}}^j \Big), \tag{3}$$

where $x_{\mathrm{s}} = F_\theta(x_{\mathrm{b}})$ and $x_{\mathrm{c}} = G_\theta(x_{\mathrm{b}})$ and $\mathcal{L}_{\mathrm{s}}$ is the Lovasz segmentation loss [1].

Classification Loss Instead of training our tracker only with the segmentation loss, we add an auxiliary loss to ensure that the instance module produces score maps localizing the target via a Gaussian distribution. Generating such a score map is important because we directly use it to update the segmentation and instance memories and to generate the final output. As explained before, we use only the first training $v_0 \in \mathcal{V}$ to optimize the instance model parameters. Instead of using only the parameters corresponding to the final iteration N_{iter} of the optimization method $\kappa^0_{(N_{\mathrm{iter}})}$ explained in Sec. 3.3 we use all intermediate parameters $\kappa^0_{(i)}$ to compute the loss to encourage fast convergence. The final target classification loss for the whole sequence $\mathcal{V}$ is defined as follows

$$\mathcal{L}_{\mathrm{c}}^{\mathrm{seq}}(\theta; \mathcal{V}) = \sum_{j=1}^{J-1} \left(\frac{1}{N_{\mathrm{iter}}} \sum_{i=0}^{N_{\mathrm{iter}}} \mathcal{L}_{\mathrm{c}}\Big(T_{\kappa^0_{(i)}}(x_{\mathrm{c}}^j),\ y_{\mathrm{c}}^j\Big) \right), \tag{4}$$

where $\mathcal{L}_{\mathrm{c}}$ is the hinge loss defined in [3]. To train our tracker we combine the segmentation and classification losses using the scalar weight η and minimize both losses jointly

$$\mathcal{L}_{\mathrm{tot}}^{\mathrm{seq}}(\theta; \mathcal{V}) = \mathcal{L}_{\mathrm{s}}^{\mathrm{seq}}(\theta; \mathcal{V}) + \eta \cdot \mathcal{L}_{\mathrm{c}}^{\mathrm{seq}}(\theta; \mathcal{V}). \tag{5}$$

Training Details We use the train sets of LaSOT [16], GOT-10k [24], Youtube-VOS [52] and DAVIS [40]. For VOT datasets that only provide annotated bounding boxes, we use these boxes and STA [59] to generate segmentation masks and treat them as ground truth annotations during training. STA [59] is trained separately on YouTube-VOS 2019 [52] and DAVIS 2017 [38]. For our model, we use ResNet-50 with pre-trained MaskRCNN weights as our backbone and initialize the segmentation model and decoder weights with the ones available from LWL [5]. We train for 200 epochs and sample 15'000 videos per epoch, which takes 96 hours to train on a single Nvidia A100 GPU. We use the ADAM [26] optimizer with a learning rate decay of 0.2 at epochs 25, 115 and 160. We weight the losses such that the segmentation loss is predominant but in the same range as the classification. We empirically choose $\eta = 10$. Further details about training and the network architecture are given in the appendix.

3.6 Inference

In this section we describe our inference approach, used during tracking.

Memory Management and Model Updating Our tracker consists of two different memories. The segmentation memory stores segmentation features and predicted segmentation masks of previous frames. In contrast, the instance memory contains classification features and Gaussian labels marking the center location of the target in the predicted segmentation mask of the previous video frame. The quality of the predicted labels directly influences the localization and segmentation quality in future video frames. Hence, it is crucial to avoid contaminating the memories with wrong predictions not corresponding to the actual target. We propose the following strategy to keep the memory as clean as possible. (a) If the instance model is able to clearly localize the target (maximum value in the score map larger than $t_{s_c} = 0.3$) and the segmentation model constructs a valid segmentation mask (at least one pixel above $t_{s_s} = 0.5$) we update both memories with the current predictions and features. (b) If either the instance localization or segmentation fail to identify the target we omit updating the segmentation memory. (c) If only the segmentation mask fails to represent the target but the instance model can localize it, we update the instance memory only. (d) If instance localization fails we omit updating either memory. Further, we trigger the few-shot learner and model predictor after 20 frames have passed but only if the corresponding memory is updated.

Final Mask Output Generation We achieve the final segmentation mask by simply thresholding the segmentation decoder output. To obtain the bounding box required for standard tracking benchmarks, we simply report the smallest axis-aligned box that contains the entire estimated object mask.

Inference Details We set the input image resolution such that the segmentation learner features have a resolution of 52×30 (stride 16), while the instance learner operates of features of size 26×15 (stride 32). The learning rate is set to 0.1 and 0.01 for the segmentation and instance learner respectively. We use a size

of 32 frames for the segmentation memory and 50 frames for the instance memory. We keep the samples corresponding to the initial frame in both memories and replace the oldest entries if the memory is full. We update both memories for the first 100 video frames and afterwards only after every 20^{th} frame. We randomly augment the sample corresponding to the initial frame with vertical flip, random translation and blurring.

4 Evaluation

Our approach is developed within the PyTracking [11] framework. The implementation is done with PyTorch 1.9 with CUDA 11.1. Our model is evaluated on a single Nvidia GTX 2080Ti GPU. Our method runs on average at 30 FPS on LaSOT [16]. The code will be made available upon publication. Each number corresponds to the average of five runs with different random seeds.

4.1 Branch Ablation Study

For the ablation study, we analyze the impact of the instance branch on three datasets and present the results in Tab. 1. First, we report the performance of LWL [5] since we build upon it to design our final tracking pipeline. We use the network weights provided by Bhat *et al.* [5] and the corresponding inference settings. We input the same segmentation masks obtained from the initial bounding box for LWL as used for our method. We observe that LWL is not robust enough for challenging tracking scenarios. The second row in Tab. 1 corresponds to our method but we omit the proposed instance branch. Hence, we use the proposed inference components and settings and train the tracker as explained in Sec. 3.5 but remove the conditioning. We observe that even without the instance localization branch our tracker can achieve competitive performance on all three datasets (*e.g.* +5.6% on LaSOT) but fully integrating the instance localization branch increases the performance even more (*e.g.* +4.4 on LaSOT). Thus, we conclude that adapting the baseline method to the tracking domain improves the tracking performance but to boost the performance and achieve state-of-the-art results an additional component able to increase the tracking robustness is required.

	Seg. Branch	Inst. Branch Conditioning	LaSOT [16]			NFS [19]			UAV123 [35]		
			AUC	P	NP	AUC	P	NP	AUC	P	NP
LWL [5]	✓	-	59.7	60.6	63.3	61.5	75.1	76.9	59.7	78.8	71.4
RTS (No Inst. Branch)	✓	✗	65.3	68.5	71.5	65.8	84.0	85.0	65.2	85.6	78.8
RTS	✓	✓	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6

Table 1. Comparison between our segmentation network baseline LWL and our pipeline, with and without Instance conditioning on different VOT datasets.

Inst. Branch Fallback	t_{s_c}	LaSOT [16] AUC	P	NP	NFS [19] AUC	P	NP	UAV123 [35] AUC	P	NP
✗	0.30	69.3	73.1	75.9	65.3	82.7	84.0	66.3	87.2	80.4
✓	0.30	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6
✓	0.20	68.6	72.3	75.0	65.3	82.7	83.9	67.0	88.7	80.7
✓	0.30	69.7	73.7	76.2	65.4	82.8	84.0	67.6	89.4	81.6
✓	0.40	69.1	72.7	75.6	63.3	79.7	81.7	67.1	89.1	80.7

Table 2. Ablation on inference strategies. The first column analyzes the effect of using the instance branch as fallback for target localization if the segmentation branch is unable to detect the target ($\max(s_s) < t_{s_s}$). The second column shows the impact of different confidence thresholds t_{s_c}.

4.2 Inference Parameters

In this part we ablate two key aspects of our inference strategy. First, we study the effect of relying on the instance branch if the segmentation decoder is unable to localize the target ($\max(s_s) < t_{s_s}$). Secondly, we study different values for t_{s_c} that determines whether the target is detected by the instance model, see Tab. 2.

We observe, that using the instance branch if the segmentation branch cannot identify the target improves the tracking performance on all datasets (*e.g.* +1.3% on UAV123). Furthermore, Tab. 2 shows that our tracking pipeline achieves the best performance when setting $t_{s_c} = 0.3$ whereas smaller or larger values for t_{s_c} decrease the tracking accuracy. Hence, it is important to find a suitable trade-off between frequently updating the model and memory to quickly adapt to appearance changes and updating only rarely to avoid contaminating the memory and model based on wrong predictions.

4.3 Comparison to the state of the art

We compare our approach on six VOT benchmarks and validate the segmentation masks quality on two VOS datasets because assessing the segmentation accuracy on tracking datasets is not possible since only bounding box annotations are provided.

LaSOT [16] We evaluate our method on the test set of the LaSOT dataset which consists of 280 sequences with 2500 frames on average. Thus, the benchmark challenges the long term adaptability and robustness of trackers. Fig. 3 shows the success plot reporting the overlap precision OP with respect to the overlap threshold T. Trackers are ranked by AUC score. In addition, Tab. 3 reports the precision and normalized precision for all compared methods. Our method outperforms the state-of-the-art trackers KeepTrack [34] and Stark-ST101 [54] by a large margin (+2.6% AUC). Our method is not only as robust as KeepTrack (see the success plot for $T < 0.2$) but also estimates far more accurate bounding boxes than any tracker ($0.8 < T < 1.0$).

GOT-10k [24] The large-scale GOT-10k dataset contains over 10.000 shorter sequences. Since we train our method on several datasets instead of only on

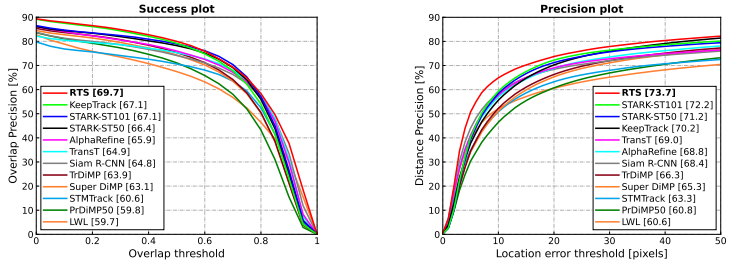

Fig. 3. Success (left) and Precision (right) plots on LaSOT [16] with other state-of-the-art methods. The AUCs for all methods are ordered and reported in the legend. Our method outperforms all existing approaches, both in Overlap Precision (left) and Distance Precision (right).

	RTS	Keep Track [34]	STARK ST-101 [54]	Alpha Refine [53]	TransT [8]	Siam R-CNN [46]	Tr DiMP [48]	Super DiMP [11]	STM Track [18]	Pr DiMP [14]	LWL [5]	DM Track [58]	LTMU [10]	DiMP [3]	Ocean [57]	D3S [31]
Precision	**73.7**	70.2	72.2	68.8	69.0	68.4	66.3	65.3	63.3	60.8	60.6	59.7	57.2	56.7	56.6	49.4
Norm. Prec	76.2	**77.2**	**76.9**	73.8	73.8	72.2	73.0	72.2	69.3	68.8	63.3	66.9	66.2	65.0	65.1	53.9
Success (AUC)	**69.7**	**67.1**	**67.1**	65.9	64.9	64.8	63.9	63.1	60.6	59.8	59.7	58.4	57.2	56.9	56.0	49.2
Δ AUC to Ours	-	↑**2.6**	↑**2.6**	↑**3.8**	↑**4.8**	↑**4.9**	↑**5.8**	↑**6.6**	↑**9.1**	↑**9.9**	↑**10.0**	↑**11.3**	↑**12.5**	↑**12.8**	↑**13.7**	↑**20.5**

Table 3. Comparison to the state of the art on the LaSOT [16] test set in terms of AUC score. The methods are ordered by AUC score.

	RTS	STA [59]	LWL [5]	PrDiMP-50 [14]	DiMP-50 [3]	SiamRPN++ [28]
$SR_{0.50}$(%)	**94.5**	**95.1**	92.4	89.6	88.7	82.8
$SR_{0.75}$(%)	**82.6**	**85.2**	82.2	72.8	68.8	-
AO(%)	**85.2**	**86.7**	84.6	77.8	75.3	73.0

Table 4. Results on the GOT-10k validation set [24] in terms of Average Overlap (AO) and success rates (SR) for overlap thresholds of 0.5 and 0.75.

	RTS	Keep Track [34]	STARK ST101 [54]	STARK ST50 [54]	STA [59]	LWL [5]	TransT [8]	Siam R-CNN [46]	Alpha Refine [53]	STM Track [18]	DTT [55]	Tr DiMP [48]	Super DiMP [11]	Pr DiMP [14]	D3S [31]
Precision	79.4	73.8	-	-	79.1	78.4	**80.3**	**80.0**	78.3	76.7	78.9	73.1	73.3	70.4	66.4
Norm. Prec	86.0	83.5	**86.9**	86.1	84.7	84.4	**86.7**	85.4	85.6	85.1	85.0	83.3	83.5	81.6	76.8
Success (AUC)	**81.6**	78.1	**82.0**	81.3	81.2	80.7	81.4	81.2	80.5	80.3	79.6	78.4	78.1	75.8	72.8
Δ AUC to Ours	-	↑**3.5**	↓**0.4**	↑**0.3**	↑**0.4**	↑**0.9**	↑**0.2**	↑**0.4**	↑**1.1**	↑**1.3**	↑**2.0**	↑**3.2**	↑**3.5**	↑**5.8**	↑**8.8**

Table 5. Comparison to the state of the art on the TrackingNet [36] test set in terms of AUC scores, Precision and Normalized precision.

GOT-10k train, we evaluate our approach on the *val* set only, which consists of 180 short videos. We compile the results in Tab. 4. Our method ranks second for all metrics, falling between two segmentation oriented methods, +0.6% over LWL [5] and −1.5% behind STA [59]. Note, that our tracker outperforms other tracking methods by a large margin.

TrackingNet [36] We compare our approach on the test set of the TrackingNet dataset, consisting of 511 sequences. Tab. 5 shows the results obtained from the online evaluation server. Our method outperforms most of the existing

	RTS	Keep Track [34]	CRACT [17]	STARK ST101 [54]	TrDiMP [48]	TransT [8]	STARK ST50 [54]	Super DiMP [11]	Pr DiMP [14]	STM Track [18]	Siam AttN [56]	Siam R-CNN [46]	KYS [4]	DiMP [3]	LWL [5]
UAV123	67.6	**69.7**	66.4	68.2	67.5	**69.1**	**69.1**	67.7	68.0	64.7	65.0	64.9	–	65.3	59.7
NFS	65.4	**66.4**	62.5	**66.2**	**66.2**	65.7	65.2	64.8	63.5	–	–	63.9	63.5	62.0	61.5

Table 6. Comparison with state-of-the-art on the UAV123 [35] and NFS [19] datasets in terms of AUC score.

	RTS	STARK ST-50 +AR [54]	STARK-ST-101-+AR [54]	LWL [27]	STA [59]	Ocean Plus [27]	Fast Ocean [27]	Alpha Refine [27]	RPT [27]	AFOD [27]	D3S [27]	STM [27]
Robustness	**0.845**	0.817	0.789	0.798	0.824	0.842	0.803	0.777	**0.869**	0.795	0.769	0.574
Accuracy	0.710	**0.759**	**0.763**	0.719	0.732	0.685	0.693	0.754	0.700	0.713	0.699	0.751
EAO	0.506	0.505	0.497	0.463	**0.510**	0.491	0.461	0.482	**0.530**	0.472	0.439	0.308
Δ EAO to Ours	-	**↑0.001**	**↑0.009**	**↑0.043**	**↓0.004**	**↑0.015**	**↑0.045**	**↑0.024**	**↓0.024**	**↑0.034**	**↑0.067**	**↑0.198**

Table 7. Results on the VOT2020-ST [27] challenge in terms of Expected Average Overlap (EAO), Accuracy and Robustness.

approaches and ranks second in terms of AUC, close behind STARK-ST101 [54] which is based on a ResNet-101 backbone. Note, that we outperform STARK-ST50 [54] that uses a ResNet-50 as backbone. We want to highlight that we achieve a higher precision score than other methods that produce a segmentation mask output such as LWL [5], STA [59], Alpha-Refine [53] and D3S [31].

UAV123 [35] The UAV dataset consists of 123 test videos that contain small objects, target occlusion, and distractors. Small objects are particularly challenging in a segmentation setup. Tab. 6 shows the achieved results in terms of success AUC. Our method achieves competitive results on UAV123 similar to TrDiMP [48] or SuperDiMP [11]. Note, that it outperforms LWL [5] by a large margin.

NFS [19] The NFS dataset (30FPS version) contains 100 test videos with fast motions and challenging sequences with distractors. Our method achieves an AUC score that is only 1% below the current best method KeepTrack [34] while outperforming numerous other trackers, including STARK-ST50 [54] (+0.2) SuperDiMP [3] (+0.6) and PrDiMP [14] (+1.9).

VOT 2020 [27] Finally, we evaluate our method on the VOT2020 short-term challenge. It consists of 60 videos and provides segmentation mask annotations. For the challenge, the multi-start protocol is used and the tracking performance is assessed based on accuracy and robustness. We compare with the top methods on the leader board and include more recent methods in Tab. 7. In this situation, our method ranks 2$^{\text{nd}}$ in Robustness, thus outperforming most of the other methods. In particular we achieve a higher EAO score than STARK [54], LWL [5], AlphaRefine [53] and D3S [31].

YouTube-VOS 2019 [52] We use the validation set of Youtube-VOS 2019 [52] which consist of 507 sequences. They contain 91 object categories out of which 26 are *unseen* in the training set. The results presented in Tab. 8 were generated by an online server after uploading the raw results. On this benchmark we are not aiming at achieving the new state of the art but rather validate the quality of the produced segmentation masks.

Method	YouTube-VOS 2019 [52]					DAVIS 2017 [40]		
	$\mathcal{G}$	$\mathcal{J}_{\text{seen}}$	$\mathcal{J}_{\text{unseen}}$	$\mathcal{F}_{\text{seen}}$	$\mathcal{F}_{\text{unseen}}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
RTS	79.7	77.9	75.4	82.0	83.3	80.2	77.9	82.6
LWL [5]	81.0	79.6	76.4	83.8	84.2	81.6	79.1	84.1
STA [59]	80.6	-	-	-	-	-	-	-
STM [37]	79.2	79.6	73.0	83.6	80.6	81.8	79.2	84.3
RTS (Box)	70.8	71.1	65.2	74.0	72.8	72.6	69.4	75.8
LWL (Box) [5]	-	-	-	-	-	70.6	67.9	73.3
Siam-RCNN [46]	67.3	68.1	61.5	70.8	68.8	70.6	66.1	75.0
D3S [50]	-	-	-	-	-	60.8	57.8	63.8
SiamMask [31]	52.8	60.2	45.1	58.2	47.7	56.4	54.3	58.5

Table 8. Results on the Youtube-VOS 2019 [52] and DAVIS 2017 [40] datasets. The table is split in two parts to separate methods using bounding box initialization or segmentation masks initialization, in order to enable a fair comparison.

Hence, we use the same model weight as for VOT without further fine tuning. When using the provided segmentation masks for initialization, we observe that our method performs slighly worse than LWL [5] and STA [59] (-1.3 $\mathcal{G}$, -0.9 $\mathcal{G}$) but still outperforms the VOS method STM [37] (+0.5 $\mathcal{G}$). We conclude that our method can generate accurate segmentation masks. When using bounding boxes to predict the initialization and predict the segmentation masks, we outperform all other methods by a large margin. This confirms that even with our bounding-box initialization strategy our method can produce accurate segmentation masks.

DAVIS 2017 [40] Similarly, we compare our method on the validation set of DAVIS 2017 which contains 30 sequences without fine-tuning the model for this benchmark. The results are shown in Tab. 8 and confirm the the observation made above that our method is able to generate accurate segmentation masks. Our method is competitive for the mask-initialization setup. In the box-initialization however, our approach outperforms all other methods in $\mathcal{J}\&\mathcal{F}$, in particular the segmentation trackers like SiamMask [50] (+16.2) and D3S [31] (+11.8).

5 Conclusion

We introduced RTS, a robust, end-to-end trainable, segmentation-driven tracking method that is able to generate accurate segmentation masks. Compared to the traditional bounding box outputs of classical visual object trackers, segmentation masks enable a more accurate representation of the target's shape and extent. The proposed instance localization branch helps increasing the robustness of our tracker to enable robust tracking even for long sequences consisting of thousands of frames. Our method outperforms by a large margin previous segmentation-driven tracking methods, and it is competitive on several VOT benchmarks. In particular, we set a new state of the art for the challenging LaSOT [16] dataset with a success AUC of 69.7%, outperforming the previous best methods by 2.6%. Competitive results on two VOS datasets confirm the high quality of the generated segmentation masks.

References

1. Berman, M., Triki, A.R., Blaschko, M.B.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 8
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: Proceedings of the European Conference on Computer Vision Workshops (ECCVW) (October 2016) 3, 4
3. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: IEEE/CVF International Conference on Computer Vision, ICCV, Seoul, South Korea. pp. 6181–6190 (2019). https://doi.org/10.1109/ICCV.2019.00628, https://doi.org/10.1109/ICCV.2019.00628 3, 4, 7, 8, 12, 13, 19, 23
4. Bhat, G., Danelljan, M., Van Gool, L., Timofte, R.: Know your surroundings: Exploiting scene information for object tracking. In: Proceedings of the European Conference on Computer Vision (ECCV) (August 2020) 13
5. Bhat, G., Lawin, F.J., Danelljan, M., Robinson, A., Felsberg, M., Gool, L.V., Timofte, R.: Learning what to learn for video object segmentation. In: European Conference on Computer Vision ECCV (2020), https://arxiv.org/abs/2003.11540 2, 4, 5, 6, 9, 10, 12, 13, 14, 19, 21, 22, 23
6. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR (2010) 3, 7
7. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 221–230 (2017) 4
8. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021) 4, 12, 13, 23
9. Chen, Y., Pont-Tuset, J., Montes, A., Van Gool, L.: Blazingly fast video object segmentation with pixel-wise metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1189–1198 (2018) 4
10. Dai, K., Zhang, Y., Wang, D., Li, J., Lu, H., Yang, X.: High-performance long-term tracking with meta-updater. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 12, 23
11. Danelljan, M., Bhat, G.: PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking (2019), accessed: 16/09/2019 10, 12, 13
12. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ATOM: accurate tracking by overlap maximization. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA. pp. 4660–4669 (2019). https://doi.org/10.1109/CVPR.2019.00479 3, 4
13. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: efficient convolution operators for tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2017) 7
14. Danelljan, M., Gool, L.V., Timofte, R.: Probabilistic regression for visual tracking. In: CVPR (2020) 3, 4, 12, 13, 23
15. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: Proceed-

ings of the European Conference on Computer Vision (ECCV) (October 2016) 3
16. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. CoRR **abs/1809.07845** (2018), http://arxiv.org/abs/1809.07845 1, 2, 3, 5, 9, 10, 11, 12, 14, 19, 21, 22, 23, 24
17. Fan, H., Ling, H.: Cract: Cascaded regression-align-classification for robust visual tracking. arXiv preprint arXiv:2011.12483 (2020) 13
18. Fu, Z., Liu, Q., Fu, Z., Wang, Y.: Stmtrack: Template-free visual tracking with space-time memory networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021) 12, 13, 23
19. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV (2017) 10, 11, 13, 21, 22, 23
20. Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., Wang, S.: Learning dynamic siamese network for visual object tracking. In: ICCV (2017) 3
21. He, A., Luo, C., Tian, X., Zeng, W.: Towards a better match in siamese network based visual object tracker. In: ECCV workshop (2018) 3
22. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **37**(3), 583–596 (2015) 3, 7
23. Hu, Y.T., Huang, J.B., Schwing, A.G.: Videomatch: Matching based video object segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 54–70 (2018) 4
24. Huang, L., Zhao, X., Huang, K.: GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019). https://doi.org/10.1109/tpami.2019.2957464, http://dx.doi.org/10.1109/TPAMI.2019.2957464 3, 9, 11, 12, 21
25. Khoreva, A., Benenson, R., Ilg, E., Brox, T., Schiele, B.: Lucid data dreaming for object tracking. In: The DAVIS Challenge on Video Object Segmentation (2017) 4
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980 9
27. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Kämäräinen, J.K., Danelljan, M., Zajc, L.Č., Lukežič, A., Drbohlav, O., He, L., Zhang, Y., Yan, S., Yang, J., Fernández, G., et al: The eighth visual object tracking vot2020 challenge results. In: Proceedings of the European Conference on Computer Vision Workshops (ECCVW) (August 2020) 13
28. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 3, 4, 12
29. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 3, 4
30. Li, X., Change Loy, C.: Video object segmentation with joint re-identification and attention-aware mask propagation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 90–105 (2018) 4

31. Lukezic, A., Matas, J., Kristan, M.: D3s - a discriminative single shot segmentation tracker. In: CVPR (2020) 4, 12, 13, 14
32. Lukezic, A., Vojír, T., Zajc, L.C., Matas, J., Kristan, M.: Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision (IJCV) **126**(7), 671–688 (2018) 3
33. Maninis, K.K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: Video object segmentation without temporal information. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(6), 1515–1530 (2018) 4
34. Mayer, C., Danelljan, M., Paudel, D.P., Gool, L.V.: Learning target candidate association to keep track of what not to track. In: IEEE/CVF International Conference on Computer Vision, ICCV (2021), https://arxiv.org/abs/2103.16556 11, 12, 13, 23, 24
35. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: Proceedings of the European Conference on Computer Vision (ECCV) (October 2016) 1, 5, 10, 11, 13, 21, 22, 23
36. Müller, M., Bibi, A., Giancola, S., Al-Subaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018) 3, 12, 21
37. Oh, S.W., Lee, J.Y., Xu, N., Kim, S.J.: Video object segmentation using space-time memory networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019) 2, 4, 14, 21
38. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Computer Vision and Pattern Recognition (2016) 9
39. Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., Sorkine-Hornung, A.: Learning video object segmentation from static images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2663–2672 (2017) 4
40. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 davis challenge on video object segmentation. arXiv:1704.00675 (2017) 2, 4, 9, 14, 21, 22
41. Son, J., Jung, I., Park, K., Han, B.: Tracking-by-segmentation with online gradient boosting decision tree. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 3056–3064 (2015). https://doi.org/10.1109/ICCV.2015.350 4
42. Tao, R., Gavves, E., Smeulders, A.W.M.: Siamese instance search for tracking. In: CVPR (2016) 3
43. Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 3, 4
44. Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., Chen, L.C.: Feelvos: Fast end-to-end embedding learning for video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9481–9490 (2019) 4
45. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: BMVC (2017) 4
46. Voigtlaender, P., Luiten, J., Torr, P.H., Leibe, B.: Siam R-CNN: Visual tracking by re-detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 2, 4, 12, 13, 14, 23

47. Wang, G., Luo, C., Sun, X., Xiong, Z., Zeng, W.: Tracking by instance detection: A meta-learning approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 3
48. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021) 4, 12, 13, 23
49. Wang, Q., Teng, Z., Xing, J., Gao, J., Hu, W., Maybank, S.J.: Learning attentions: Residual attentional siamese network for high performance online visual tracking. In: CVPR (2018) 3
50. Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H.: Fast online object tracking and segmentation: A unifying approach. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2019) 4, 14
51. Wug Oh, S., Lee, J.Y., Sunkavalli, K., Joo Kim, S.: Fast video object segmentation by reference-guided mask propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7376–7385 (2018) 4
52. Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., Huang, T.: Youtube-vos: A large-scale video object segmentation benchmark (2018) 4, 9, 13, 14, 21, 22
53. Yan, B., Wang, D., Lu, H., Yang, X.: Alpha-refine: Boosting tracking performance by precise bounding box estimation. In: CVPR (2021) 2, 4, 12, 13, 23
54. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10448–10457 (October 2021) 2, 4, 11, 12, 13, 23, 24
55. Yu, B., Tang, M., Zheng, L., Zhu, G., Wang, J., Feng, H., Feng, X., Lu, H.: High-performance discriminative tracking with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9856–9865 (October 2021) 4, 12
56. Yu, Y., Xiong, Y., Huang, W., Scott, M.R.: Deformable siamese attention networks for visual object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 13
57. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: Proceedings of the European Conference on Computer Vision (ECCV) (August 2020) 12
58. Zhang, Z., Zhong, B., Zhang, S., Tang, Z., Liu, X., Zhang, Z.: Distractor-aware fast tracking via dynamic convolutions and mot philosophy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021) 12
59. Zhao, B., Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Generating masks from boxes by mining spatio-temporal consistencies in videos. In: IEEE/CVF International Conference on Computer Vision, ICCV (2021), https://arxiv.org/abs/2101.02196 4, 5, 9, 12, 13, 14, 21
60. Zheng, L., Tang, M., Chen, Y., Wang, J., Lu, H.: Learning feature embeddings for discriminant model based tracking. In: Proceedings of the European Conference on Computer Vision (ECCV) (August 2020) 3
61. Zhu, Z., Wang, Q., Bo, L., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: ECCV (2018) 3

Appendix

In this Appendix, we provide further details on various aspects of our tracking pipeline. First, we provide additional architectural and inference details in Sections A and B. Second, we provide additional ablation studies, in particular on the loss weighting parameter η on different benchmarks to show the importance of the auxiliary instance localization loss in Section C. Then, we provide success plots for different VOT benchmarks as well as a detailed analysis of our results on LaSOT [16] by comparing our approach against the other state-of-the-art methods for all the dataset attributes in Section D. Finally, we provide some additional visual comparison to other trackers in Section E.

A Additional Architecture details

Classification Scores Encoder H_θ First, we describe in Figure A1 the architecture of the Classification Scores Encoder H_θ. It takes as input the $H \times W$-dimensional scores predicted by the Instance Localization (*Classification*) branch and outputs a 16 channels deep representation of those scores. The score encoder consists of a convolutional layer followed by a max-pool layer with stride one and two residual blocks. The output of the residual blocks has 64 channels. Thus, the final convolutional layer reduces the number of channels of the output to 16 to match the encoded scores with the mask encoding. All the convolutional layers use (3×3) kernels with a stride of one to preserve the spatial size of the input classification scores.

Segmentation Decoder D_θ The segmentation decoder has the same structure has in LWL [5]. Together with the backbone it shows a U-Net structure and mainly consists of four decoder blocks. It takes as input the extracted ResNet-50 backbone features and the combined encoding x_f from both the instance localization branch ($H_\theta(s_\mathrm{c})$) and the segmentation branch (x_m), with $x_\mathrm{f} = x_\mathrm{m} + H_\theta(s_\mathrm{c})$. Since the encoded instance localization scores have a lower spatial resolution than the mask encoding x_m, we upscale the encoded instance localization scores using a bilinear interpolation before adding it with the mask encoding x_m. We refer the reader to [5] for more details about the decoder structure.

Segmentation Branch We use the same architectures for the feature extractor F_θ, the label encoder E_θ, the weight predictor W_θ, the few-shot learner A_θ and the segmentation model T_τ as proposed in LWL [5]. Hence, we refer the reader to [5] for more details.

Instance Localization Branch We use the same architectures for the feature extractor G_θ, the model predictor P_θ and the instance model T_κ as proposed in DiMP [3]. Hence, we refer the reader to [3] for more details.

B Additional Inference details

Search region selection The backbone does not extract features on the full image. Instead, we sample a smaller image patch for extraction, which is centered

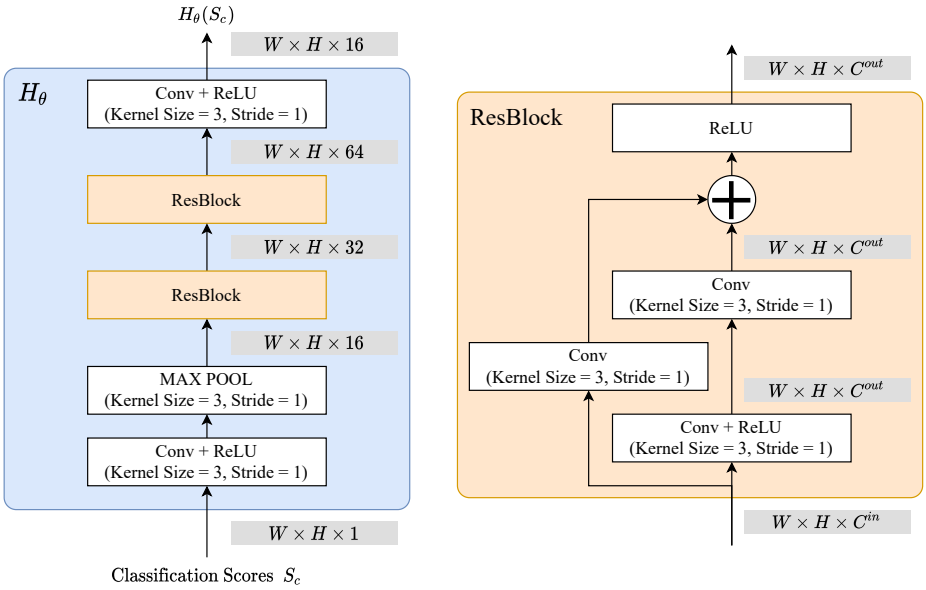

Fig. A1. Classification Scores Encoder H_θ.

at the current target location and 6 times larger than the current estimated target size, when it does not exceed the size of the image. The estimation of the target state (position and size) is therefore crucial to ensure an optimal crop. In most situations, the segmentation output is used to determine the target state since it has a high accuracy. The *target center* is computed as the center of mass of the predicted per-pixel segmentation probability scores. The *target size* is computed as the variance of the segmentation probability scores.

If the segmentation branch cannot find the target (as described in the main paper), but the instance branch still outputs a high enough confidence score, we use it to update the target position. This is particularly important in sequences where the target is becoming too small for some time, but we still can track the target position.

When both branch cannot find the target, the internal state of the tracker is not updated. We upscale the search area based on the previous 60 valid predicted scales. This is helpful in situations where the size of the object shrinks although its size does not change. This typically happens during occlusions or if the target goes out of the frame partially on completely.

C Additional Ablations

In this section, we provide additional ablation studies related to our method, first on the weighting of the segmentation and classification losses used for training,

η	LaSOT [16] AUC	GOT-10k [24] AO	TrackingNet [36] AUC	NFS [19] AUC	UAV123 [35] AUC
0.0	67.7	84.0	81.2	63.7	64.7
0.4	69.8	84.0	81.4	66.2	67.4
10	69.7	85.2	81.6	65.4	67.6

Table A1. Ablation on the classification vs. segmentation loss weighting on different datasets in terms of AUC (area-under-the-curve) and AO (average overlap)

	YouTube-VOS 2019 [52]					**DAVIS 2017 [40]**		
Method	$\mathcal{G}$	$\mathcal{J}_{\text{seen}}$	$\mathcal{J}_{\text{unseen}}$	$\mathcal{F}_{\text{seen}}$	$\mathcal{F}_{\text{unseen}}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
RTS	79.7	77.9	75.4	82.0	83.3	80.2	77.9	82.6
RTS (YT-FT)	80.3	78.8	76.2	82.9	83.5	80.3	77.7	82.9
LWL [5]	81.0	79.6	76.4	83.8	84.2	81.6	79.1	84.1
STA [59]	80.6	-	-	-	-	-	-	-
STM [37]	79.2	79.6	73.0	83.6	80.6	81.8	79.2	84.3

Table A2. Results on the Youtube-VOS 2019 [52] and DAVIS 2017 [40] datasets with a fined tuned model and inference parameters refered as *RTS (YT-FT)*.

second on the parameters that might make a difference specifically for VOS benchmarks like Youtube-VOS [52].

Weighting segmentation and classification losses For this ablation, we study the weighting of the segmentation loss $\mathcal{L}_{\text{s}}$ and the instance localization loss $\mathcal{L}_{\text{c}}$ in the total loss $\mathcal{L}_{\text{tot}}$ used to train our model and its influence on the overall performance during tracking. We recall that

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{s}} + \eta \cdot \mathcal{L}_{\text{c}}. \tag{6}$$

Table A1 shows the results when training the tracker with three different values of η on five VOT datasets. First we examine the case where we omit the auxiliary instance localization loss ($\eta = 0.0$), which means that the whole pipeline is trained for segmentation and the instance branch is not trained to produce specifically accurate localization scores. We observe that for this setting leads to the lowest performance on all tested datasets, often by a large margin. Secondly, we test a dominant segmentation loss ($\eta = 0.4$) because the segmentation branch needs to be trained for a more complex task than the instance branch. We see a performance gain for almost all datasets. Thus, employing the auxiliary loss to train the instance localization branch helps to improve the tracking performance. We observed that using the auxiliary loss leads to localization scores generated during inference that are sharper, cleaner and localize the center of the target more accurately. Finally, we put an even higher weight on the classification term ($\eta = 10$). This setup leads to an even more accurate localization and leads to the on average best results. Thus, we set $\eta = 10$ to train our tracking pipeline.

Finetuning on Youtube-VOS [16] In this section, we analyze whether we can gear our pipeline towards VOS benchmarks. To do that, we take our model and inference parameters and modify them slightly. On the one hand, the model

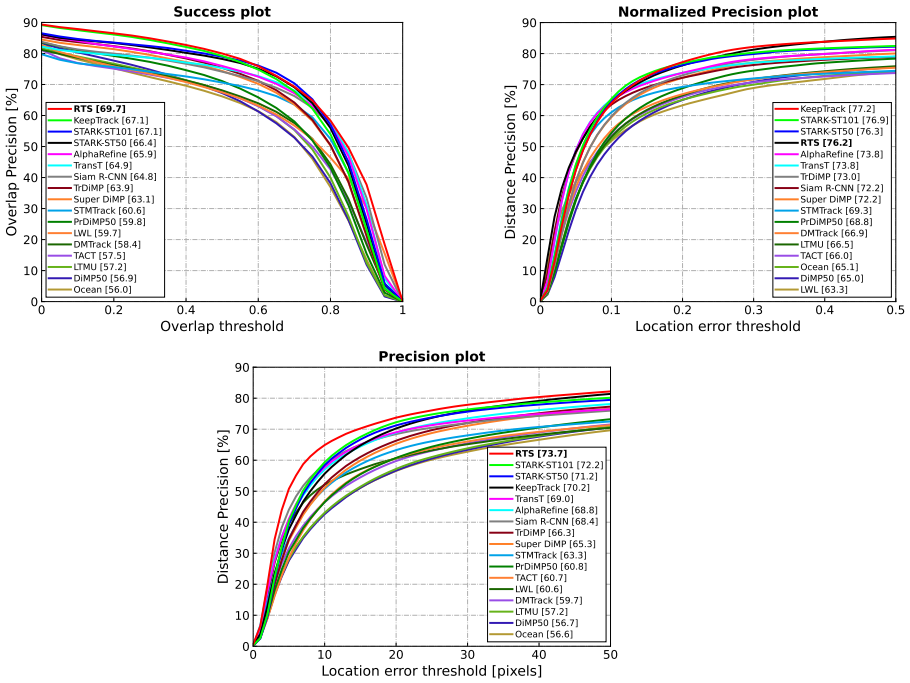

Fig. A2. Success, precision and normalized precision plots on LaSOT [16]. Our approach outperforms all other methods by a large margin in AUC, reported in the legend.

is fined-tuned for 50 epochs using Youtube-VOS [52] only for both training and validation. We also increase the initialization phase from 100 to 200 frames, and remove the relative target scale change limit from one frame to the next (in our model, we limit that scale change to 20% for increased robustness).

The results are presented in Table A2 for Youtube-VOS [52] and Davis [40]. We observe that the performances between both of our models stay very close for Davis but that the finetuned model is getting closer to the baseline LWL [5] for Youtube-VOS. The more frequent updates seem to help and not restricting the scale change of objects from one frame to the next seems to play a role, since we get an improvement of 0.6 in $\mathcal{G}$ score.

D Additional Evaluation results

In this section we provide additional plots of our approach on different benchmarks and a attribute analysis on LaSOT [16].

Success plots for LaSOT [16], NFS [19] and UAV123 [35] We provide in Figure A2 all the plots for the metrics we report for LaSOT [16] in the paper:

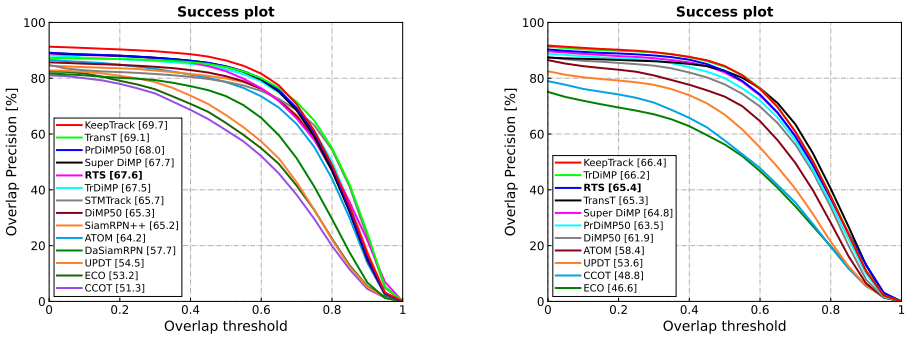

Fig. A3. Success plots on the UAV123 [35] (left) and NFS [19] (right) datasets in terms of overall AUC score, reported in the legend.

	Illumination Variation	Partial Occlusion	Deformation	Motion Blur	Camera Motion	Rotation	Background Clutter	Viewpoint Change	Scale Variation	Full Occlusion	Fast Motion	Out-of-View	Low Resolution	Aspect Ratio Change	Total
LTMU [10]	56.5	54.0	57.2	55.8	61.6	55.1	49.9	56.7	57.1	49.9	44.0	52.7	51.4	55.1	57.2
LWL [5]	65.3	56.4	61.6	59.1	64.7	57.4	53.1	58.1	59.3	48.7	46.5	51.5	48.7	57.9	59.7
PrDiMP50 [14]	63.7	56.9	60.8	57.9	64.2	58.1	54.3	59.2	59.4	51.3	48.4	55.3	53.5	58.6	59.8
STMTrack [18]	65.2	57.1	64.0	55.3	63.3	60.1	54.1	58.2	60.6	47.8	42.4	51.9	50.3	58.8	60.6
SuperDiMP [3]	67.8	59.7	63.4	62.0	68.0	61.4	57.3	63.4	62.9	54.1	50.7	59.0	56.4	61.6	63.1
TrDiMP [48]	67.5	61.1	64.4	62.4	68.1	62.4	58.9	62.8	63.4	56.4	53.0	60.7	58.1	62.3	63.9
Siam R-CNN [46]	64.6	62.2	65.2	63.1	68.2	64.1	54.2	65.3	64.5	55.3	51.5	62.2	57.1	63.4	64.8
TransT [8]	65.2	62.0	67.0	63.0	67.2	64.3	57.9	61.7	64.6	55.3	51.0	58.2	56.4	63.2	64.9
AlphaRefine [53]	69.4	62.3	66.3	65.2	70.0	63.9	58.8	63.1	65.4	57.4	53.6	61.1	58.6	64.1	65.3
KeepTrack Fast [34]	**70.1**	63.8	66.2	65.0	70.7	65.1	60.1	67.6	66.6	59.2	**57.1**	63.4	**62.0**	65.6	66.8
KeepTrack [34]	**69.7**	64.1	67.0	**66.7**	**71.0**	65.3	**61.2**	66.9	**66.8**	**60.1**	**57.7**	**64.1**	**62.0**	**65.9**	**67.1**
STARK-ST101 [54]	67.5	**65.1**	**68.3**	64.5	69.5	**66.6**	57.4	**68.8**	**66.8**	58.9	54.2	63.3	59.6	65.6	**67.1**
RTS	68.7	**66.9**	**71.6**	**67.7**	**74.4**	**67.9**	**61.4**	**69.7**	**69.3**	**60.5**	53.8	**66.3**	**62.7**	**68.2**	**69.7**

Table A3. LaSOT [16] attribute-based analysis. Each column corresponds to the results computed on all sequences in the dataset with the corresponding attribute. Our method outperforms all others in 12 out of 14 attributes.

Success, *Normalized Precision* and *Precision* plots. For completeness, we provide the success plots for NFS [19] and UAV123 [35] in Figure A3.

Attribute analysis on LaSOT [16] In this section, we focus on the dataset sequences attributes. We compare our approach to numerous other trackers, and provide the detailed results in Table A3. Furthermore, we highlight the strength of our approach in Figure A4 by focusing the comparison only to the two current state-of-the-art methods KeepTrack [34] and STARK-ST101 [54].

There are 14 attributes provided for LaSOT [16] sequences, representing different kind of challenges the tracker has to deal with in different situations. Compared to existing trackers, our method achieves better AUC scores in 12 out of 14 attributes. In particular, we outperform KeepTrack [34] and STARK-ST101 [54] by a large margin for the following attributes: *Camera Motion* (+3.4% and +3.9%), *Background Clutter* (+0.2% and +4.0%), *Scale Variation* (+2.5% and 2.5%), *Deformation* (+4.6% and +3.3%) and *Aspect Ratio Change* (+2.3% and +2.6%). Our method is only outperformed on two attributes by KeepTrack [34] and KeepTrack Fast [34] for *Fast Motion* (-3.9% and -3.3%) and for *Illumination Variation* (-1.0% and -1.4%).

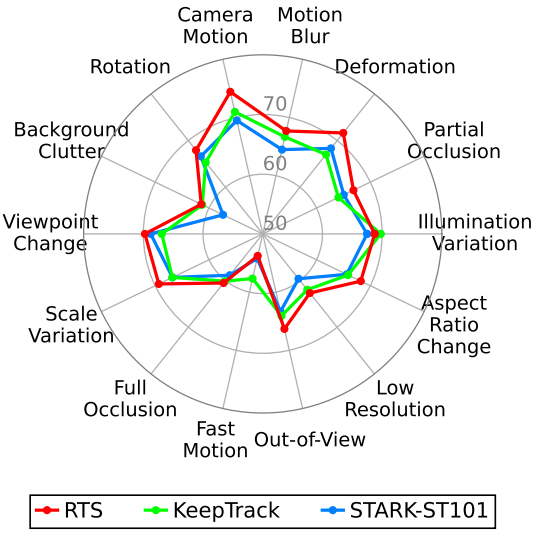

Fig. A4. Attributes comparison on LaSOT [16].

E Additional Content

Figure A5 shows additional visual results compared to other state-of-the-art trackers on 6 different sequences of LaSOT [16]. For more content, we refer the reader to: https://github.com/visionml/pytracking.

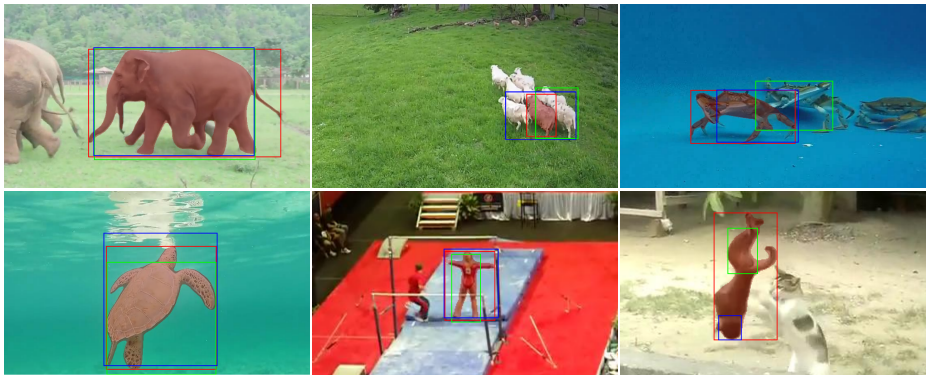

Fig. A5. Qualitative results on LaSOT [16] of our approach compared to the previous state-of-the-art methods KeepTrack [34] and STARK-ST101 [54]. As they do not produce segmentation masks, we represent ours as a red overlay and print for all methods the predicted bounding boxes with the following color code:
KeepTrack STARK-ST101 RTS