SeqTrack: Sequence to Sequence Learning for Visual Object Tracking

Xin Chen[1], Houwen Peng[2,†], Dong Wang[1,†], Huchuan Lu[1], Han Hu[2]
[1]Dalian University of Technology [2]Microsoft Research

Abstract

In this paper, we present a new sequence-to-sequence learning framework for visual tracking, dubbed SeqTrack. It casts visual tracking as a sequence generation problem, which predicts object bounding boxes in an autoregressive fashion. This is different from prior Siamese trackers and transformer trackers, which rely on designing complicated head networks, such as classification and regression heads. SeqTrack only adopts a simple encoder-decoder transformer architecture. The encoder extracts visual features with a bidirectional transformer, while the decoder generates a sequence of bounding box values autoregressively with a causal transformer. The loss function is a plain cross-entropy. Such a sequence learning paradigm not only simplifies tracking framework, but also achieves competitive performance on benchmarks. For instance, SeqTrack gets 72.5% AUC on LaSOT, establishing a new state-of-the-art performance. Code and models are available at here.

1. Introduction

Visual object tracking is a fundamental task in computer vision. It aims to estimate the position of an arbitrary target in a video sequence, given only its location in the initial frame. Existing tracking approaches commonly adopt a divide-and-conquer strategy, which decomposes the tracking problem into multiple subtasks, such as object scale estimation and center point localization. Each subtask is addressed by a specific head network. For example, SiamRPN [27] and its follow-up works [3, 7, 48, 55, 58] adopt classification heads for object localization and regression heads for scale estimation, as sketched in Fig. 1(a). STARK [53] and transformer-based trackers [4, 10, 17, 44] design corner head networks to predict the bounding box corners of target objects, as visualized in Fig. 1(b).

Such a divide-and-conquer strategy has demonstrated superior performance on tracking benchmarks and thereby become the mainstream design in existing models. However, two deficiencies still exist. First, each subtask requires a customized head network, leading to a complicated tracking framework. Second, each head network requires one or more learning loss functions, *e.g.*, cross-entropy loss [7,27], ℓ_1 loss [7,27,53,55], generalized IoU loss [7,53,55], which make the training difficult due to extra hyperparameters.

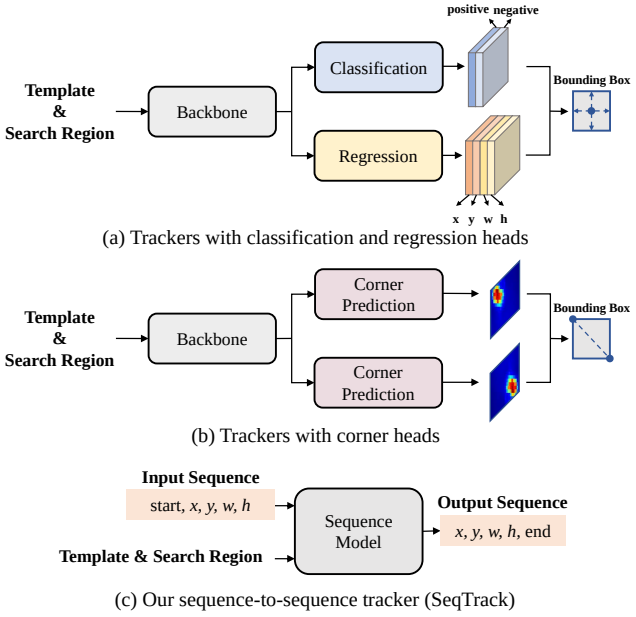

Figure 1. Comparison of tracking frameworks. (a) The framework with object classification head and bounding box regression head. (b) The framework with corner prediction heads. (c) Sequence-to-sequence tracking framework without complicated head networks.

To address these issues, in this paper, we propose a new Sequence-to-sequence Tracking (SeqTrack) framework, as shown in Fig. 1(c). By modeling tracking as a sequence generation task, SeqTrack gets rid of complicated head networks and redundant loss functions. It is based upon the intuition that if the model knows where the target object is, we could simply teach it how to read the bounding box out, rather than explicitly performing additional classification and regression using a divide-and-conquer strategy.

To this end, we convert the four values of a bounding box into a sequence of discrete tokens and make the model to learn generating this sequence token-by-token. We adopt a simple encoder-decoder transformer to model the generation. The encoder is to extract visual features of video

† Corresponding authors.

frames, while the decoder is to generate the sequence of bounding box values using the extracted features. The generation is executed in an autoregressive fashion, which means the model generates a token depending on previously observed ones. At each step, a new generated token value is fed back into the model to produce the next one. We impose a causal mask on the self-attention modules in the decoder to prevent tokens from attending to subsequent tokens. Such a causal masking mechanism ensures that the generation of the token at position i only depends on its proceeding tokens at positions less than i. The visual features are integrated into the decoder through cross-attention layers [46]. The generation ends when it outputs four token values of the bounding box. The output sequence is directly used as the result.

Experiments demonstrate our SeqTrack method is effective, achieving new state-of-the-art performance on several tracking benchmarks. For instance, SeqTrack-B256 obtains 74.7% AO score on GOT-10k [20], outperforming the recent OSTrack-256 tracker [55] by 3.7% under aligned settings, *i.e.*, using the same encoder architecture and input resolution. Moreover, compared to the recent state-of-the-art tracker MixFormer [10], SeqTrack-B256 runs 1.4 times faster (40 *v.s.* 29 *fps*) while getting 0.7% superior AUC score on LaSOT [16]. It is worth noting that all these prior methods heavily rely on well-designed head networks and the corresponding complicated loss functions [30, 41]. In contrast, our SeqTrack only adopts a plain encoder-decoder transformer architecture with a simple cross-entropy loss.

In summary, the contributions of this work are two-fold:

- We propose a sequence-to-sequence learning method for visual tracking. It casts tracking as a generation task, which offers a new perspective on tracking modeling.
- We present a new family of sequence tracking models, which strike a good trade-off between speed and accuracy. Experiments verify the efficacy of the new models.

2. Related Work

Visual Tracking. Existing tracking approaches commonly adopt a divide-and-conquer strategy to decompose tracking into multiple subtasks. They first extract visual features of video frames using a deep neural network [14, 19, 46], and then design multiple task-specific head networks to predict the bounding boxes of target objects. According to the difference of head networks, prior trackers can be divided into two categories: 1) trackers based on classification and regression, and 2) trackers based on corner predictions.

Most prevalent trackers [3, 7, 12, 27, 51, 55] belong to the first category, which models tracking with a classification head for foreground-background prediction and a regression head for object scale estimation. For the classification head network, most trackers adopt stacked convolutional layers with various loss functions, including cross-entropy loss [7, 27], focal loss [51, 55], modified ℓ_2 loss [3, 12], and KL-divergence loss [13]. For the regression head, Siamese trackers and some transformer-based trackers adopt stacked convolutional layers with ℓ_1 loss [26, 27] and IoU loss [7, 50], while discriminative trackers [3, 12] employ the IoU-Net [21] with MSE loss [12].

STARK [53] and its follow-up works [4, 10, 44] belong to the corner prediction category, which locates target objects using corner prediction heads. They employ a two-branch network to generate two probability maps for the top-left and the bottom-right corners of the target object. The final object bounding boxes are obtained by calculating the expectation of corners' probability distribution. The loss function is a combined ℓ_1 and generalized IoU [41] loss.

Such well-designed head networks and loss functions complicate existing tracking frameworks and also make the training difficult. In contrast, our method casts tracking as a sequence generation problem, getting rid of complicated heads and unnecessary loss functions. It only uses a single cross-entropy loss with a plain transformer architecture.

Sequence Learning. Sequence-to-sequence learning is originally proposed for natural language modeling [9, 45], and recently applied to computer vision. Pix2Seq [5] is a representative work that casts object detection as a token generation task conditioned on the observed pixel inputs. Besides object detection, such a sequence learning method has also been successfully extended to other vision tasks, such as instance segmentation and keypoint detection [6]. Moreover, in cross modality domain, sequence learning is becoming increasingly popular. For example, text-to-image generation models like DALL-E [40] and vision-language models like Flamingo [1] all adopt sequence-to-sequence learning to unify multi-modality pretraining.

Our sequence learning framework shares a similar spirit with Pix2Seq [5]. Both of them cast vision tasks as a sequence generation problem, and discretize the continuous values of bounding box coordinates into integers. However, our method differs from Pix2Seq in three fundamental ways. 1) The constructions of the sequences are different. Pix2Seq uses object corner coordinates and object categories to set up the sequence, while our method uses center point coordinates and object scales. 2) The architectures are different. Pix2Seq adopts ResNet [19] as its backbone network followed by a encoder-decoder transformer. Our method is more compact, only using a single encoder-decoder transformer. It employs ViT [14] as the encoder for feature extraction and causal transformer blocks as the decoder for sequence generation. 3) The tasks are different, Pix2Seq is designed for detection, while ours is for tracking. Some previous tracking designs, such as online template update, can be seamlessly integrated into our method.

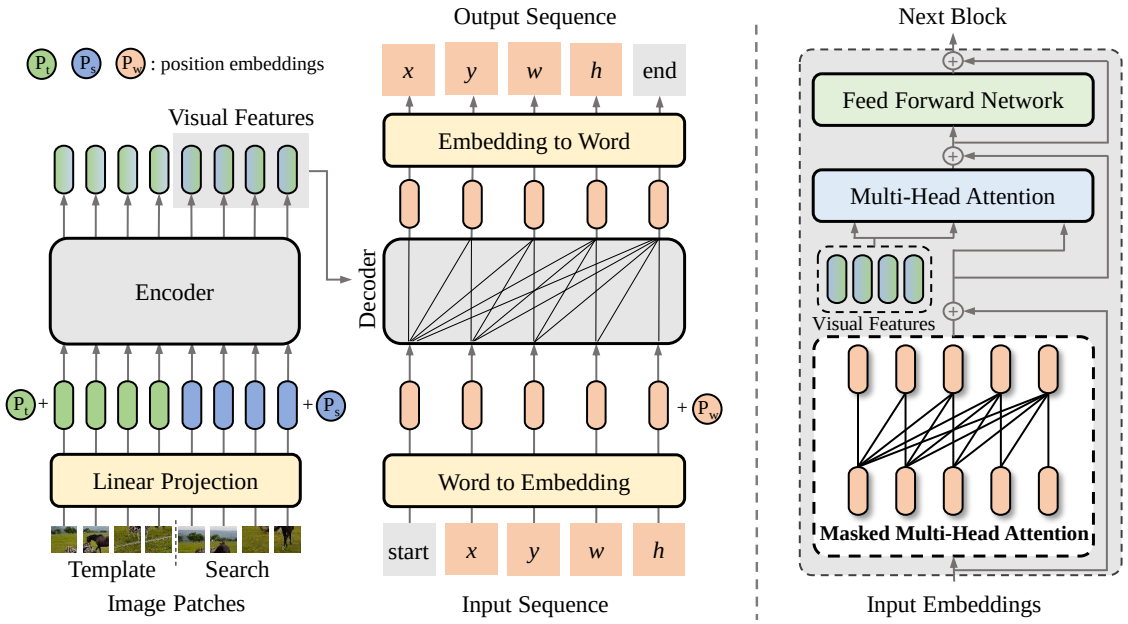

Figure 2. (a) Architecture of the proposed SeqTrack. The key component is an encoder-decoder transformer. The encoder extracts visual features of input video frames. The causal decoder autoregressively generates the sequence of the bounding box tokens using the extracted features. (b) The detailed transformer block in the causal decoder. The input embeddings interact in a casual manner through a masked multi-head attention. The visual feature is incorporated into the decoder through a multi-head attention layer.

3. Method

This section presents the proposed SeqTrack method in detail. First, we briefly overview our sequence-to-sequence tracking framework. Then, we depict image and sequence representations, and the proposed model architecture. Finally, we introduce the training and inference pipelines and the integration of tracking prior knowledge.

3.1. Overview

The overall framework of SeqTrack is presented in Fig. 2(a). It adopts a simple encoder-decoder transformer architecture. The object bounding box is first converted into a sequence of discrete tokens, *i.e.*, $[x,y,w,h]$. The encoder extracts visual features of input video frames, while the decoder autoregressively generates the sequence of bounding box tokens using the extracted features. A causal attention mask is imposed on the self-attention modules in the decoder to restrict the tokens to only attend to their proceeding tokens. In addition to the four bounding box tokens, we also use two special tokens: `start` and `end`. The `start` token tells the model to begin the generation, while the `end` token represents the completion of the generation. During training, the input sequence of the decoder is $[\texttt{start},x,y,w,h]$, and the target sequence is $[x,y,w,h,\texttt{end}]$. During inference, the input sequence of the decoder initially contains a single `start` token. At each step, a new bounding box token is generated and appended to the input sequence to produce the next one. When the four token values of the bounding box are generated, the prediction ends.

3.2. Image and Sequence Representation

Image Representation. The input to the encoder comprises a template image $\boldsymbol{t} \in \mathbb{R}^{3\times H\times W}$ and a search image $\boldsymbol{s} \in \mathbb{R}^{3\times H\times W}$. The image $\boldsymbol{t}$ represents the object of interest, while $\boldsymbol{s}$ represents the search region in subsequent video frames. In existing trackers [2, 7, 27, 53], the resolution of template images is typically smaller than that of search images. In contrast, we use the same size for the two images, because we find that adding more background in the template is helpful for improving tracking performance. The search and template images are partitioned into patches: $\boldsymbol{s}_p \in \mathbb{R}^{N\times P^2\times 3}$ and $\boldsymbol{t}_p \in \mathbb{R}^{N\times P^2\times 3}$, where (P,P) is the patch size, $N{=}HW/P^2$ is the patch number. A linear projection is used to map the image patches to visual embeddings. Learnable position embeddings [46] are added to the patch embeddings to retain positional information. The combined embeddings are then fed into the encoder.

Sequence Representation. We convert the target bounding box into a sequence of discrete tokens. Specifically, a bounding box is determined by its center point $[x,y]$ and scale $[w,h]$. There are several bounding box formats, such as $[x,y,w,h]$ and $[w,h,x,y]$. We use the format of $[x,y,w,h]$, because it aligns with human prior knowledge: localizing object position $[x,y]$ first, and then estimating its scale $[w,h]$. Each of the continuous coordinates is uniformly discretized into an integer between $[1,\, n_{bins}]$. We use a shared vocabulary $\boldsymbol{V}$ for all coordinates. Each integer between $[1,\, n_{bins}]$ can be regarded as a word in $\boldsymbol{V}$, so the size of $\boldsymbol{V}$ is n_{bins} $(4,000$ in our experiments). The final input

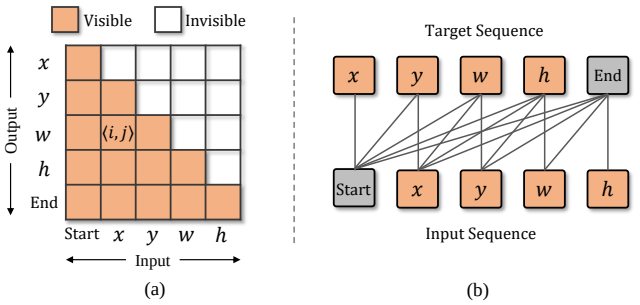

Figure 3. (a) Illustration of the causal attention mask in the decoder using a maximum sequence length of 5 tokens. An orange cell at row i and column j indicates that the attention mechanism is allowed to attend to the jth input token, when producing the ith output token. (b) Illustration of the input and target sequences. Similar to autoregressive language modeling [46], the input sequence is the target sequence with one position offset.

sequence is $[\texttt{start},x,y,w,h]$, and the target sequence is $[x,y,w,h,\texttt{end}]$. Each word in $\boldsymbol{V}$ corresponds to a learnable embedding, which is optimized during training. The special token `start` also corresponds to a learnable embedding. The corresponding embeddings of the input words are fed into the decoder. Since the transformer is permutation-invariant, we augment word embeddings with learnable position embeddings [46]. For the final model outputs, we need to map the embeddings back to words. To this end, we employ a multi-layer perceptron with a softmax to sample words from $\boldsymbol{V}$ according to the output embeddings.

3.3. Model Architecture

Our model adopts a simple encoder-decoder transformer architecture, as plotted in Fig. 2. The encoder is applied to extract visual features of input video frames, while the decoder is to predict the bounding boxes of target objects in an autoregressive fashion.

Encoder. The encoder is a standard vision transformer (ViT) [14]. The architecture is the same as ViT [14] except for two minor modifications: 1) The `CLASS` token is removed, since it is designed for image classification task. 2) A linear projection is appended to the last layer to align the feature dimensions of the encoder and decoder. The encoder receives patch embeddings of template and search images, and outputs their corresponding visual features. Only the features of the search image are fed into the decoder. The function of the encoder is to extract the visual features of search and template images in a joint way, and learn feature-level correspondence through attention layers.

Decoder. The decoder of SeqTrack is a causal transformer [46]. As shown in Fig. 2(b), each transformer block consists of a masked multi-head attention, a multi-head attention, and a feed forward network (FFN). More concretely, the masked multi-head attention receives the word embeddings from the preceding block and utilizes a causal mask to ensure that the output of each sequence element only depends on its previous sequence elements. In other words, the attention mask restricts the output embedding at position i to only attend to the input embeddings at positions less than i, as shown in Fig. 3(a). After that, the multi-head attention integrates the extracted visual features into the word embeddings, which allows the word embeddings to attend to the visual features derived from the encoder. Finally, a feed forward network (FFN) is applied to generate embeddings for the next block.

3.4. Training and Inference

Training. Similar to language modeling [46], SeqTrack is trained to maximize the log-likelihood of the target tokens conditioned on the preceding subsequence and input video frames using a cross-entropy loss. The learning objective function is formulated as:

$$\text{maximize} \sum_{j=1}^{L} \log Q(\hat{\boldsymbol{z}}_j | \boldsymbol{s}, \boldsymbol{t}, \hat{\boldsymbol{z}}_{<j}), \tag{1}$$

where $Q(\cdot)$ is the softmax probability, $\boldsymbol{s}$ is the search image, $\boldsymbol{t}$ is the template, $\hat{\boldsymbol{z}}$ is the target sequence, j is the position of the token, and L is the length of the target sequence. Here, $\hat{\boldsymbol{z}}_{<j}$ represents the preceding subsequence used to predict the current token $\hat{\boldsymbol{z}}_j$. The input sequence is the target sequence with one position offset (omit `start` and `end`), as visualized in Fig. 3(b). Such an offset, combined with the causal masking, guarantees the autoregressive property of the sequence model. The target sequence can be regarded as a description of the object bounding box. The training is to teach the model to "read out" the words of the description based on the preceding words.

Inference. During inference, the encoder perceives the template image and the search region in subsequent video frames. The initial input to the decoder is the `start` token, which tells the model to start the generation. Then, the model "reads out" the target sequence $[x,y,w,h,\texttt{end}]$ token by token. For each token, the model samples it from the vocabulary $\boldsymbol{V}$ according to the maximum likelihood, *i.e.*, $\hat{\boldsymbol{z}}_j = \arg\max_{\boldsymbol{z}_j} Q(\boldsymbol{z}_j | \boldsymbol{s}, \boldsymbol{t}, \hat{\boldsymbol{z}}_{<j})$, where $\boldsymbol{z}_j$ is the words in $\boldsymbol{V}$. In addition, we also introduce online template update and window penalty to integrate prior knowledge during inference, further improving the model accuracy and robustness. The details are described in the following subsection.

3.5. Prior Knowledge Integration

Prior knowledge, such as window penalty [2,27] and online update [12,28,53], has been widely incorporated into existing tracking models and proved effective [7,10,43,55]. In this subsection, we discuss how to introduce such prior knowledge to the proposed sequence-to-sequence learning framework to further improve tracking performance.

Online Update. Since that the appearance of a target object may change dramatically during online tracking, the

Model	Encoder	Input Resolution	Params (M)	FLOPs (G)	Speed (*fps*)
SeqTrack-B256	ViT-B	256×256	89	66	40
SeqTrack-B384	ViT-B	384×384	89	148	15
SeqTrack-L256	ViT-L	256×256	309	232	15
SeqTrack-L384	ViT-L	384×384	309	524	5

Table 1. Details of SeqTrack model variants.

initial template image is not always reliable for target localization. To address this issue, we integrate online template update [53] into our method. More specifically, in addition to the initial template image, we introduce a dynamic template to capture the appearance changes of target objects. The dynamic template is updated on the fly. It is well recognized that poor-quality templates may lead to inferior tracking performance [10]. As a consequence, we use the likelihood of the generated tokens to select reliable dynamic templates automatically. Specifically, we average the softmax scores over the four generated bounding box values. If the averaged score is larger than a specific threshold τ and the update interval T_u is reached, the dynamic template will be updated with the tracking result in the current frame, otherwise it maintains the previous state. Experiments demonstrate this simple approach can improve tracking accuracy (see the ablation in Sec. 4.3). Moreover, unlike previous methods [10, 53], our method does not bring any additional score head to determine whether to update the template, which usually requires a second stage training.

Window Penalty. It is empirically validated that the pixel displacement between two consecutive frames is relatively small [2, 27]. To penalize large displacements, we introduce a new window penalty strategy to our method during online inference. Concretely, the target object's position in the previous frame corresponds to the center point of the current search region. The discrete coordinates of the center point in the current search region are $[\frac{n_{bins}}{2}, \frac{n_{bins}}{2}]$. We penalize the likelihood of integers (*i.e.*, words) in the vocabulary $\boldsymbol{V}$ based on their difference to $\frac{n_{bins}}{2}$, when generating x and y. A large difference between an integer and $\frac{n_{bins}}{2}$ will get a large penalty accordingly. In implementation, the softmax scores of integers form a vector of size n_{bins}. We simply multiply this vector by the Hanning window with the same size. In this way, the large displacements are suppressed. Unlike the previous practice [7, 27], we do not introduce additional hyperparameters to tune the magnitude of the penalty.

4. Experiments

4.1. Implementation Details

Model. We develop four variants of SeqTrack models with different encoder architectures and input resolutions, as elaborated in Tab. 1. We adopt ViT-B [14] as our encoder architecture for SeqTrack-B256 and B384, and ViT-L [14] for SeqTrack-L256 and L384. The encoders are initialized with the MAE [18] pre-trained parameters. The patch size is set to 16×16. The decoder consists of 2 transformer blocks, which is the same for all models. The decoder hidden size is 256, the number of attention heads is 8, and the hidden size of the feed forward network (FFN) is 1024. The number of quantization bins n_{bins} and the vocabulary size are all set to $4,000$. The dimension of word embedding is 256, which is consistent with the decoder hidden size. The embedding-to-word sampling uses a 3-layer perceptron followed by a softmax. The hidden dimension of the perceptron is 256. The output dimension is n_{bins}, which is aligned with the number of words in $\boldsymbol{V}$. The word with the maximum likelihood is sampled as the output word. In addition, we present model parameters, FLOPs, and inference speed in Tab. 1. The speed is measured on Intel Core i9-9900K CPU @ 3.60GHz with 64 GB RAM and a single 2080 Ti GPU. All the models are implemented with Python 3.8 and PyTorch 1.11.0.

Training. Our training data includes the training splits of COCO [31], LaSOT [16], GOT-10k [20], and TrackingNet [37]. Aligned with VOT2020 evaluation protocol [25], we remove the $1k$ forbidden videos in GOT-10k during training. For the evaluation on GOT-10k test set, we follow the official requirements [20] and only use the training split of GOT-10k. The template and search images are obtained by expanding the target bounding boxes by a factor of 4. Horizontal flip and brightness jittering are used for data augmentation. We train the model with AdamW [32] optimizer and set the learning rate of the encoder to $1e{-}5$, the decoder and remaining modules to $1e{-}4$, and the weight decay to $1e{-}4$. The training of SeqTrack are conducted on Intel Xeon CPU E5-2690 v4 @ 2.60GHz with 512 GB RAM and 8 Tesla A100 GPUs with 80GB memory. Each GPU holds 8 image pairs, resulting in a total batch size of 64. The model is trained for a total of 500 epochs with $60k$ image pairs per epoch. The learning rate decreases by a factor of 10 after 400 epochs.

Inference. The online template update interval T_u is set to 25 by default, while the update threshold τ is set to 0.015. For window penalty, the softmax likelihood of the $4,000$ words in the vocabulary $\boldsymbol{V}$ are directly multiplied by a 1D Hanning window of size $4,000$.

4.2. State-of-the-Art Comparisons

We compare our SeqTrack with state-of-the-art trackers on eight tracking benchmarks.

LaSOT. LaSOT [16] is a large-scale long-term tracking benchmark. The test set contains 280 videos with an average length of 2448 frames. As reported in Tab. 2, SeqTrack-B256 performs slightly better than the recent state-of-the-art method OSTrack-256 [55], getting 0.8% AUC score improvement, using the same ViT-B encoder architecture

Table 2. State-of-the-art comparisons on four large-scale benchmarks. We add a symbol * over GOT-10k to indicate that the corresponding models are only trained with the GOT-10k training set. Otherwise, the models are trained with all the training data presented in Sec. 4.1. The top three results are highlight with **red**, blue and green fonts, respectively.

	Method	LaSOT [16]			LaSOT$_{ext}$ [15]			TrackingNet [37]			GOT-10k* [20]		
		AUC	P$_{Norm}$	P	AUC	P$_{Norm}$	P	AUC	P$_{Norm}$	P	AO	SR$_{0.5}$	SR$_{0.75}$
Seq2Seq	SeqTrack-L384	**72.5**	81.5	**79.3**	**50.7**	**61.6**	57.5	**85.5**	**89.8**	**85.8**	**74.8**	81.9	**72.2**
	SeqTrack-L256	72.1	**81.7**	79.0	50.5	61.5	57.2	85.0	89.5	84.9	74.5	83.2	72.0
	SeqTrack-B384	71.5	81.1	77.8	50.5	**61.6**	57.5	83.9	88.8	83.6	74.5	84.3	71.4
	SeqTrack-B256	69.9	79.7	76.3	49.5	60.8	56.3	83.3	88.3	82.2	74.7	**84.7**	71.8
Corner Prediction	SimTrack [4]	70.5	79.7	-	-	-	-	83.4	87.4	-	69.8	78.8	66.0
	Mixformer-L [10]	70.1	79.9	76.3	-	-	-	83.9	88.9	83.1	-	-	-
	Mixformer-22k [10]	69.2	78.7	74.7	-	-	-	83.1	88.1	81.6	70.7	80.0	67.8
	AiATrack [17]	69.0	79.4	73.8	47.7	55.6	55.4	82.7	87.8	80.4	69.6	63.2	80.0
	UTT [33]	64.6	-	67.2	-	-	-	79.7	-	77.0	67.2	76.3	60.5
	CSWinTT [44]	66.2	75.2	70.9	-	-	-	81.9	86.7	79.5	69.4	78.9	65.4
	STARK [53]	67.1	77.0	-	-	-	-	82.0	86.9	-	68.8	78.1	64.1
Classification + Regression	OSTrack-384 [55]	71.1	81.1	77.6	50.5	61.3	**57.6**	83.9	88.5	83.2	73.7	83.2	70.8
	OSTrack-256 [55]	69.1	78.7	75.2	47.4	57.3	53.3	83.1	87.8	82.0	71.0	80.4	68.2
	SwinTrack [29]	71.3	-	76.5	49.1	-	55.6	84.0	-	82.8	72.4	-	67.8
	RTS [39]	69.7	76.2	73.7	-	-	-	81.6	86.0	79.4	-	-	-
	Unicorn [52]	68.5	-	-	-	-	-	83.0	86.4	82.2	-	-	-
	SLT [24]	66.8	75.5	-	-	-	-	82.8	87.5	81.4	67.5	76.5	60.3
	SBT [50]	66.7	-	71.1	-	-	-	-	-	-	70.4	80.8	64.7
	ToMP [34]	68.5	79.2	73.5	45.9	-	-	81.5	86.4	78.9	-	-	-
	KeepTrack [35]	67.1	77.2	70.2	48.2	-	-	-	-	-	-	-	-
	AutoMatch [57]	58.3	-	59.9	-	-	-	76.0	-	72.6	65.2	76.6	54.3
	TransT [7]	64.9	73.8	69.0	-	-	-	81.4	86.7	80.3	67.1	76.8	60.9
	TrDiMP [48]	63.9	-	61.4	-	-	-	78.4	83.3	73.1	68.8	80.5	59.7
	SiamAttn [56]	56.0	64.8	-	-	-	-	75.2	81.7	-	-	-	-
	SiamBAN [8]	51.4	59.8	-	-	-	-	-	-	-	-	-	-
	DSTrpn [42]	43.4	54.4	-	-	-	-	64.9	-	58.9	-	-	-
	Ocean [58]	56.0	65.1	56.6	-	-	-	-	-	-	61.1	72.1	47.3
	SiamR-CNN [47]	64.8	72.2	-	-	-	-	81.2	85.4	80.0	64.9	72.8	59.7
	DiMP [3]	56.9	65.0	56.7	39.2	47.6	45.1	74.0	80.1	68.7	61.1	71.7	49.2
	SiamPRN++ [26]	49.6	56.9	49.1	34.0	41.6	39.6	73.3	80.0	69.4	51.7	61.6	32.5
	ATOM [12]	51.5	57.6	50.5	37.6	45.9	43.0	70.3	77.1	64.8	55.6	63.4	40.2
	MDNet [38]	39.7	46.0	37.3	27.9	34.9	31.8	60.6	70.5	56.5	29.9	30.3	9.9

and input resolution. SeqTrack-B384 surpasses all previous trackers with an AUC score of 71.5%. Furthermore, SeqTrack-L384 and L256 obtain new state-of-the-art AUC scores of 72.5% and 72.1%, respectively. SeqTrack-L384 outperforms the prior best tracker SwinTrack [29] by 1.2%. Fig. 4 shows the results of attribute-based evaluation, illustrating that our SeqTrack-L384 performs better than other competing trackers on almost all attributes. In particular, our method has great advantages in the attributes of deformation and background clutter, demonstrating a superior discriminative ability of the model.

LaSOT$_{ext}$. LaSOT$_{ext}$ [15] is a recently released dataset with 150 video sequences and 15 object classes. The results on LaSOT$_{ext}$ are also presented in Tab. 2. Under aligned settings, SeqTrack-B256 obtains 2.1% higher AUC score than OSTrack-256. With a more powerful ViT-L encoder, SeqTrack-L384 achieves the best AUC score of 50.7%.

TrackingNet. TrackingNet [37] is a large-scale dataset containing 511 videos, which covers diverse object categories and scenes. As reported in Tab. 2, SeqTrack-B384 and SeqTrack-B256 achieve competitive results compared with the previous state-of-the-art trackers. SeqTrack-L384 gets the best AUC of 85.5%, surpassing the previous best tracker SwinTrack by 1.5%.

GOT-10k. GOT-10k [20] test set contains 180 videos covering a wide range of common challenges in tracking. Following the official requirements, we only use the GOT-10k training set to train our models. As reported in Tab. 2, SeqTrack-B256 achieves 3.7% gains over OSTrack-256 under aligned settings. SeqTrack-L384 obtains the best AO score of 74.8%, outperforming the previous state-of-the-art method by 1.1%.

TNL2K, NFS and UAV123. We evaluate our trackers on three additional benchmarks, including TNL2K [49], NFS [23], and UAV123 [36]. TNL2K is a recently released large-scale dataset with 700 challenging video sequences. NFS and UAV123 are two small-scale benchmarks including 100 and 123 videos, respectively. On the large-scale TNL2K benchmark, our SeqTrack-L384 obtains a

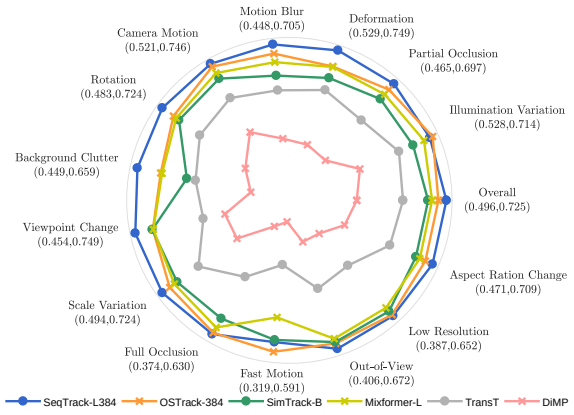

Figure 4. AUC scores of different attributes on LaSOT [16]

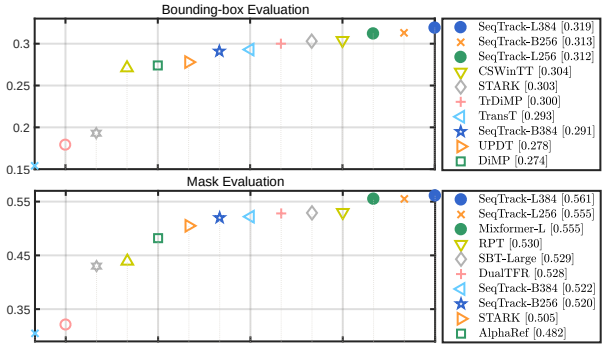

Figure 5. EAO rank plots on VOT2020. For the mask evaluation, we use Alpha-Refine [54] to predict masks.

new state-of-the-art performance with 57.8% AUC score, as reported in Tab. 3. On the small-scale benchmarks NFS and UAV123, Tab. 3 shows our SeqTrack models also achieve competitive results, being comparable or slightly better than the most recent trackers OSTrack and SimTrack.

VOT2020. VOT2020 [25] benchmark contains 60 challenging videos. VOT uses binary segmentation masks as the groundtruth. To predict the segmentation masks, we equip SeqTrack with Alpha-Refine [54]. We evaluate our models by submitting both the bounding boxes and the segmentation masks. As shown in Fig. 5, our SeqTrack-L384 achieves the best results with EAO of 31.9% and 56.1% on bounding box and mask evaluations, respectively.

4.3. Ablation and Analysis.

We use SeqTrack-B256 without the online template update module as the baseline model in our ablation study. The result of the baseline is reported in Tab. 4 (#1).

Autoregressive v.s. Bidirectional. Our method generates a sequence in an *autoregressive* manner, which predicts the bounding box values one by one. We compare this autoregressive method with another *bidirectional* method that predicts all coordinate values simultaneously. In the bidirectional method, the input sequence consists of four special tokens similar to the `start` token. The decoder receives the sequence and predicts the four coordinates in a batch.

Method	TNL2K [49]	NFS [23]	UAV123 [36]
SeqTrack-L384	**57.8**	66.2	68.5
SeqTrack-L256	56.9	66.9	69.7
SeqTrack-B384	56.4	66.7	68.6
SeqTrack-B256	54.9	**67.6**	69.2
OSTrack [55]	55.9	66.5	70.7
SimTrack [4]	55.6	-	**71.2**
STARK [53]	-	66.2	68.2
TransT [7]	50.7	65.7	69.1
TrDiMP [48]	-	66.5	67.5
DiMP [3]	44.7	61.8	64.3
Ocean [58]	38.4	49.4	57.4
ATOM [12]	40.1	58.3	63.2
ECO [11]	32.6	52.2	53.5
RT-MDNet [22]	-	43.3	52.8
SiamFC [2]	29.5	37.7	46.8

Table 3. Comparison with state-of-the-art methods on additional benchmarks in AUC score.

The causal attention mask is removed, allowing tokens to attend to each other. As shown in Tab. 4 (#2), the bidirectional method performs much inferior to the autoregressive one, demonstrating that the causal relation between tokens is important for sequence modeling in tracking.

Inputs of the Encoder. We compare different input methods for the encoder. As described in Sec. 3.3, the template and the search region are fed into the encoder together. Then the encoder extracts their visual features in a *joint* way. We compare it with a *separate* approach: like Siamese methods [2, 27], two weight-sharing encoders extract the features of the template and the search images separately, and then feed their features into the decoder. Tab. 4 (#3) shows that the separate feature extraction method is inferior to the joint one by 7.2% AUC on LaSOT. The underlying reason might be that the joint method allows the encoder to learn better feature correspondence between the template and search images.

Inputs of the Decoder. We first compare different input sequences to the decoder. Tab. 4 (#4 and #5) present two other formats of sequence: 1) $[w,h,x,y]$, where the model first generates object's scale and then generates its position coordinates; and 2) $[x_{min},y_{min},x_{max},y_{max}]$, where $[x_{min},y_{min}]$ denotes the top-left corner while $[x_{max},y_{max}]$ denotes the bottom-right one. We observe that the default format $[x,y,w,h]$ gets the best result, because it aligns with human prior knowledge: localizing object position first, and then estimating its scale. We also explore the influence of the number of quantization bins n_{bins}, as shown in Fig. 6. Increasing the number of bins n_{bins} can improve the performance because the quantization error is reduced accordingly. The performance becomes saturated when n_{bins} is larger than $4,000$, thus we set n_{bins} to $4,000$.

Moreover, we compare different input visual features to the decoder. By default, we only feed the features of the search image into the decoder, as illustrated in Fig. 2. Here,

#	Method	LaSOT	GOT-10k	Δ
1	Baseline	69.2	73.7	–
2	Autoregressive → Bidirectional	64.8	72.4	**-2.9**
3	Joint → Separate	62.0	66.1	**-7.4**
4	$[x,y,w,h] \rightarrow [w,h,x,y]$	67.1	71.8	**-2.0**
5	$[x,y,w,h] \rightarrow [x_{min},y_{min},x_{max},y_{max}]$	68.3	70.7	**-2.0**
6	Concat of Search-Template	69.6	73.3	**-0.0**
7	Avg. of Search-Template	69.2	72.2	**-0.8**
8	+ Likelihood-based Online Update	69.9	76.1	**+1.6**
9	+ Naive Online Update	69.3	73.1	**-0.3**
10	− Window Penalty	68.8	73.1	**-0.5**

Table 4. Ablation Study on LaSOT [16] and GOT-10k [20]. We use gray, green, purple, yellow, and pink colors to denote baseline setting, framework, input to encoder, input to decoder, and tracking prior integration, respectively. Δ denotes the performance change (averaged over benchmarks) compared with the baseline.

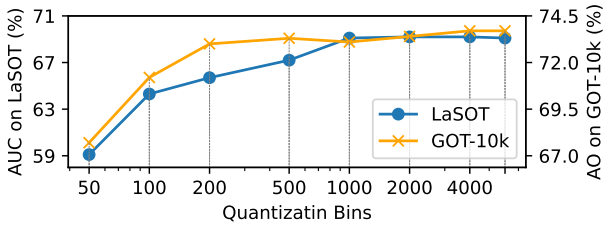

Figure 6. Influence of the number of quantization bins

we compare it with two other alternatives: 1) the feature concatenation of the search and template images, and 2) the *averaged* feature of the search and template images. For the first method, all the features are fed into the decoder. For the second method, the features are first projected into a 1D embedding, and then fed into the decoder. From Tab. 4 (#6 and #7), we can observe that these two methods perform comparable to the default method that only uses search image features. Overall, the decoder is not sensitive to the form of input visual features.

Prior Knowledge Integration. For the online update, our method utilizes the likelihood of generated tokens to select reliable templates. As reported in Tab. 4 (#8), our method improves the tracking performance. We also explore a naive online update method, in which the dynamic template is updated without selection. Tab. 4 (#9) shows this method obtains inferior performance. These results suggest that selecting reliable templates with our likelihood-based method is effective. For the window penalty, Tab. 4 (#10) demonstrates that the performance degrades without it.

Visualization of Cross Attention Map. To better understand how SeqTrack "reads out" the target state, we visualize the cross attention (averaged over heads) of the last decoder block. Fig. 7 shows cross attention maps as the model generates tokens. When generating the first token x, the attention is relatively diverse. When generating subsequent tokens, attention quickly concentrates on the target object. Attention is more focused on key information, *e.g.*, the person's arm and the zebra's tail when generating x and w, and the foot when generating y and h.

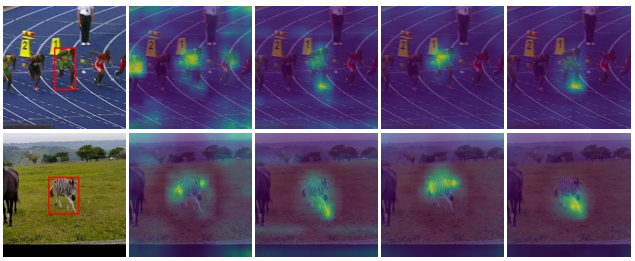

Figure 7. Decoder's cross attention to visual features when generating tokens. The first column is the search region image, and the second to last columns are the cross attention maps when x, y, w, h are generated, respectively.

#	Temporal	VOT2020 [25]	GOT-10k [20]	TrackingNet [37]
1	×	29.3	73.7	82.8
2	✓	30.1	74.9	83.0

Table 5. Ablation study of temporal sequence.

Temporal Sequence. At last, we present an additional experiment to demonstrate that our method is flexible to integrate temporal information during tracking. Specifically, we construct a temporal sequence containing the coordinates of the target object in four historical frames. The temporal sequence is prepended to the `START` token. When generating new tokens, all coordinate tokens of the historical frames are visible. In this way, the model perceives the historically moving trajectory of the target object. Tab. 5 shows that this simple approach improves the performance on several benchmarks. This demonstrates the potential of our SeqTrack method for modeling long-range temporal information in visual tracking.

5. Conclusion

This work proposes a new sequence-to-sequence tracking framework, *i.e.*, SeqTrack, that casts visual tracking as a sequence generation problem. It uses a simple encoder-decoder transformer architecture, getting rid of complicated head networks and loss functions. Extensive experiments demonstrate SeqTrack is effective, achieving competitive performance compared to state-of-the-art trackers. We hope this work could catalyze more compelling research on sequence learning for visual tracking.

Limitation. One limitation of SeqTrack is that, despite achieving competitive performance, it is difficult to handle the cases when objects move out-of-view or are occluded by distractors, because the method does not have explicit re-detection modules. Moreover, we build up the sequence model only using few video frames. A more promising solution is to model the entire video as a sequence, and teach the model to "read out" the target state frame by frame in an autoregressive manner. We will investigate this sequence-to-sequence modeling for long-term video in future work.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022. 2

[2] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H S Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016. 3, 4, 5, 7

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6182–6191, 2019. 1, 2, 6, 7

[4] Boyu Chen, Peixia Li, Lei Bai, Lei Qiao, Qiuhong Shen, Bo Li, Weihao Gan, Wei Wu, and Wanli Ouyang. Backbone is all your need: A simplified architecture for visual object tracking. In *ECCV*, pages 375–392, 2022. 1, 2, 6, 7

[5] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2021. 2

[6] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. *arXiv preprint arXiv:2206.07669*, 2022. 2

[7] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, pages 8126–8135, 2021. 1, 2, 3, 4, 5, 6, 7

[8] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *CVPR*, pages 6668–6677, 2020. 6

[9] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014. 2

[10] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *CVPR*, pages 13608–13618, 2022. 1, 2, 4, 5, 6

[11] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, pages 6638–6646, 2017. 7

[12] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019. 2, 4, 6, 7

[13] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 2, 4, 5

[15] 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. *IJCV*, pages 439–461, 2021. 6

[16] 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 *CVPR*, pages 5374–5383, 2019. 2, 5, 6, 7, 8

[17] Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. AiATrack: Attention in attention for transformer visual tracking. In *ECCV*, pages 146–164, 2022. 1, 6

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022. 5

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[20] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE TPAMI*, pages 1562–1577, 2019. 2, 5, 6, 8

[21] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, pages 784–799, 2018. 2

[22] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time MDNet. In *ECCV*, pages 83–98, 2018. 7

[23] 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*, pages 1125–1134, 2017. 6, 7

[24] Minji Kim, Seungkwan Lee, Jungseul Ok, Bohyung Han, and Minsu Cho. Towards sequence-level training for visual tracking. In *ECCV*, pages 534–551, 2022. 6

[25] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking VOT2020 challenge results. In *ECCV*, pages 547–601, 2020. 5, 7, 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 *CVPR*, pages 4282–4291, 2019. 2, 6

[27] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018. 1, 2, 3, 4, 5, 7

[28] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *ACM TIST*, pages 1–48, 2013. 4

[29] Liting Lin, Heng Fan, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. In *NeurIPS*, 2022. 6

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 2

[31] 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*, pages 740–755, 2014. 5

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 5

[33] Fan Ma, Mike Zheng Shou, Linchao Zhu, Haoqi Fan, Yilei Xu, Yi Yang, and Zhicheng Yan. Unified transformer tracker for object tracking. In *CVPR*, pages 8781–8790, 2022. 6

[34] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *CVPR*, pages 8731–8740, 2022. 6

[35] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *ICCV*, pages 13444–13454, 2021. 6

[36] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *ECCV*, pages 445–461, 2016. 6, 7

[37] 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 *ECCV*, pages 300–317, 2018. 5, 6, 8

[38] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016. 6

[39] Matthieu Paul, Martin Danelljan, Christoph Mayer, and Luc Van Gool. Robust visual tracking by segmentation. In *ECCV*, pages 571–588, 2022. 6

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, pages 8821–8831, 2021. 2

[41] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, pages 658–666, 2019. 2

[42] Jianbing Shen, Yuanpei Liu, Xingping Dong, Xiankai Lu, Fahad Shahbaz Khan, and Steven CH Hoi. Distilled siamese networks for visual tracking. *IEEE TPAMI*, pages 8896–8909, 2021. 6

[43] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W.H. Lau, and Ming-Hsuan Yang. VITAL: Visual tracking via adversarial learning. In *CVPR*, pages 8990–8999, 2018. 4

[44] Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Transformer tracking with cyclic shifting window attention. In *CVPR*, pages 8791–8800, 2022. 1, 2, 6

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112, 2014. 2

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 2, 3, 4

[47] Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by redetection. In *CVPR*, pages 6578–6588, 2020. 6

[48] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, pages 1571–1580, 2021. 1, 6, 7

[49] Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *CVPR*, pages 13763–13773, 2021. 6, 7

[50] Fei Xie, Chunyu Wang, Guangting Wang, Yue Cao, Wankou Yang, and Wenjun Zeng. Correlation-aware deep tracking. In *CVPR*, pages 8751–8760, 2022. 2, 6

[51] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020. 2

[52] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. In *ECCV*, pages 733–751, 2022. 6

[53] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, pages 10448–10457, 2021. 1, 2, 3, 4, 5, 6, 7

[54] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *CVPR*, pages 5289–5298, 2021. 7

[55] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *ECCV*, pages 341–357, 2022. 1, 2, 4, 5, 6, 7

[56] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R. Scott. Deformable siamese attention networks for visual object tracking. In *CVPR*, pages 6728–6737, 2020. 6

[57] Zhipeng Zhang, Yihao Liu, Xiao Wang, Bing Li, and Weiming Hu. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, pages 13339–13348, 2021. 6

[58] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, pages 771–787, 2020. 1, 6, 7