Transforming Model Prediction for Tracking

Christoph Mayer Martin Danelljan Goutam Bhat Matthieu Paul Danda Pani Paudel
Fisher Yu Luc Van Gool
Computer Vision Lab, D-ITET, ETH Zürich, Switzerland

Abstract

Optimization based tracking methods have been widely successful by integrating a target model prediction module, providing effective global reasoning by minimizing an objective function. While this inductive bias integrates valuable domain knowledge, it limits the expressivity of the tracking network. In this work, we therefore propose a tracker architecture employing a Transformer-based model prediction module. Transformers capture global relations with little inductive bias, allowing it to learn the prediction of more powerful target models. We further extend the model predictor to estimate a second set of weights that are applied for accurate bounding box regression. The resulting tracker ToMP relies on training and on test frame information in order to predict all weights transductively. We train the proposed tracker end-to-end and validate its performance by conducting comprehensive experiments on multiple tracking datasets. ToMP sets a new state of the art on three benchmarks, achieving an AUC of 68.5% on the challenging LaSOT [14] dataset. The code and trained models are available at https://github.com/visionml/pytracking

1. Introduction

Generic visual object tracking is one of the fundamental problems in computer vision. The task involves estimating the state of the target object in every frame of a video sequence, given only the initial target location. One of the key problems in object tracking is learning to robustly detect the target object, given a scarce annotation. Among existing methods, Discriminative Correlation Filters (DCF) [1,4,9,10,18,20,29,35] have achieved much success. These approaches learn a target model to localize the target in each frame, by minimizing a discriminative objective function. The target model, often set to a convolutional kernel, provides a compact and generalizable representation of the tracked object, leading to the popularity of DCFs.

The objective function in DCF integrates both foreground and background knowledge over the previous frames, providing effective global reasoning when learning the model. However, it also imposes severe inductive bias on the predicted target model. Since the target model is obtained by solely minimizing an objective over the previous frames, the model predictor has limited flexibility. For instance, it cannot integrate any learned priors in the predicted target model. On the other hand, Transformers have also been shown to provide strong global reasoning across multiple frames, thanks to the use of self and cross attention. Consequently, Transformers have been applied to generic object tracking [6,40,43,45] with considerable success.

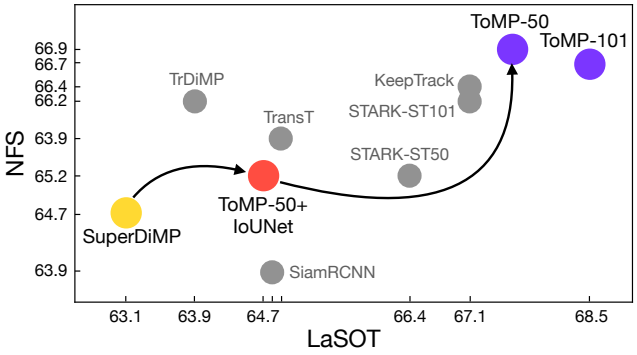


Figure 1. Performance improvements when transforming the model optimizer based tracker SuperDiMP [8] (●) step-by-step. First, we replace the model optimizer by a Transformer based model predictor (●). Secondly, we replace the probabilistic IoUNet by a new regressor and predict its weights with the same model predictor (●). The performance (success AUC) is reported on NFS [17] and LaSOT [14] and compared with recent trackers (●). ToMP-50 and ToMP-101 refer to the different employed backbones ResNet-50 [19] and ResNet-101 [19].

In this work, we propose a novel tracking framework that aims at bridging the gap between DCF and Transformer based trackers. Our approach employs a compact target model for localizing the target, as in DCF. The weights of this model are however obtained using a Transformer-based model predictor, allowing us to learn more powerful target models, compared to DCFs. This is achieved by introducing novel encodings of the target state, allowing the Transformer to effectively utilize this information. We further extend our model predictor to generate weights for a bounding

box regressor network, in order to condition its predictions on the current target. Our proposed approach ToMP obtains significant improvement in tracking performance compared to state-of-the-art DCF-based methods, while also outperforming recent Transformer based trackers (see Fig. 1).

Contributions: In summary, our main contributions are the following: **i)** We propose a novel Transformer-based model prediction module in order to replace traditional optimization based model predictors. **ii)** We extend the model predictor to estimate a second set of weights that are applied for bounding box regression. **iii)** We develop two novel encodings that incorporate target location and target extent allowing the Transformer-based model predictor to utilize this information. **iv)** We propose a parallel two stage tracking procedure at test time to decouple target localization and bounding box regression in order to achieve robust and accurate target detection. **v)** We perform a comprehensive set of ablation experiments to assess the contribution of each building block of our tracking pipeline and evaluate it on seven tracking benchmarks. The proposed tracker ToMP sets a new state of the art on three including LaSOT [14] where it achieves an AUC of 68.5% (see Fig. 1). In addition we show that our tracker ToMP outperforms other Transformer based trackers for every attribute of LaSOT [14].

2. Related Work

Discriminative Model Prediction: DCF based approaches learn a target model to distinguish the target from background by minimizing an objective. For long Fourier-transform based solvers were predominant for DCF based trackers [4, 11, 20, 29]. Danelljan *et al*. [9] employed a two layer Perceptron as target model and used Conjugate Gradient to solve the optimization problem. Recently, multiple methods have been introduced that enable end-to-end training by casting the tracking problem into a meta-learning problem [1, 39, 47]. These methods are based on the idea of unrolling the iterative optimization algorithm for a fixed number of iterations and to integrate it in the tracking pipeline to allow end-to-end training. Bhat *et al*. [1] learn a discriminative feature space and predict the weights of the target model based on the target state in the initial frame and refine the weights with an optimization algorithm.

Transformers for Tracking: Recently, several trackers have been introduced that use Transformers [6, 40, 43, 45]. Transformers are typically employed to predict discriminative features to localize the target object and regress its bounding box. The training features are processed by the Transformer Encoder whereas the Transformer Decoder fuses training and test features using cross attention layers to compute discriminative features [6, 40, 45].

DTT [45] feeds these features to two networks that predict the location and the bounding box of the target. In contrast, TransT [6] employs a feature fusion network that consists of multiple self and cross attention modules. The fused output features are fed into a target classifier and a bounding box regressor. TrDiMP [40] adopts the DiMP [1] model predictor to produce the model weights given the output features of the Transformer Encoder as training samples. Afterwards, the target model computes the target score map by applying the predicted weights on the output features produced by the Transformer Decoder. TrDiMP adopts the probabilistic IoUNet [12] for bounding box regression. Similar to our tracker, TrDiMP encodes target state information but integrates it via two different cross attention modules in the Decoder instead of using two encoding modules in front of the Transformer.

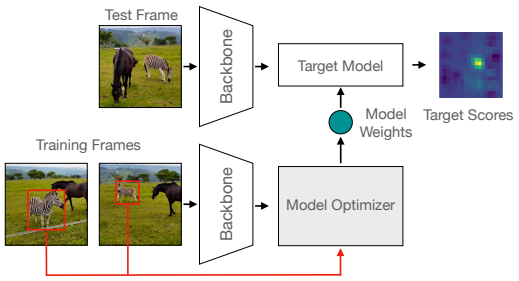
In contrast to the aforementioned Transformer based trackers, STARK [43] adopts the Transformer architecture from DETR [5]. Instead of fusing the training and test features in the Transformer Decoder they are stacked and processed jointly by the full Transformer. A single object-query then produces the Decoder output that is fused with the Transformer Encoder features. These features are then further processed to directly predict the bounding box of the target. In contrast, our tracker employs the same Transformer architecture from DETR [5] but to replace the model optimizer. In the end, our resulting Transformer-based model predictor estimates the weights of two separate models: the target classifier and the bounding box regressor.

3. Method

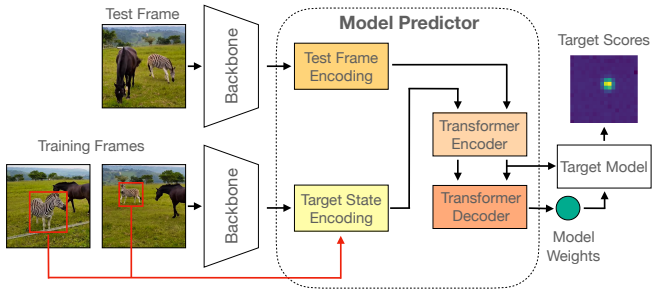
In this work, we propose a Transformer-based target model prediction network for tracking called ToMP. We first revisit existing optimization based model predictors and discuss their limitations in Sec. 3.1. Next, we describe our Transformer-based model prediction approach in Sec. 3.2. We extend this approach to perform joint target classification and bounding box regression in Sec. 3.3. Finally, we detail our offline training procedure and online tracking pipeline in Sec. 3.4 and Sec. 3.5, respectively.

3.1. Background

One of the popular paradigms for visual object tracking is discriminative model prediction based tracking. These approaches, visualized in Fig. 2a, use a target model to localize the target object in the test frame. The weights (parameters) of this target model are obtained from the model optimizer, using the training frames and their annotation. While a variety of target models are used in the literature [1, 9, 23, 29, 35, 39, 47], discriminative trackers share a common base formulation to produce the target model weights. This involves solving an optimization problem such that the target model produces the desired target states $y_i \in \mathcal{Y}$ for the training samples $\mathcal{S}_{\text{train}} \in \{(x_i, y_i)\}_{i=1}^{m}$. Here, $x_i \in \mathcal{X}$ refers to a deep feature map of frame i and m



(a) Tracker with optimization based model prediction.



(b) Proposed tracker with Transformer based model prediction.

Figure 2. Comparison between trackers that employ optimization based model prediction and our Transformer-based model prediction. The model optimizer [ ] in Fig. 2a is replaced by the model predictor in Fig. 2b that consists of the proposed modules [, , ,].

denotes the total number of training frames. The optimization problem reads as follows,

$$w = \arg\min_{\tilde{w}} \sum_{(x,y)\in\mathcal{S}_{\text{train}}} f(h(\tilde{w}; x), y) + \lambda g(\tilde{w}). \quad (1)$$

Here, the objective consists of the residual function f which computes an error between the target model output $h(\tilde{w}; x)$ and the ground truth label y. $g(\tilde{w})$ denotes the regularization term weighted by a scalar λ, while w represents the optimal weights of the target model. Note that the training set $\mathcal{S}_{\text{train}}$ contains the annotated first frame, as well as the previous tracked frames with the tracker's predictions being used as pseudo-labels.

Learning the target model by explicitly minimizing the objective of (1) provides a robust target model that can distinguish the target from the previously seen background. However, such a strategy suffers from notable limitations. The optimization based methods compute the target model using only limited information available in previously tracked frames. That is, they cannot integrate learned priors in the target model prediction so as to minimize *future* failures. Similarly, these methods typically lack the possibility to utilize the current test frame in a transductive manner when computing the model weights to improve tracking performance. The optimization based methods also require setting multiple optimizer hyper-parameters, and can overfit/underfit on the training samples. Another limitation of optimization based trackers is their procedure that produces the discriminative features. Usually, the features provided to the target model are simply the extracted test features. Instead of reinforced features by using the target state information contained in the training frames. Extracting such enhanced features would allow reliable differentiation between the target and background regions in the test frame.

3.2. Transformer-based Target Model Prediction

In order to overcome the aforementioned limitations of optimization based target localization approaches, we propose to replace the model optimizer by a novel target model predictor based on Transformers (see Fig. 2b). Instead of explicitly minimizing an objective as stated in (1), our approach learns to directly predict the target model purely from data by end-to-end training. This allows the model predictor to integrate target specific priors in the predicted model so that it can focus on characteristic features of the target, in addition to the features that allow to differentiate the target from the *seen* background. Furthermore, our model predictor also utilizes the current test frame features, in addition to the previous training features, to predict the target model in a transductive manner. As a result, the model predictor can utilize the current frame information to predict a more suitable target model. Finally, instead of applying the target model on a fixed feature space, defined by the pre-trained feature extractor, our approach can utilize the target information to dynamically construct a more discriminative feature space for every frame.

An overview of the proposed tracker employing the Transformer-based model prediction is shown in Fig. 2b. Similar to the optimization based trackers, it consists of a test and training branch. We first encode the target state information in the training frames and fuse it with the deep image features [ ]. Similarly, we also add an encoding to the test frame in order to mark it as test frame [ ]. The features from both the training and test branches are then jointly processed in the Transformer Encoder [ ] that produces enhanced features by reasoning globally across frames. Next, the Transformer Decoder [ ] predicts the target model weights [ ] using the output of the Transformer Encoder. Finally, the predicted target model is applied on the enhanced test frame features to localize the target. Next, we describe the main components in our tracking pipeline.

Target Location Encoding: We propose a target location encoding that allows the model predictor to incorporate the target state information from the training frames, when predicting the target model. In particular, we use the embedding $e_{\text{fg}} \in \mathbb{R}^{1\times C}$ that represents *foreground*. Together with a Gaussian $y_i \in \mathbb{R}^{H\times W\times 1}$ centered at the target location,

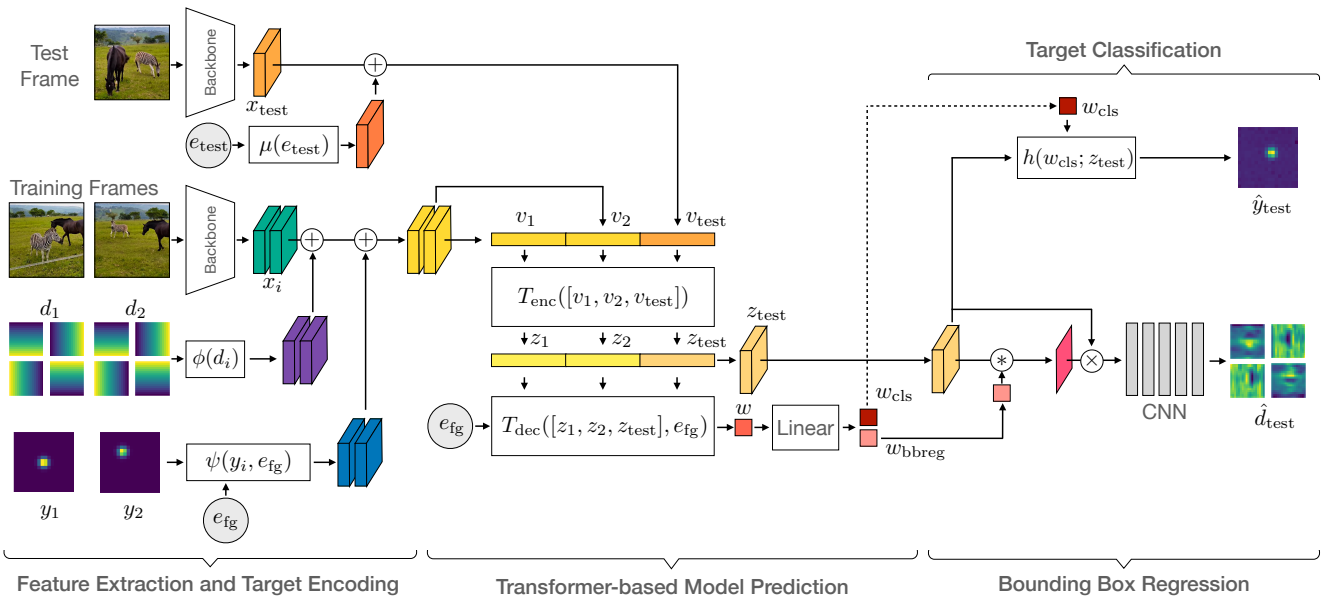


Figure 3. Overview of the entire ToMP tracking pipeline for joint model prediction. First, the training [■] and test [■] features are extracted using a backbone. Then the target location [■] and bounding box [■] encodings are added to the training features. For the test frame the test embedding is encoded [■] and added to the test features. The features are then concatenated and jointly processed by the Transformer-based model predictor that produces the weights used for target classification [■] and bounding box regression [■].

we define the target encoding function

$$\psi(y_i, e_{\text{fg}}) = y_i \cdot e_{\text{fg}}, \tag{2}$$

where "$\cdot$" denotes point-wise multiplication with broadcasting. Note, that $H_{\text{im}} = s \cdot H$ and $W_{\text{im}} = s \cdot W$ correspond to the spatial dimension of the image patch and s to the stride of the backbone network used to extract the deep features $x \in \mathbb{R}^{H \times W \times C}$. Next, we combine the target encoding and the deep image features x as follows

$$v_i = x_i + \psi(y_i, e_{\text{fg}}). \tag{3}$$

This provides us the training frame features $v_i \in \mathbb{R}^{H \times W \times C}$ which contain encoded target state information. Similarly, we also add a test encoding to identify the features corresponding to the test frame as,

$$v_{\text{test}} = x_{\text{test}} + \mu(e_{\text{test}}), \tag{4}$$

where $\mu(\cdot)$ repeats the token e_{test} for each patch of x_{test}.

Transformer Encoder: We aim to predict our target model using the foreground and background information from both the training, as well as the test frames. To achieve this, we use a Transformer Encoder [5, 37] module to first jointly process the features from the training frames and the test frame. The Transformer Encoder serves two purposes in our approach. First, as described later, it computes the features used by the Transformer Decoder module to predict the target model. Secondly, inspired by STARK [43], our Transformer Encoder also outputs enhanced test frame features, which serve as the input to the target model when localizing the target.

Given multiple encoded training features $v_i \in \mathbb{R}^{H \times W \times C}$ and an encoded test feature $v_{\text{test}} \in \mathbb{R}^{H \times W \times C}$, we reshape the features to $\mathbb{R}^{(H \cdot W) \times C}$ and concatenate all m training features v_i and the test feature v_{test} along the first dimension. These concatenated features are then processed jointly in a Transformer Encoder

$$[z_1, \ldots, z_m, z_{\text{test}}] = T_{\text{enc}}([v_1, \ldots, v_m, v_{\text{test}}]). \tag{5}$$

The Transformer Encoder consists of multi-headed self-attention modules [37] that enable it to reason globally across a full frame and even across multiple training and test frames. In addition, the encoded target state identifies *foreground* and *background* regions and enables the Transformer to differentiate between both regions.

Transformer Decoder: The outputs of the Transformer Encoder (z_i and z_{test}) are used as inputs for the Transformer Decoder [5, 37] to predict the target model weights

$$w = T_{\text{dec}}([z_1, \ldots, z_m, z_{\text{test}}], e_{\text{fg}}). \tag{6}$$

Note that the inputs z_i and z_{test} are obtained by jointly reasoning over the whole training and test samples, allowing us to predict a discriminative target model. We use the same learned *foreground* embedding e_{fg} as used for target state encoding as input query of the Transformer Decoder such that the Decoder predicts the target model weights.

Target Model: We use the DCF target model to obtain the target classification scores

$$h(w, z_{\text{test}}) = w * z_{\text{test}}. \tag{7}$$

Here, the weights of the convolution filter $w \in \mathbb{R}^{1\times C}$ are predicted by the Transformer Decoder. Note that the target model is applied on the output test features z_{test} of the Transformer Encoder. These features are obtained after joint processing of training and test frames, and thus support the target model to reliably localize the target.

3.3. Joint Localization and Box Regression

In the previous section, we presented our Transformer based architecture for predicting the target model. Although the target model can localize the object center in each frame, a tracker needs to also estimate an accurate bounding box of the target. DCF based trackers typically employ a dedicated bounding box regression network [9] for this task. While it is possible to follow a similar strategy, we decide to predict both models jointly since target localization and bounding box regression are related tasks that can benefit from one another. In order to achieve this, we extend our model as follows. First, instead of only using the target center location when generating the target state encoding, we also encode target size information to provide a richer input to our model predictor. Secondly, we extend our model predictor to estimate weights for a bounding box regression network, in addition to the target model weights. The resulting tracking architecture is visualized in Fig. 3. Next, we describe each of these changes in detail.

Target Extent Encoding: In addition to the extracted deep image features x_i and the target location encoding $\psi(y_i, e_{\text{fg}})$, we add another encoding to incorporate information about the bounding box of the target. In order to encode the bounding box $b_i = \{b_i^x, b_i^y, b_i^w, b_i^h\}$ encompassing the target object in the training frame i, we adopt the *ltrb* representation [16, 36, 42, 45]. First, we map each location (j^x, j^y) on the feature map x_i back to the image domain using $(k^x, k^y) = (\lfloor \frac{s}{2} \rfloor + s \cdot j^x, \lfloor \frac{s}{2} \rfloor + s \cdot j^y)$. Then, we compute the normalized distance of each remapped location to the four sides of the bounding box b_i as follows,

$$\begin{aligned} l_i &= (k^x - b_i^x)/W_{\text{im}}, \quad r_i = (k^x - b_i^x - b_i^w)/W_{\text{im}}, \\ t_i &= (k^y - b_i^y)/H_{\text{im}}, \quad b_i = (k^y - b_i^y - b_i^h)/H_{\text{im}}, \end{aligned} \tag{8}$$

where $W_{\text{im}} = s \cdot W$ and $H_{\text{im}} = s \cdot H$. These four sides are used to produce the dense bounding box representation $d = (l, t, r, b)$, where $d \in \mathbb{R}^{H\times W\times 4}$. In this representation, we encode the bounding box using a Multi-Layer Perceptron (MLP) ϕ and thereby increase the number of dimensions from 4 to C before adding the obtained encoding to Eq. (3) such that

$$v_i = x_i + \psi(y_i, e_{\text{fg}}) + \phi(d_i). \tag{9}$$

Here, v_i is the resulting feature map which is used as input to the Transformer Encoder, see Fig. 3.

Model Prediction: We extend our architecture to predict weights for the target model, as well as bounding box regression. Concretely, we pass the output w of the Transformer Decoder through a linear layer to obtain the weights for bounding box regression w_{bbreg} and target classification w_{cls}. The weights w_{cls} are then directly used within the target model $h(w_{\text{cls}}; z_{\text{test}})$ as before. The weights w_{bbreg}, on the other hand, are used to condition the output test features z_{test} of the Transformer Encoder with target information for bounding box regression, as explained next.

Bounding Box Regression: To make the encoder output features z_{test} target aware, we follow Yan *et al.* [43] and first compute an attention map $w_{\text{bbreg}} * z_{\text{test}}$ using the predicted weights w_{bbreg}. The attention weights are then multiplied point-wise with the test features z_{test} before feeding them into a Convolutional Neural Network (CNN). The last layer of the CNN uses an exponential activation function to produce the normalized bounding box prediction in the same *ltrb* representation as described in Eq. (8). In order to obtain the final bounding box estimation, we first extract the center location by applying the $\text{argmax}(\cdot)$ function on the target score map $\hat{y}_{\text{test}}$ predicted by the target model. Next, we query the dense bounding box prediction $\hat{d}_{\text{test}}$ at the center location of the target object to obtain the bounding box. We use two dedicated networks for target localization and bounding box regression in contrast to Yan *et al.* [43] that uses one network trying to predict both. This allows us as explained in Sec. 3.5 to decouple target localization from bounding box regression during tracking.

3.4. Offline Training

In this section, we describe the protocol to train the proposed tracker ToMP. Similar to recent end-to-end trained discriminative trackers [1, 12], we sample multiple training and test frames from a video sequence to form training subsequences. In particular, we use two training frames and one test frame. In contrast to recent Transformer based trackers [6,43,45] but similar to DCF based trackers [1,9,12], we keep the same spatial resolution for training and test frames. We pair each image I_i with the corresponding bounding box b_i. We use the target state of the training frames to encode target information and use the bounding box of the test frame only to supervise training by computing two losses based on the predicted bounding boxes and the derived center location of the target in the test frame.

We employ the target classification loss from DiMP [1] that consists of different losses for background and foreground regions. Further, we employ the generalized Intersection over Union loss [33] using the *ltrb* bounding box representation [36] to supervise bounding box regression

$$L_{\text{tot}} = \lambda_{\text{cls}} L_{\text{cls}}(\hat{y}, y) + \lambda_{\text{giou}} L_{\text{giou}}(\hat{d}, d), \tag{10}$$

where λ_{cls} and λ_{giou} are scalars weighting the contribution of each loss. Note that in contrast to FCOS [36] and related trackers [16] we omit an additional centerness loss since

it would be redundant in addition to our classification loss that serves the same purpose. A detailed study examining the impact of centerness is available in the supplementary.

Training Details: We train our tracker on the training splits of the LaSOT [14], GOT10k [21], Trackingnet [32] and MS-COCO [27] datasets. We sample 40k sub-sequences and train for 300 epochs on two Nvidia Titan RTX GPUs. We use ADAMW [28] with a learning rate of 0.0001 that we decay by a factor of 0.2 after 150 and 250 epochs and weight decay of 0.0001. We set $\lambda_{\text{cls}} = 100$ and $\lambda_{\text{giou}} = 1$. We construct a training sub-sequence by randomly sampling two training frames and a test frame from a 200 frame window within a video sequence. We then extract the image patches after randomly translating and scaling the image relative to the target bounding box. Moreover, we use random image flipping and color jittering for data augmentation. We set the spatial resolution of the target scores to 18×18 and set the search area scale factor to 5.0. Further training and architecture details are provided in the supplementary.

3.5. Online Tracking

During tracking, we use the annotated first frame, as well as previously tracked frames as our training set $\mathcal{S}_{\text{train}}$. While we always keep the initial frame and its annotation, we include one previously tracked frame and replace it with the most recent frame that achieves a target classifier confidence higher than a threshold. Hence, the training set $\mathcal{S}_{\text{train}}$ contains at most two frames.

We observed that incorporating previous tracking results in $\mathcal{S}_{\text{train}}$ improves the target localization considerably. However, including predicted bounding box estimations degrades the bounding box regression performance due to inaccurate predictions, see Sec. 4.1. Hence, we run the model predictor twice. First, we include intermediate predictions in $\mathcal{S}_{\text{train}}$ to obtain the classifier weights. In the second pass, we only use the annotated initial frame to predict the bounding box. Note that for efficiency both steps can be performed in parallel in a single forward pass. In particular, we reshape the feature map corresponding to two training and one test frame to a sequence and duplicate it. Then, we stack both in the batch dimension to process them jointly with the model predictor. To only allow attention between the initial frame with ground truth annotation and the test frame when predicting the model for bounding box regression, we make use of the so-called *key_padding_mask* that allows us to ignore certain keys when computing attention.

4. Experiments

We evaluate our proposed tracking architecture ToMP on seven benchmarks. Our approach is based on PyTorch 1.7 and is developed within the PyTracking [8] framework. PyTracking is available under the GNU GPL 3.0 license. On a single Nvidia RTX 2080Ti GPU, ToMP-101 and ToMP-50 achieve 19.6 and 24.8 FPS and use a ResNet-101 [19] and ResNet-50 [19] as backbone respectively.

4.1. Ablation Study

We perform a comprehensive analysis of the proposed tracker. First, we analyze the contribution of the different proposed target state encodings and then examine the effect of different inference settings. Finally, we report the performance achieved when replacing the target classifier or the bounding box regressor of SuperDiMP with ours. All ablation experiments in this part use a ResNet-50 as backbone.

Target State Encoding: In order to analyze the effect of the different target state encodings we train different variants of our network and evaluate them on multiple datasets. The first five rows of Tab. 1 correspond to versions with different target location encodings. All other settings are kept the same. In addition to the foreground and test embedding, we include a learned background embedding (instead of setting $e_{\text{bg}} = 0$) to our analysis as follows: $\psi(y_i, e_{\text{fg}}, e_{\text{bg}}) = y_i \cdot e_{\text{fg}} + (1 - y_i) \cdot e_{\text{bg}}$. However, Tab. 1 shows (4$^{\text{th}}$ vs. 5$^{\text{th}}$ row) that adding such a learned background embedding decreases the tracking performance. We further observe that setting the foreground embedding $e_{\text{fg}} = 0$ (1$^{\text{st}}$ row) and only relying on the target extent encoding $\phi(\cdot)$ still achieves high tracking performance but clearly lacks behind all other versions that include the foreground embedding. We conclude that using only the foreground encoding e_{fg} and the test encoding e_{test} leads to the best performance (4$^{\text{th}}$ row).

In the second part of Tab. 1 we choose the best settings for the target location encoding and remove either the target extent encoding $\phi(\cdot)$ or decouple the Transformer Decoder query from the foreground embedding e_{fg}. We observe that using a separate query (6$^{\text{th}}$ row) decreases the overall performance. Similarly, we notice that incorporating target extent information via the proposed encoding is crucial. Otherwise, the performance drops significantly (7$^{\text{th}}$ row).

Model Predictor: Since our model predictor estimates two different model weights, it seems natural to use two different Transformer queries: one to produce the target model

	e_{fg}	e_{bg}	e_{test}	$\phi(\cdot)$	$q_{\text{dec}} = e_{\text{fg}}$	LaSOT	NFS	OTB
①	✗	✗	✗	✓	n.a.	66.0	64.8	68.2
②	✓	✗	✗	✓	✓	67.1	**66.6**	**70.0**
③	✓	✓	✗	✓	✓	67.1	66.3	69.4
④	✓	✗	✓	✓	✓	**67.6**	**66.9**	**70.1**
⑤	✓	✓	✓	✓	✓	**67.4**	66.0	69.5
⑥	✓	✗	✓	✓	✗	66.0	66.2	69.9
⑦	✓	✗	✓	✗	✓	63.1	64.2	64.0

Table 1. For e_{fg}, e_{bg} and e_{test} learning the embedding is denoted by ✓ whereas ✗ means setting it to zero. Using the encoding $\phi(\cdot)$ is denoted by ✓ whereas ✗ refers to omitting it. For $q_{\text{dec}} = e_{\text{fg}}$ the symbol ✓ means sharing the learned embedding e_{fg} for encoding and querying the Decoder wheres ✗ means learning two separate embeddings for both tasks. (Our final model is in the 4$^{\text{th}}$ row).

Number of Decoder queries	Linear Layer	Decoder query q_{dec}	LaSOT	NFS	OTB
1	✓	$q_{\text{dec}} = e_{\text{fg}}$	**67.6**	**66.9**	**70.1**
2	✗	$q_{\text{dec}} \neq e_{\text{fg}}$	63.7	62.8	67.9

Table 2. Analysis of different model predictor architectures and its impact on the tracking performance in terms of success AUC.

Two Stage Model Prediction	Previous Tracking Results	LaSOT	NFS	OTB
n.a.	✗	65.7	65.3	67.8
✓	✓	**67.6**	**66.9**	**70.1**
✗	✓	62.0	64.8	62.8

Table 3. Analysis of different inference settings an of their impact on the tracking performance in terms of success AUC.

Model Predictor	Bounding Box Regressor	LaSOT	NFS	UAV	LaSOT ExtSub
DiMP [1]	Prob. IoUNet [12]	63.1	64.8	**67.7**	43.7
ToMP	Prob. IoUNet [12]	**64.7**	**65.2**	65.0	**45.2**
ToMP	**ToMP**	**67.6**	**66.9**	**69.0**	**45.4**

Table 4. Impact of replacing DiMP [1] and the probabilistic IoUNet [12] with ToMP for localization and box regression.

weights and the other to obtain the bounding box regressor weights. However, this involves decoupling the query from the foreground embedding e_{fg} and the experiments in Tab. 2 show a significant performance drop for this case.

Inference Settings: During online tracking, we use the initial frame and its annotation as training frames. In addition, we include the most recent frame and its target prediction if the classifier confidence is above a certain threshold. Tab. 3 shows that including previous tracking results leads to higher tracking performance than using only the initial frame. Disabling the described two stage model prediction approach and predicting the weights of the target model and bounding box regressor at once decreases the tracking performance drastically (-5.6 AUC on LaSOT). The reason is the sensitivity of the bounding box predictor to inaccurate predicted boxes that are encoded and used for training.

Transforming Model Prediction Step-by-Step: Our model predictor can estimate model weights for the target model and bounding box regressor. In this part, we will transform an optimization based tracker step-by-step to assess the impact of each transformation step. Tab. 4 shows that replacing the model optimizer in SuperDiMP (1st row) with our proposed model predictor to only predict the target model (2nd row) outperforms SuperDiMP on three out of four datasets. Our tracker ToMP that jointly predicts model weights for target localization and bounding box regression (3rd row) achieves the best performance on all four datasets. We conclude that predicting the weights of the target model improves the performance and likewise predicting the weights of the bounding box regressor. Note that we report the average over five runs for all trackers based on the probabilistic IoUNet due to its stochasticity.

4.2. Comparison to the State of the Art

We compare our tracker ToMP on seven tracking benchmarks. The same settings and parameters are used for all datasets. We recompute the metrics of all trackers using the raw predictions if available or otherwise report the results given in the respective paper.

LaSOT [14]: First, we compare ToMP on the large-scale LaSOT dataset (280 test sequences with 2500 frames on average). The success plot in Fig. 5a shows the overlap precision OP_T as a function of the threshold T. Trackers are ranked w.r.t. their *area-under-the-curve* (AUC) score, shown in the legend. Tab. 5 shows more results including precision and normalized precision for each tracker. Both versions of ToMP with different backbones outperform the recent trackers STARK [43], TransT [6], TrDiMP [40] and DTT [45] in AUC and sets a new state-of-the-art result. Note that even ToMP with ResNet-50 outperforms STARK-ST101 with ResNet-101 (67.6 vs 67.1). Fig. 4 shows the success AUC gain of ToMP compared to recent Transformer based trackers for different attributes annotated in LaSOT [14]. We want to highlight that ToMP outperforms TransT [6] and TrDiMP [40] on each attribute by more than one percent point. Similarly, ToMP achieves higher performance than STARK-ST101 for every attribute. It achieves the highest gain over STARK for *Background Clutter*, showing the disadvantage of using small templates instead of training frames with a large field of view that allow not only to leverage target, but also background information.

LaSOTExtSub [13]: This dataset is an extension of LaSOT. It only contains test sequences assigned to 15 new classes with 10 videos each. The sequences contain 2500 frames on average showing challenging tracking scenarios of small, fast moving objects with distractors present.

	ToMP 101	**ToMP 50**	STARK ST101 [43]	Keep Track [30]	STARK ST50 [43]	Alpha Refine [44]	TransT [6]	Siam R-CNN [38]	Tr DiMP [40]	Super DiMP [8]	SAOT [48]	STM Track [16]	DTT [45]	Pr DiMP [12]
Precision	**73.5**	**72.2**	**72.2**	70.2	71.2	68.0	69.0	68.4	66.3	65.3	-	63.3	-	60.8
Norm. Prec	**79.2**	**78.0**	76.9	77.2	76.3	73.2	73.8	72.2	73.0	72.2	70.8	69.3	-	68.8
Success (AUC)	**68.5**	**67.6**	67.1	67.1	66.4	65.3	64.9	64.8	63.9	63.1	61.6	60.6	60.1	59.8

Table 5. Comparison on the LaSOT [14] test set ordered by AUC.

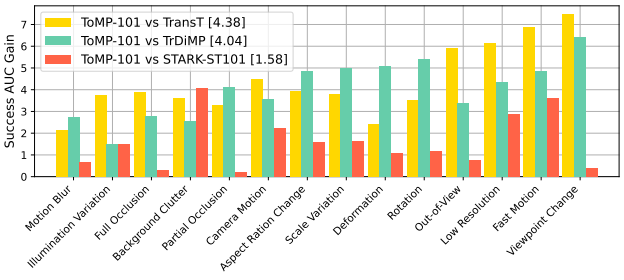
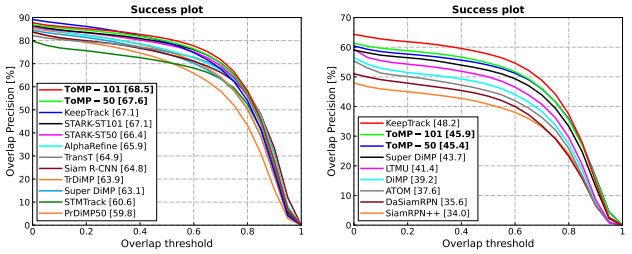


Figure 4. Per attribute analysis on LaSOT [14] between ToMP and recent Transformer based trackers. The bar heights correspond to the gain of our tracker and the legend shows the average gain.



(a) LaSOT [14] (b) LaSOTExtSub [13]

Figure 5. Success plots, showing OP_T, on LaSOT [14] and LaSOTExtSub [13] and AUC is reported in the legend.

	ToMP 101	ToMP 50	STARK ST101 [43]	TransT [6]	STARK ST50 [43]	Siam R-CNN [38]	Alpha Refine [44]	STM Track [16]	Tr DTT [45]	Tr DiMP [40]	Keep Track [30]	Super DiMP [8]	Pr DiMP [12]	Siam FC++ [42]
Precision	78.9	78.6	-	**80.3**	-	80.0	78.3	76.7	78.9	73.1	73.8	73.3	70.4	70.5
Norm. Prec	86.4	86.2	**86.9**	**86.7**	86.1	85.4	85.6	85.1	85.0	83.3	83.5	83.5	81.6	80.0
Success (AUC)	**81.5**	81.2	**82.0**	81.4	81.3	81.2	80.5	80.3	79.6	78.4	78.1	78.1	75.8	75.4

Table 6. Comparison on the TrackingNet [32] test set.

	ToMP 101	ToMP 50	Keep Track [30]	STARK CRACT [15]	STARK ST101 [43]	TrDiMP [40]	TransT [6]	STARK ST50 [43]	Super DiMP [8]	Pr DiMP [12]	STM Track [16]	Siam AttN [46]	Siam R-CNN [38]	KYS [2]	DiMP [1]
UAV123	66.9	69.0	**69.7**	66.4	68.2	67.5	**69.1**	**69.1**	67.7	68.0	64.7	65.0	64.9	–	65.3
OTB-100	70.1	70.1	70.9	**72.6**	68.1	71.1	69.4	68.5	70.1	69.6	**71.9**	71.2	70.1	69.5	68.4
NFS	**66.7**	**66.9**	66.4	62.5	66.2	66.2	65.7	65.2	64.8	63.5	–	–	63.9	63.5	62.0

Table 7. Comparison with the state of the art on the OTB-100 [41], NFS [17] and UAV123 [31] datasets in terms of AUC score.

Fig. 5b shows the success plot where the results of most trackers are obtained from [13], *e.g.*, DaSiamRPN [49], SiamRPN++ [26], ATOM [9], DiMP [1] and LTMU [7]. ToMP exceeds the performance of all trackers except KeepTrack [30] that employs explicit distractor matching between frames. In particular, we outperform SuperDiMP [8] that uses a model optimizer ($+2.2\%$).

TrackingNet [32]: We evaluate ToMP on the large-scale TrackingNet dataset that contains 511 test sequences without publicly available ground-truth. An online evaluation server is used to obtain the tracking metrics shown in Tab. 6 by submitting the raw tracking results. Both versions of ToMP achieve competitive results close to the current state of the art. In particular, ToMP-101 achieves the second best performance in terms of AUC behind STARK [43], outperforming other Transformer based trackers such as TransT [6] and TrDiMP [40].

UAV123 [31]: The UAV dataset consists of 123 test videos that contain small objects, target occlusion, and distractors. Tab. 7 shows the achieved results in terms of success AUC. Again, ToMP achieves competitive results compared to the current state of the art achieved by KeepTrack [30].

OTB-100 [41]: We also report results on the OTB-100 dataset that contains 100 short sequences. Multiple trackers achieve results above 70% AUC. Among them are both versions of ToMP, see Tab. 7. ToMP achieve the same performance as SuperDiMP [8] but slightly higher results than TransT [6] and slightly lower than TrDiMP [40].

NFS [17]: We compete on the NFS dataset (30FPS version) containing 100 test videos. It contains fast motions and challenging sequences with distractors. Both versions of ToMP exceed the performance of the current best method KeepTrack [30] by $+0.5\%$ and $+0.3\%$, see Tab. 7.

	ToMP 101	ToMP 50	STARK ST50 [43]	Super DiMP [8,25]	STARK ST101 [43]	DPMT [25]	TRAT [25]	UPDT [3,25]	DiMP [1,25]	ATOM [9,25]
Accuracy	0.453	0.453	0.478.	0.477	**0.481**	**0.492**	0.464	0.465	0.457	0.462
Robustness	**0.814**	0.789	**0.799**	0.728	0.775	0.745	0.744	0.755	0.734	0.734
EAO	**0.309**	0.297	**0.308**	0.305	0.303	0.303	0.280	0.278	0.274	0.271

Table 8. Comparison to the state of the art of bounding box only methods on VOT2020ST [25] in terms of EAO score.

VOT2020 [25]: Finally, we evaluate on the 2020 edition of the Visual Object Tracking short-term challenge. We compare with the top methods in the challenge [25], as well as more recent methods. The dataset contains 60 videos annotated with segmentation masks. Since ToMP produces bounding boxes we only compare with trackers that produce the bounding boxes as well. The trackers are evaluated following the multi-start protocol and are ranked according to the EAO metric that is based on tracking accuracy and robustness, defined using IoU overlap and failure rate respectively. The results in Tab. 8 show that ToMP-101 achieves the best overall performance, with the highest robustness and competitive accuracy compared to previous methods.

4.3. Limitations

Transformer Encoders consist of self-attention layers that compute similarity matrices between multiple training and test frame features and thus lead to a large memory footprint that impacts training and inference run-time. Thus, in future work this limitation should be addressed by evaluating alternatives such as [22, 24, 34] aiming at decreasing the memory burden. Another limiting factor of ToMP arises from challenging tracking sequences. In particular, distractors present while the target is occluded is a typical failure scenario of ToMP, since it is lacking explicit distractor handling as in KeepTrack [30].

5. Conclusion

We propose a novel tracking architecture employing a Transformer-based model predictor. The model predictor estimates the weights of the compact DCF target model to localize the target in the test frame. In addition, the predictor produces a second set of weights used for precise bounding box regression. To achieve this, we develop two new modules that encode target location and its bounding box in the training features. We conduct comprehensive experimental validation and analysis of ToMP on several challenging datasets, and set a new state of the art on three.

Acknowledgments: This work was partly supported by the ETH Zürich Fund (OK), Siemens Smart Infrastructure, the ETH Future Computing Laboratory (EFCL) financed by a gift from Huawei Technologies, an Amazon AWS grant, and an Nvidia hardware grant.

References

[1] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 5, 7, 8

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020. 8

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 8

[4] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 1, 2

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, August 2020. 2, 4

[6] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 1, 2, 5, 7, 8

[7] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 8

[8] Martin Danelljan and Goutam Bhat. PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking, 2019. Accessed: 1/05/2021. 1, 6, 7, 8

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 5, 8

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2017. 1

[11] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2016. 2

[12] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 5, 7, 8

[13] Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Mingzhen Huang, Juehuan Liu, Yong Xu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision (IJCV)*, 129(2):439–461, 2021. 7, 8

[14] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 6, 7, 8

[15] Heng Fan and Haibin Ling. Cract: Cascaded regression-align-classification for robust visual tracking. *arXiv preprint arXiv:2011.12483*, 2020. 8

[16] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. 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. 5, 7, 8

[17] 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. 1, 8

[18] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2017. 1

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 6

[20] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(3):583–596, 2015. 1, 2

[21] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(5):1562–1577, 2021. 6

[22] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5156–5165, July 2020. 8

[23] Dai Kenan, Wang Dong, Lu Huchuan, Sun Chong, and Li Jianhua. Visual tracking via adaptive spatially-regularized correlation filters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[24] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 8

[25] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernández, and et al. The eighth visual object tracking vot2020 challenge results. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, August 2020. 8

[26] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. 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. 8

[27] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 6

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 6

[29] Alan Lukezic, Tomás Vojír, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision (IJCV)*, 126(7):671–688, 2018. 1, 2

[30] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13444–13454, October 2021. 7, 8

[31] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, October 2016. 8

[32] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, 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)*, 2018. 6, 8

[33] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 5

[34] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3531–3539, January 2021. 8

[35] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[36] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 5

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 4

[38] Paul Voigtlaender, Jonathon Luiten, Philip H.S. Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by re-detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7, 8

[39] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[40] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. 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. 1, 2, 7, 8

[41] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1834–1848, 2015. 8

[42] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, February 2020. 5, 8

[43] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10448–10457, October 2021. 1, 2, 4, 5, 7, 8

[44] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 7, 8

[45] Bin Yu, Ming Tang, Linyu Zheng, Guibo Zhu, Jinqiao Wang, Hao Feng, Xuetao Feng, and Hanqing Lu. High-performance discriminative tracking with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9856–9865, October 2021. 1, 2, 5, 7, 8

[46] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R. Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 8

[47] Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, and Hanqing Lu. Learning feature embeddings for discriminant model based tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020. 2

[48] Zikun Zhou, Wenjie Pei, Xin Li, Hongpeng Wang, Feng Zheng, and Zhenyu He. Saliency-associated object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9866–9875, October 2021. 7

[49] Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 8