Tracking the Known and the Unknown by Leveraging Semantic Information

Ardhendu Shekhar Tripathi[1]
ardhendu-shekhar.tripathi@vision.ee.ethz.ch
Martin Danelljan[1]
martin.danelljan@vision.ee.ethz.ch
Luc Van Gool[1,2]
vangool@vision.ee.ethz.ch
Radu Timofte[1]
radu.timofte@vision.ee.ethz.ch

[1] Computer Vision Lab
ETH Zurich
Switzerland
[2] ESAT-PSI
KU Leuven
Belgium

Abstract

Current research in visual tracking is largely focused on the generic case, where no prior knowledge about the target object is assumed. However, many real-world tracking applications stem from specific scenarios where the class or type of object is known. In this work, we propose a tracking framework that can exploit this semantic information, without sacrificing the generic nature of the tracker. In addition to the target-specific appearance, we model the class of the object through a semantic module that provides complementary class-specific predictions. By further integrating a semantic classification module, we can utilize the learned class-specific models even if the target class is unknown. Our unified tracking architecture is trained end-to-end on large scale tracking datasets by exploiting the available semantic metadata. Comprehensive experiments are performed on five tracking benchmarks. Our approach achieves state-of-the-art performance while operating at real-time frame-rates. The code and the trained models are available at https://tracking.vision.ee.ethz.ch/track-known-unknown/.

1 Introduction

Tracking an object through a video sequence is an important problem in computer vision. The application domain encompasses numerous real-time video analysis systems, including autonomous driving, robotics and augmented reality. Recent years have seen rapid advancements in the visual tracking field. In the area of generic object tracking, where the target is only defined by a bounding box in the first frame, the advancements in accuracy and robustness have been driven by the development of powerful appearance models [8, 10, 26], deep learning [10, 22, 23] and large-scale datasets [12, 18, 29]. Yet, the problem still poses major challenges, in particular when the target undergoes appearance changes or when distractor objects are present.

Despite the growing interest in visual tracking, current research is only focused on the fully generic case, where no prior information about the target object is known. However, this does not fully reflect most real-world applications. In practice, additional information

about the object, such as its *class*, is often available. For instance, in autonomous driving the tracking of vehicles, pedestrians and bicyclists is of crucial importance. In this case, the tracking system could exploit object category knowledge for more robust and accurate tracking. In other examples, the object class is unknown, but could be identified by the tracking framework. This allows the tracker to adapt to the identified class, by *e.g.* exploiting trained class-specific models. In this work, we set out to exploit target class information in order to close the gap between generic and class-specific tracking.

We introduce a tracking framework that is able to utilize the target class information, while preserving the general nature of generic trackers. Our approach is designed to effectively operate in the following two scenarios. (i) When the target class information is available during tracking. (ii) When the class information is not available, in which it is instead estimated by the framework. In both cases, our method benefits from trained class-specific models to aid the tracking. Our approach also generalizes to classes unseen during training by utilizing a powerful target-specific model. This component also ensures instance-specific tracking in the presence of distractor objects of the same class.

Our tracking framework consists of a unified multi-task network architecture. To detect the object, we predict a target-specific model in a Siamese fashion. Class information is integrated by a separate semantic detection module. To accommodate the case when the object class is not given, we incorporate a semantic classifier module that provides the class probabilities. A fusion module combines scores provided by the target-specific and semantic detectors to obtain the final detection score. Finally, we integrate a target estimation component, capable of predicting highly accurate bounding boxes. Our approach is trained in a joint manner on large-scale tracking datasets by exploiting the semantic class information provided as metadata. Comprehensive experiments are performed on five tracking benchmarks: TrackingNet [29], LaSOT [12], VOT2018 [21], UAV123 [28] and NFS [13]. Our approach achieves state-of-the-art results an all datasets while operating at over 30 FPS.

2 Related Work

In recent years, deep learning has led to significant advancements in generic visual tracking. In particular, many Siamese network based approaches [2, 22, 23, 38] offer fast tracking at competitive performance. Most notably, SiamRPN [22] and its recent derivative SiamRPN++ [23] integrate a region proposal network to obtain accurate target bounding boxes, leading to highly impressive performance. While Siamese trackers benefit from their offline learning capabilities, the generated target model provides only limited discriminative power. The latter had already been successfully addressed by correlation filter trackers [1, 4, 6, 8, 9, 17, 26], but at the cost of lower frame-rates and boundary artifacts. More recently, Danelljan *et al*. [5] introduced an efficient approach for learning a discriminative target model online by applying tailored optimization techniques directly in the spatial domain. Moreover, an IoU-Net [19] inspired target estimation module was proposed, capable of predicting accurate bounding boxes. Recent efforts also investigate the aggregation of multiple components for more efficient tracking. Gao *et al*. [14] propose a Gaussian Process based adaptive appearance model which is then combined with a correlation filter component.

Semantic information has been explored in other domains of computer vision, for achieving better performance. Savinov *et al*. [35] formulate a semantic class dependent cost function for optimizing the reprojection error in 3D reconstruction. Sevilla *et al*. in [36] observe

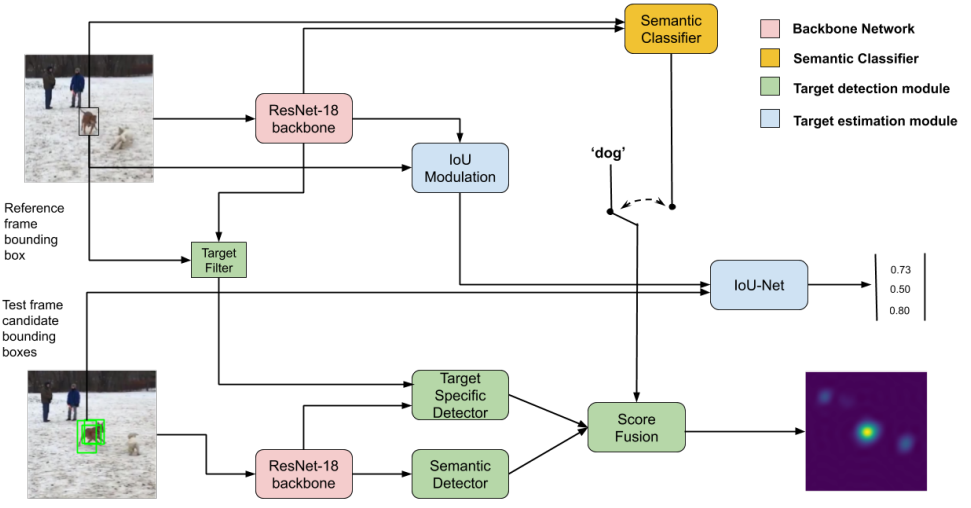

Figure 1: Overview of our network architecture. See section 3 for a detailed description.

that optical flow across an image varies depending on the object classes present in the image. They propose a model based on semantic segmentation for class specific estimation of motion over the whole image. Similarly, Wulff *et al*. [40] exploit rigid scene structure for optical flow estimation by using motion and semantic cues to detect independently moving regions in a scene.

Despite the aforementioned successes, the use of semantic information for generic tracking remains largely unexplored. He *et al*. [15] propose a two fold Siamese network for tracking that combines semantic and appearance features. However, this work only utilizes the features from a network pre-trained for classification. In contrast, our approach is capable of utilizing semantic information available during tracking and *explicitly* models the semantic class of the target. Predicting the class of an object is central in the related field of object detection. Faster R-CNN [33] employs a Region Proposal Network (RPN) for efficient generation and classification of candidate objects. Further, single stage detectors such as SSD [25] and YOLO9000 [32] provide accurate object detection in real-time. In the domain of tracking, single class-specific tracking, of *e.g*. pedestrians or vehicles, has been popularly addressed in a multi-object setting [34, 42]. Such approaches often combine object detection outputs with instance-specific models to associate candidate boxes between frames. However, these approaches are designed to track objects of a single class. Instead, we address the generic tracking setting, while aiming to exploit semantic attributes for improved performance.

3 Proposed Method

In this work, we propose a tracking architecture that is capable of exploiting any semantic class information that may be available during train and inference stages. Our approach is visualized in figure 1. Our architecture contains a target detection and a target estimation module. The detection module is tasked with robustly finding a coarse location of the target,

while the target estimation module aims to refine this estimate to provide an accurate bounding box. Both modules share a backbone feature extraction network, for which we employ a ResNet-18 model [16] that is pre-trained on ImageNet [11].

Our target detection module is composed of two parts, a target-specific detector and a semantic detector. The target-specific detection branch is inspired by Siamese approaches. It predicts a target template filter based on the bounding box and features in the reference frame. This filter is then used on the backbone features of the test frame to compute the target detection scores. On the other hand, the semantic detector provides individual class detection scores using the backbone features from the test frame. The target-specific and semantic detection scores are fused using a dedicated network component, providing the final detection scores used for tracking. In order to accommodate the case when target class is unknown during inference, the fusion module additionally receives class probabilities provided by the classifier module. Finally, target estimation is performed by predicting the bounding box overlap between the target and a set of bounding box estimates. For this task, we adopt the target-specific IoU-Net architecture employed in the ATOM tracker [10]. We detail each of these components in the subsequent subsections.

3.1 Target-specific Detector

The design of this module is inspired by the Siamese approaches to enable end-to-end training of the tracking system. As for Siamese methods, our entire tracking architecture is trained using image pairs $(I_{\text{ref}}, I_{\text{test}})$, consisting of a reference image I_{ref} and test image I_{test} extracted from the same video and containing the target object. The target detector module predicts a convolution filter f based on I_{ref}, that is used to detect the object of interest in I_{test}. The filter is computed using the features $\phi_{\text{TD}}(I_{\text{ref}})$ extracted from the reference image I_{ref}. We employ features extracted from `Block3` of the ResNet-18 backbone.

Features $\phi_{\text{TD}}(I_{\text{ref}})$ from the reference image are first passed through a convolutional layer. We then pool [19] the region corresponding to the target box B_{ref} to a fixed size to generate the filter $f(\phi_{\text{TD}}(I_{\text{ref}}), B_{\text{ref}})$. The filter is applied on features extracted from the test frame $\phi_{\text{TD}}(I_{\text{test}})$ to generate the target specific detection scores. During training, we penalize the squared error between the generated scores and a Gaussian function y_{test} centered at the target in I_{test}. The loss for offline training of the target-specific detector is defined as,

$$L_{\text{TD}} = \left\| h\big(f(\phi_{\text{TD}}(I_{\text{ref}}), B_{\text{ref}}) * \phi_{\text{TD}}(I_{\text{test}}), y_{\text{test}}\big) \right\|^2 , \quad h(s,y) = \begin{cases} s-y, & y > T \\ \max(0,s) & y \le T \end{cases} . \tag{1}$$

Here, $*$ represents the convolution operator. The function h is a robust loss function, that avoids penalizing easy background samples by ignoring negative output scores s in such regions. The threshold T defines the background area as the region $y_{\text{test}} < T$ with ground-truth scores y_{test} lower than T. This allows the learning to focus on the target region and hard negative samples.

3.2 Semantic Detector

The aim of the semantic detector module is to generate individual class detection score maps in a fully convolutional manner. Similar to the target-specific detector, the semantic detector uses features from `Block3` of the ResNet-18 backbone network. The semantic detector head D contains two additional residual blocks for feature extraction followed by a convolutional

layer, generating the final class detection score maps $D(I)$. To train the semantic detector we exploit the metadata for each sequence, containing information about the object class. This annotation is present in most recent tracking datasets [12, 18, 29]. We employ the loss,

$$L_{\text{SD}} = \sum_{c=1}^{C} \left\| w^c_{\text{ref}} \cdot h(D^c(I_{\text{ref}}), \delta_{c,c_{\text{GT}}} \cdot y_{\text{ref}}) \right\|^2 + \sum_{c=1}^{C} \left\| w^c_{\text{test}} \cdot h(D^c(I_{\text{test}}), \delta_{c,c_{\text{GT}}} \cdot y_{\text{test}}) \right\|^2 . \quad (2)$$

Here, c denotes the class index, C the total number of classes considered and the ground-truth target class is denoted as c_{GT}. The semantic detector is trained on both the reference and test frames. For each image, we minimize the squared error between the detection scores $D^c(I)$ and the ground-truth through the robust loss h defined in (1). For the annotated class $c = c_{\text{GT}}$ we set the target scores to the Gaussian y, centered at the target in the image. In all other cases, we set the target scores to zero, as realized by the Kronecker delta function δ.

It is important to note that tracking datasets annotate only the target object. Bounding boxes and class information for other objects potentially present in the images are unknown. The background might therefore contain objects of the considered semantic classes, leading to inaccurate training. To alleviate this problem, we employ a weighting strategy in the loss to reduce the impact of the background region where only limited information is available. We compute the weights w_{ref} and w_{test} as $w^c = 2y$ if $c = c_{\text{GT}}$ and $w^c = y + 1$, otherwise. Here, y is the Gaussian label scores for the frame. This kind of weighting scheme allows us not to suppress other objects of the same class as the tracked object which may be present in the background and are not annotated. On the other hand for the other classes, it is less likely that an object of that particular class is present at the vicinity of the target object. We therefore include background samples in the training (2) for non-target classes $c \neq c_{\text{GT}}$, but with a lower weight.

3.3 Semantic Classifier

For handling the case where the class information is not provided at the time of inference, we include a semantic classifier module. The aim of the classifier is to provide semantic information about the target object, allowing the system to benefit from the semantic detector module for tracking. The semantic classifier uses the features $\phi_{\text{SC}}(I)$ from `Block4` of the ResNet-18 backbone. We first perform average pooling [19] over the target box B, followed by three fully connected layers. The final softmax layer provides the class probabilities $\hat{p}^c(I, B)$ for each class c.

The classification module is trained jointly with the rest of our tracking framework using the class metadata for each sequence. We train the semantic classifier on both the reference and test frame using the standard cross entropy loss,

$$L_{\text{SD}} = -\sum_{c=1}^{C} \delta_{c,c_{\text{GT}}} \log \hat{p}^c(I_{\text{ref}}, B_{\text{ref}}) - \sum_{c=1}^{C} \delta_{c,c_{\text{GT}}} \log \hat{p}^c(I_{\text{test}}, B_{\text{test}}) \quad (3)$$

During inference, classification is only performed on the reference frame.

3.4 Fusion Module

In the fusion module, we strive to combine the detection scores from the target-specific detector and the semantic detector to get a final confidence score for the detection module.

As detailed in section 3.1, the target-specific scores on the test image are obtained as $s_{\text{td}} = f(\phi_{\text{TD}}(I_{\text{ref}}), B_{\text{ref}}) * \phi_{\text{TD}}(I_{\text{test}})$. If the object class c is given during inference, the semantic detection scores are directly given by extracting the appropriate class scores $s_{\text{sd}} = D^c(I_{\text{test}})$.

In the case when semantic information is unavailable during inference, we employ the semantic classification module to extract the class probability vector $\hat{p}(I_{\text{ref}}, B_{\text{ref}})$ from the reference frame. While these probabilities could be used to directly modulate the detection scores $D^c(I_{\text{test}})$, they may contain noise or uncertainties due to class ambiguities or miss-classification in difficult cases. To alleviate these issues, we introduce a class transition matrix K of size $C \times C$. This matrix encodes the conditional probabilities $\mathsf{K}_{ij} = p(c = i|\hat{c} = j)$ of the proper class c represented by the semantic detector, given the class $\hat{c}$ estimated by the classifier. By the matrix multiplication $p = \mathsf{K}\hat{p}(I_{\text{ref}}, B_{\text{ref}})$ we obtain the refined class probabilities p by marginalizing the estimated class variable $\hat{c}$. The conditional probability matrix K models classification noise and correlation between classes, effectively filtering the probability vector generated by the classifier. The filtered probabilities p are then used to compute the expected semantic detection scores as,

$$s_{\text{sd}} = E_{c\sim p}(D^c(I_{\text{test}})) = \sum_{c=1}^{C} p^c D^c(I_{\text{test}}) \,. \tag{4}$$

We employ a linear fusion strategy to obtain the final scores s_{f}. The scalar weights w_{td} and w_{sd} for the target-specific and semantic scores respectively are learned to achieve the optimal final score,

$$s_{\text{f}} = w_{\text{td}} \cdot s_{\text{td}} + w_{\text{sd}} \cdot s_{\text{sd}} \,. \tag{5}$$

We learn the weights $w_{\text{td}}, w_{\text{sd}}$ along with the probability matrix K in our tracking framework during offline training. For the fusion module, we add a corresponding loss,

$$L_{\text{FD}} = \|h(s_{\text{f}}, y_{\text{test}})\|^2 \,. \tag{6}$$

During training, we simulate both the case that the target class is and is not known. For each image pair in the training batch, we randomly sample whether to use the class information in the fusion, with a probability that linearly decays from 1.0 to 0.5 during the course of training. This gives time for the classifier to converge to reasonable accuracy before being employed in the fusion above.

3.5 Target Estimation Module

For the target estimation module, we adopt overlap estimation component from ATOM [10]. The module estimates the target bounding box by predicting the intersection over union (IoU) overlap between the target and an estimate box. The bounding box can then be further refined by maximizing the IoU overlap w.r.t. the bounding box parameters using gradient ascent. The IoU prediction network comprises of two convolutional layers followed by a precise average pool layer [19]. An additional modulation vector encapsulates the target appearance in the reference frame and aids the IoU branch. To train the target estimation module, we follow the procedure in [10], sampling $n = 16$ proposal boxes $\hat{B}_i$ in the test frame and regressing the ground-truth IoU using the mean squared error,

$$L_{TE} = \frac{1}{n}\sum_{i=1}^{n} \left\| P(\hat{B}_i) - \text{IoU}(\hat{B}_i, B_{\text{test}}) \right\|^2 \,. \tag{7}$$

Here, $P(\hat{B}_i)$ is the predicted IoU of $\hat{B}_i$ and $\text{IoU}(\hat{B}_i, B_{\text{test}})$ is the ground truth IoU.

3.6 Training

The network is trained in an end-to-end manner using bounding-box-annotated image pairs, containing the reference and the test image. We additionally exploit the available class information for each pair. The total training loss is achieved by summing up the individual losses presented in the previous sections as,

$$L_{total} = \lambda_{\mathrm{D}}(L_{\mathrm{TD}} + L_{\mathrm{SD}} + L_{\mathrm{FD}}) + \lambda_{\mathrm{C}} L_{\mathrm{SC}} + L_{\mathrm{TE}} \tag{8}$$

We set the weights $\lambda_{\mathrm{D}} = 300$ for the detection losses and $\lambda_{\mathrm{C}} = 1$ for the classification loss.

For training we use the train splits of the LaSOT [12], TrackingNet [29], GOT10k [18] and COCO dataset [24]. We sample batches of image pairs from the videos with a maximum temporal distance of 100 frames. We augment the sampled batches with transformations like blurring and flipping for improving the generalizing power of the trained model. The batch size was set to 80. We train for 50 epochs by sampling 40,000 image pairs per epoch on a single Nvidia TITAN X GPU. We use ADAM [20] with learning rate decay of 0.2 every 15^{th} epoch. The backbone network is pre-trained on ImageNet. The ground truth Gaussian label scores y which are used for training the detection modules (eqs. (1), (2) and (6)) were constructed using a standard deviation of 1/4 relative to size of the test frame bounding box B_{test}. The training and inference modules for this work were implemented in PyTorch [31].

Most recent tracking datasets [12, 18, 29] include class information of the target object. In this work, we condense the diverse and often very fine-grained class annotation from the different datasets to 25 representative classes using the WordNet hierarchy [27]. This includes a class 'other' for encompassing objects that do not map to any of the proper classes. Details of the selected classes are provided in the supplementary material.

3.7 Online Tracking

Tracking is performed by our framework by first predicting the target-specific filter model f (section 3.1) and modulation weights (section 3.5) on the first frame, given the initial target box. For simplicity, we do not update either of these in subsequent frames. If the target class is unknown, we further apply the semantic classifier to obtain the target class probabilities $\hat{p}^c$ in the first frame. Given a new frame, we evaluate the target-specific and semantic detectors to obtain the fused detection scores (5). Finally, we perform target estimation (section 3.5) at the location with the highest fused detection score.

We also propose a version of our approach that integrates efficient online learning (OL) to improve the robustness to distractor objects. This is achieved by replacing the filter f predicted by our architecture with the online learning approach recently introduced by the ATOM tracker [10]. In essence, a 2-layer convolutional network is learned online by minimizing a least-squares regression loss using a Conjugate Gradient based optimization strategy. We utilize the same trained features $\phi_{\mathrm{TD}}(I)$ employed by our target-specific detector. The OL is performed using the same settings as in [10]. Since the fusion module is not trained alongside the ATOM-based target detector, we do not employ it for this version of our tracker. Both versions of our approach achieve over 30 FPS on a single Nvidia TITAN X GPU.

4 Experiments

Here, we provide extensive ablative and state-of-the-art experiments to validate our method.

4.1 Ablation Study

We perform an ablative study on three datasets. To ensure that our tracker is not overfitting on data similar to the training sets or object classes therein, we report results on the combined NFS [13] and UAV123 [28] datasets. These datasets contain a large variety of challenging videos, including novel objects, fast motions, significant appearance changes and distractors. Moreover, we show results on the test split of the challenging LaSOT [12] dataset. The methods are compared in terms of the widely used area-under-the-curve (AUC) measure, computed as the integral of the overlap precision curve (see [39] for details).

Baseline: As a baseline, we train the proposed architecture without the semantic detection, classification and fusion component. Hence, only the target-specific detector (sec. 3.1) and the target estimation components (sec. 3.5) are used. This implies that no semantic annotation is used during training or inference. Thus, this baseline resembles a Siamese tracking pipeline. We denote this version as **Baseline**.

Training with semantic info: This version employs the semantic classifier and the semantic detector, in addition to the baseline. Semantic information is thus used during training. However, the output from the semantic modules are not exploited during inference. This approach, denoted by **+SemInfoTrain**, achieves a significant relative improvement of 1.8% and 4.8% on the NFS+UAV123 and LaSOT datasets respectively compared to the baseline. This shows that including semantic information during training generates more powerful features for generic tracking, providing *e.g.* more inter-class discriminative power.

Fusion: Here we investigate the effect of fusing the semantic and the target-specific detection scores, respectively, using our score fusion module (sec. 3.4). We refer to this version as **+Fusion** in table 1. We evaluate the impact of fusing semantic information during inference in two cases: i) when the target class is unknown during inference (**class unknown**) and ii) when it is known (**class known**). Exploiting the semantic output in our fusion module leads to a further relative improvements of 1.0% and 1.7% on NFS+UAV123 and LaSOT respectively in terms of AUC. This clearly demonstrates the effectiveness of the semantic detector module in our tracking framework. Interestingly, further exploiting the class information during inference did not significantly improve the results. A more detailed investigation showed that this is due to the excellent performance of our semantic classifier, which achieves a top 1 and top 5 accuracy of 93.2% and 99.4%, respectively, on the LaSOT test set. Figure 3 shows qualitative examples of the target-specific detection scores s_{td}, the semantic detection scores s_{sd} and the fusion module s_{f} (see sec. 3.4) on 3 test frames.

	Baseline	+SemInfoTrain	+Fusion class unknown	+Fusion class known
NFS+UAV123	59.7	60.8	61.4	61.5
LaSOT	50.3	52.7	53.6	53.4

Table 1: Ablative study of our approach on the combined NFS and UAV123 datasets and the LaSOT dataset. Results are reported in terms of AUC (%). Analysis of using the semantic information during training (**+SemInfoTrain**) over the Baseline. Fusing the semantic detector prediction with the target detector prediction (**+Fusion**) further boosts the AUC both, in the case when the semantic class is known during inference and when it is not known.

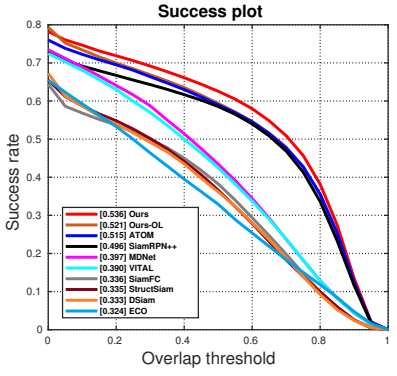

Figure 2: Success plot for LaSOT. The AUC score is shown in the legend. Both our proposed approaches outperform all previous methods.

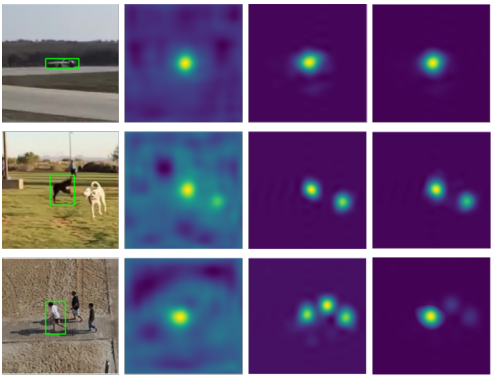

Figure 3: Visualization of detection scores s_{td}, s_{sd} and s_f.

	ECO [8]	SiamFC [3]	CFNet [33]	MDNet [30]	UPDT [6]	DaSiam-RPN [43]	ATOM [10]	SiamRPN++ [23]	**Ours**	**Ours-OL**
Precision (%)	49.2	53.3	53.3	56.5	55.7	59.1	64.8	**69.4**	65.6	**68.0**
Normalized Precision (%)	61.8	66.6	65.4	70.5	70.2	73.3	77.1	**80.0**	76.8	**79.3**
Success (AUC) (%)	55.4	57.1	57.8	60.6	61.1	63.8	70.3	**73.3**	71.5	**73.4**

Table 2: State-of-the-art comparison on the test split of the TrackingNet dataset in terms of precision, normalized precision and success. **Ours-OL** outperforms the previous methods in terms of the AUC measure.

4.2 State-of-the-Art

Here, we compare our method with other state-of-the-art trackers on 5 datasets namely, LaSOT [12], TrackingNet [29], VOT2018 [21], NFS [13] and UAV123 [28]. We report the results for our end-to-end trained approach (**Ours**), along with the version that additionally includes online learning (**Ours-OL**). See section 3.7 for details. For a fair comparison, we assume the target class to be *unknown* in all experiments, i.e. we do not utilize available semantic meta data during tracking. In these experiments, our approach thus only benefits from the semantic annotations available at training time. It should be noted our approach does not require semantic annotations for all employed training data.

LaSOT: We deploy our methods on the test split of the LaSOT [12] dataset, which contains

	DRT [32]	RCO [21]	UPDT [6]	DaSiam-RPN [43]	MFT [21]	LADCF [40]	ATOM [10]	SiamRPN++ [23]	**Ours**	**Ours-OL**
Expected Average Overlap (EAO)	0.356	0.376	0.378	0.383	0.385	0.389	0.401	**0.414**	0.257	**0.413**
Robustness (failure rate)	0.201	0.155	0.184	0.276	**0.140**	**0.159**	0.204	0.234	0.457	0.185
Accuracy (mean IoU)	0.519	0.507	0.536	0.586	0.505	0.503	0.590	**0.600**	0.573	**0.587**

Table 3: State-of-the-art comparison on the VOT2018 dataset in terms of Expected Average Overlap (EAO), robustness and accuracy.

	CCOT[7]	MDNet[30]	ECO[8]	DaSiamRPN[43]	UPDT[3]	ATOM[10]	SiamRPN++[23]	**Ours**	**Ours-OL**
UAV123	51.3	52.8	52.5	58.6	54.5	**64.4**	61.3	**63.5**	62.4
NFS	48.8	42.2	46.6	-	53.7	58.4	-	**59.4**	**60.5**

Table 4: State-of-the-art comparison on the NFS and UAV123 datasets in terms of AUC score (%). **Ours-OL** achieves the best AUC characteristic for the NFS dataset.

280 long and challenging videos. Figure 2 shows the success plots for various trackers on the LaSOT test split. Both versions of our approach achieves top performance, outperforming ATOM with a relative gain of 4.1% and 1.2% for Ours and Ours-OL respectively.

TrackingNet: The test split of the TrackingNet [29] dataset contains 511 videos. The results are obtained through an evaluation server. Table 2 lists the results for the top performing trackers. Ours-OL outperforms ATOM by achieving an impressive AUC score of 73.4%.

VOT2018: Table 3 shows the results on the VOT2018 [21] benchmark. For the VOT2018 dataset, Ours-OL achieves comparable performance to the best performing method SiamRPN++ in terms of the Expected Average Overlap (EAO). It is important to note that SiamRPN++ employs a ResNet-50 backbone network whereas we employ a ResNet-18 backbone for both our methods. Still, Ours-OL is able to outperform SiamRPN++ in terms of rubustness by achieving a 21% reduction in failure rate.

UAV123: The UAV123 dataset contains 123 videos intended for UAV applications. Table 4 states the results on the UAV123 dataset. ATOM achieves the best AUC score of 64.4%. Both proposed trackers, Ours and Ours-OL, achieve competitive performance of 63.5% and 62.4% AUC score respectively, outperforming SiamRPN++ by a significant margin.

Need For Speed (NFS): The 30 FPS version of the NFS [13] dataset, containing 100 videos, is used. Results are shown in table 4. Both, our end-to-end method (Ours) and our online model update based method (Ours-OL) achieve superior results, with relative gains of 1.7% and 3.6% respectively in AUC over the previous best tracker ATOM.

5 Conclusion

We propose an end-to-end trainable architecture for visual tracking, that exploits available semantic class information both during training and inference. Our tracker can operate in the case where the object class is unknown, while also capable of benefiting from semantic information which is available in a multitude of real-world tracking applications. In both cases, our tracking framework utilizes target-specific appearance and semantic modelling to obtain the final target prediction. Moreover, we integrate an online learning strategy to increase the robustness towards distractor objects. We validate our approach through extensive ablative experiments, showing the benefit of training with semantic information and fusing semantic predictions. Lastly, our approach achieves the state-of-the-art results on 4 tracking datasets.

Acknowledgments This work was partly supported by ETH General Fund (OK), Huawei, and Nvidia through a hardware grant. We also thank Goutam Bhat for his inputs and help.

References

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

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *European Conference on Computer Vision ECCV*, pages 493–509, 2018.

[4] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer. Adaptive color attributes for real-time visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1090–1097, 2014.

[5] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.

[6] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1430–1438, 2016.

[7] Martin Danelljan, Andreas Robinson, Fahad Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision ECCV*, pages 472–488, 2016.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 6931–6939, 2017.

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *TPAMI*, 39(8):1561–1575, 2017.

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. *CoRR*, abs/1809.07845, 2018.

[13] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017.

[14] Jin Gao, Qiang Wang, Junliang Xing, Haibin Ling, Weiming Hu, and Stephen John Maybank. Tracking-by-fusion via gaussian process regression extended to transfer learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[15] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015.

[18] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018.

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

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[21] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al. The sixth visual object tracking vot2018 challenge results. In *ECCV workshop*, 2018.

[22] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

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

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

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[26] Alan Lukezic, Tomás Vojír, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *IJCV*, 126(7): 671–688, 2018.

[27] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[28] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016.

[29] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

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

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[32] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[34] Kostia Robert. Night-time traffic surveillance: A robust framework for multi-vehicle detection, classification and tracking. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 1–6. IEEE, 2009.

[35] Nikolay Savinov, Lubor Ladicky, Christian Hane, and Marc Pollefeys. Discrete optimization of ray potentials for semantic 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5511–5518, 2015.

[36] Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3889–3898, 2016.

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

[38] Jack Valmadre, Luca Bertinetto, João F Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[39] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37 (9):1834–1848, 2015.

[40] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4671–4680, 2017.

[41] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual tracking. *CoRR*, abs/1807.11348, 2018.

[42] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Structural constraint data association for online multi-object tracking. *International Journal of Computer Vision*, pages 1–21, 2019.

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