# Autoregressive Visual Tracking

Xing Wei[†] Yifan Bai[†] Yongchao Zheng[†] Dahu Shi[‡§] Yihong Gong[†✉]
[†]Xi'an Jiaotong University [‡]Zhejiang University [§]Hikvision Research Institute
{weixing, ygong}@mail.xjtu.edu.cn {yfbai, zyc}@stu.xjtu.edu.cn shidahu@zju.edu.cn

## Abstract

*We present* ARTrack, *an autoregressive framework for visual object tracking. ARTrack tackles tracking as a coordinate sequence interpretation task that estimates object trajectories progressively, where the current estimate is induced by previous states and in turn affects subsequences. This time-autoregressive approach models the sequential evolution of trajectories to keep tracing the object* ***across frames****, making it superior to existing template matching based trackers that only consider the* ***per-frame*** *localization accuracy. ARTrack is simple and direct, eliminating customized localization heads and post-processings. Despite its simplicity, ARTrack achieves state-of-the-art performance on prevailing benchmark datasets. Source code is available at* https://github.com/MIV-XJTU/ARTrack.

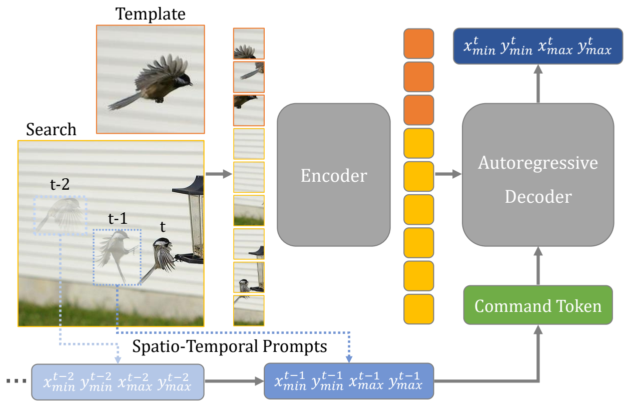

Template
Search
t-2
t-1
t
Encoder
Autoregressive Decoder
Command Token
Spatio-Temporal Prompts
$x^t_{min}\ y^t_{min}\ x^t_{max}\ y^t_{max}$
$x^{t-2}_{min}\ y^{t-2}_{min}\ x^{t-2}_{max}\ y^{t-2}_{max}$
$x^{t-1}_{min}\ y^{t-1}_{min}\ x^{t-1}_{max}\ y^{t-1}_{max}$

Figure 1. **Our ARTrack framework**. First, we embed visual features of template and search by an encoder. Then the coordinate tokens at current time step are interpreted by the decoder, conditioning on the previous estimates (as spatio-temporal prompts), and the command and visual tokens.

## 1. Introduction

Visual object tracking [5, 20, 34, 38, 48, 52] is a foundational objective in the realm of computer vision, whereby the tracker endeavors to estimate the location of an arbitrary target in each video frame, based on its initial state. Despite its ostensibly straightforward definition, the tracking task poses a significant challenge in real-world settings due to a variety of issues including but not limited to object deformation, scale variation, occlusion, and distraction from similar objects. Fortunately, visual tracking capitalizes on abundant temporal data as its input comprises a sequence of video frames. Observationally, humans leverage temporal information to gain a perception of the target's deformation, velocity, and acceleration trends, enabling them to maintain consistent tracking results in the face of indiscriminative or temporarily unavailable visual information.

The present mainstream approaches [10, 13, 45, 61, 64] for visual object tracking typically view it as a **per-frame template matching** problem, neglecting the potential temporal dependencies among the video frames. These methods generally follow three primary stages: (i) deep neural network-based feature extraction from the search and template images, (ii) an integration module using either convolution [2, 4] or attention mechanisms [10, 61] for feature matching/fusion, and (iii) bounding-box localization through customized heads for corner [13, 61], center/scale [64] estimation, and target classification [4, 61]. In some cases, the first two stages can be combined using a unified architecture [13, 64]. Post-processing techniques are usually employed during the localization step, such as Hanning window penalty [10, 56, 64, 68] and box optimization [4, 56]. Some methods incorporate a template update mechanism to improve the target feature representation. Representative techniques in this category include template image selection [61], feature integration [56], and time evolution [62, 66]. However, customized heads and post-processing techniques are complex and may require individual training and inference, which compromises the simple end-to-end framework. Moreover, tracking emphasizes preserving localization accuracy across the sequence, while conventional per-frame training methods prioritize immediate localization accuracy, resulting in an **objective mismatch** between training and inference [35].

This study proposes a novel framework to visual object tracking that differs from the mainstream methods, which typically employ a per-frame template matching task. Instead, the authors propose to consider tracking as **coordinate sequence interpretation**, with the objective of learning a simple end-to-end model for direct trajectory estimation. The proposed approach is based on the idea that given a sequence of frames and an initial object box, the tracker should "interpret" a sequence of coordinates that trace the object, in a manner similar to a language modeling task. The proposed framework models the sequential evolution of object trajectories across frames by decoding the entire trajectory sequence step by step. The current estimate is influenced by previous states and in turn influences the subsequences, thus unifying the task objectives of training and inference. Furthermore, the proposed approach simplifies the tracking pipeline by avoiding customized heads and post-processings, relying instead on direct coordinate regression.

The proposed autoregressive visual tracking framework, called ARTrack, is depicted in Figure 1. The first step in this framework is to construct discrete token sequences from object trajectories using a quantization and serialization scheme [8]. The framework then adopts an encoder-decoder architecture to perceive visual information and generate target sequences gradually. In this autoregressive framework, the previous outcomes serve as **spatio-temporal prompts**, propagating preceding motion dynamics into succeeding frames for more coherent tracking results. Notably, the model is trained using a structured loss function that maximizes the likelihood of the target sequence, consistent with the task objective at test time. The authors demonstrate the efficacy of this approach through extensive experiments, showing that the simple and neat ARTrack framework achieves state-of-the-art results on prevailing tracking benchmarks, outperforming other highly customized trackers.

## 2. Related Work

**Tracking framework.** Current prevailing trackers [2, 10, 15, 23, 36, 37, 54, 69] typically rely on the matching between the template and search images. The core design is the integration module for feature fusion. To address the issue of target appearance variations along the temporal dimension, some online methods [4,12,15,16,29,34,38,44] learn a target-dependent model for online template update, which usually needs separate training. They also require post-processings, such as Hanning window penalty [10, 56, 64, 68], and box optimization [4, 56].

By comparison, few single-object tracking methods [40] in recent years focus on utilizing the motion information, while it is prevalent in multi-object tracking [3, 11, 21, 26, 27, 59]. These methods usually integrate a motion model to utilize the motion information, generating proposals which are then associated with the results from a pre-defined detector, *e.g.* RAN [21] use a recurrent autoregressive network for online multi-object tracking. Very recently, the improved version of SwinTrack [40] add a novel motion token to incorporate temporal context for tracking. In this paper, we propose a simple approach to conduct visual template matching and motion modeling in a unified framework.

**Transformer in visual tracking.** The attention mechanisms have been employed in recent trackers, including those mentioned in references [6, 10, 13, 56, 61, 64]. For instance, TransT [10] utilizes attention to fuse features and establish long-distance feature associations while adaptively focusing on relevant information. MixFormer [13] uses iterative mixed attention to integrate feature extraction and target information. OSTrack [64] applies an early candidate elimination module to eliminate unnecessary search region tokens. In contrast, our model is a straightforward encoder-decoder architecture without any specialized heads, resulting in a straightforward and pure transformer-based tracker.

**Language modeling for vision.** In recent years, there has been significant progress in language modeling. Some methods aim to create a joint representation model for language and vision tasks, such as those proposed in [1, 39, 43]. One particular method, Pix2Seq [8, 9], formulates vision tasks as language modeling tasks conditioned on pixel token input. By representing bounding boxes and class labels as discrete sequences, this method unifies computer vision tasks. Inspired by Pix2Seq, we introduce the language modeling framework into visual object tracking, constructing a time-autoregressive model for direct trajectory estimation. Our approach simplifies the tracking framework, eliminating unnecessary post-processing, and decodes object coordinates step by step with coherent spatio-temporal prompts.

## 3. Tracking as Sequence Interpretation

We cast visual tracking as a sequential coordinate interpretation task, formulated as a conditional probability:

$$P\left(\boldsymbol{Y}^t|\boldsymbol{Y}^{t-N:t-1}, (\boldsymbol{C}, \boldsymbol{Z}, \boldsymbol{X}^t)\right), \quad (1)$$

where $\boldsymbol{Z}$ and $\boldsymbol{X}^t$ are the given template and search images at time step $t$, $\boldsymbol{C}$ is the command token, and $\boldsymbol{Y}$ denotes the target sequence associated with $\boldsymbol{X}$. Template $\boldsymbol{Z}$ could also be renewed at each time step with an updating mechanism [13, 56], or simply being the initial one [40, 64]. As can be seen, we formulate tracking as a time-autoregressive process where the current outcome is a function of the recent $N$ pasts, conditioned on the template and the search image. This is an autoregressive model [32, 63] of order $N$, referred to as the AR($N$) model for short. Specifically when $N = 0$, Equation (1) degenerates to a per-frame model $P(\boldsymbol{Y}^t|\boldsymbol{C}, \boldsymbol{Z}, \boldsymbol{X}^t)$, which is not conditioned on the previous states. The introduced autoregressive model is compatible with visual tracking as it is a sequence prediction task per se. The estimated target state in the current frame is influenced by adjacent preceding target states and also affects subsequent frames. We term this tracking framework ARTrack, and it consists of the following main components.

- Sequence construction: Given a video sequence and an initial object box, the visual tracker predicts a sequence of bounding boxes. They are mapped into a unified coordinate system and converted into discrete token sequences with a shared vocabulary.
- Network architecture: We use an encoder-decoder architecture, where the encoder embeds visual features and the decoder interprets the target sequence.
- Objective function: The model is trained over video frames with a structured loss function to maximize the log-likelihood of the target sequence. We also explore a task-specific objective to improve performance.

## 3.1. Sequence Construction from Object Trajectory

We describe the object trajectories as discrete token sequences with a shared vocabulary.

**Tokenization.** Inspired by the Pix2Seq framework [8], we discretize continuous coordinates to avoid a large number of parameters required to describe the continuous coordinates, which is called tokenization. Specifically, the object box at time step $t$ is composed of four tokens, *i.e.* $[x^t_{\min}, y^t_{\min}, x^t_{\max}, y^t_{\max}]$, each of which is an integer between $[1, n_{\text{bins}}]$. When the number of *bins* is greater than or equal to the image resolution, zero quantization error can be achieved. Then we use the quantized term to index a learnable vocabulary to get the token corresponding to the coordinate. This allows the model to depict the location of object in terms of discrete tokens, and also allows the off-the-shelf decoders in the language model to be used for coordinate regression. This novel regression avoids direct non-linear mapping from image features to coordinates, which is often difficult. In detokenization, we match the output token feature with the shared vocabulary to find the most likely location.

**Trajectory coordinate mapping.** Most trackers crop a search region to reduce computation cost instead of tracking on the full-resolution frame [10, 13, 40, 61, 64]. This means that the network outputs the coordinates of the object in the current frame, relative to the search region [28]. To obtain a unified representation, it is necessary to map the boxes of different frames into the same coordinate system. In our method, we cache the box coordinates in the global coordinate system for the preceding $N$ frames and map them to the current coordinate system after the search region is cropped. However, if we use the full frame for searching, this coordinate mapping step is no more necessary.

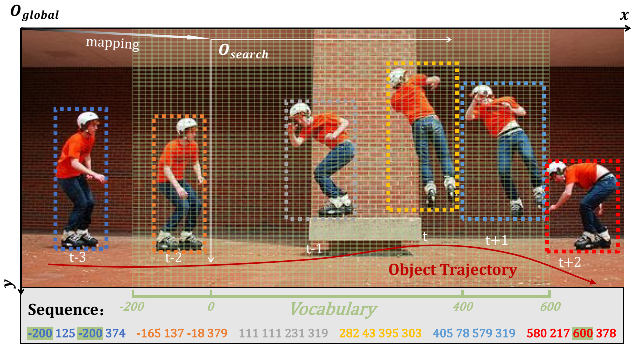


Figure 2. **Sequence construction and coordinate mapping**. The object trajectory is constructed by using coordinates from previous frames in the global coordinate system. During tracking, the trajectory is mapped to the current coordinate system to construct a sequence. Any coordinates that are out-of-range are clamped and masked in green. To index a vocabulary, we discretize continuous coordinates into quantized terms. The representation range of the vocabulary covers the range of the search region.

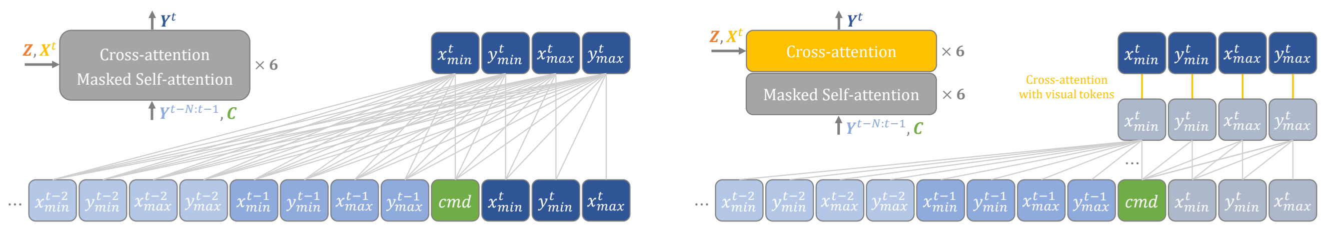
**Representation range of vocabulary.** The representation range of the vocabulary can be set based on the size of the search region, but the preceding trajectory sequence may sometimes extend beyond the boundaries of the search region due to rapid object movement. To account for this, we expand the representation range by a multiple of the search region range (for instance, if the search region range is $[0.0, 1.0]$, we expand it to $[-0.5, 1.5]$). This enables the vocabulary to include coordinates that lie outside the search region, which in turn allows the model to capture more preceding motion cues for tracking and to predict bounding boxes that extend beyond the search region.

## 3.2. Network Architecture

Given the target sequences that have constructed from object trajectories, we use an encoder-decoder structure for learning and inference. Such a network architecture is widely used in modern visual recognition [7, 50, 65] and language modeling [49, 53].

**Encoder.** The encoder can be a general image encoder that encodes pixels into hidden feature representations, such as ConvNet [25, 51], vision Transformer (ViT) [17, 24, 58], or a hybrid architecture [60]. In this work, we use the same ViT encoder as OSTrack [64] for visual feature encoding. The template and search images are first split into patches, flattened and projected to generate a sequence of token embeddings. Then we add template and search tokens with positional and identity embeddings, concatenate and feed them into a plain ViT backbone to encode visual features.

**Decoder.** We use a Transformer decoder for target sequence generation. It decodes the whole sequence progres-



(a) default decoder (b) altered decoder

Figure 3. **The default and altered decoders**. We explore two types of decoders: (a) decodes the whole sequence progressively, conditioned on preceding coordinate tokens, a command token, and visual features. (b) is similar to (a), whose self- and cross- attention layers are decoupled and stacked individually. And it conducts the cross-attention with visual features in parallel.

sively, conditioned on preceding coordinate tokens, a command token, and visual features. The preceding coordinate tokens ($\boldsymbol{Y}^{t-N:t-1}$) serve as *spatio-temporal prompts*, propagating motion dynamics into succeeding frames. The command token ($\boldsymbol{C}$) provides a trajectory proposal and then matches the template ($\boldsymbol{Z}$) with the search ($\boldsymbol{X}^t$) for more accurate coordinate prediction ($\boldsymbol{Y}^t$). This simple decoding approach removes the complexity and customization in architectures of modern visual trackers, *e.g.*, localization heads and post-processings, since the coordinate can be immediately detokenized from the shared vocabulary. The decoder works with two kinds of attention. The self-attention (with causal mask) is performed among coordinate tokens to convey spatio-temporal information. And the cross-attention combines motion cues with visual cues to make the final prediction. The two operations are performed alternatively in each decoder layer to mix the two kinds of embeddings. We illustrate decoder's structure in Figure 3a. To improve tracking efficiency, we investigate an altered decoder by modifying the decoder layer. Specifically, the self- and cross- attention layers are decoupled and stacked individually. In this way, we can conduct the cross-attention on visual features in parallel, which is the most time-consuming computation in decoder. The altered decoder is illustrated in Figure 3b.

## 3.3. Training and Inference

ARTrack is a simple framework that enables end-to-end training and inference.

**Training.** Beyond per-frame training and optimization, ARTrack is learned over video sequences. It adopts a structured objective that maximizes the log-likelihood of token sequences with a softmax cross-entropy loss function:

$$\texttt{maximize} \sum_{t=1}^{T} \log P\left(\boldsymbol{Y}^t | \boldsymbol{Y}^{t-N:t-1}, (\boldsymbol{C}, \boldsymbol{Z}, \boldsymbol{X}^t)\right), \quad (2)$$

where $T$ is the length of the target sequence. This learning method unifies the task objectives between training and inference, *i.e.*, maintaining the localization accuracy across video frames. At startup ($t \leq N$), the cached spatio-temporal prompts ($\boldsymbol{Y}^{t-N:t-1}$) are filled with the initial one ($\boldsymbol{Y}^1$) and gradually updated with new predictions.

This is generic objective function [8] that ignores token's physical properties, *e.g.*, the spatial relation of coordinates. Though we find such a task-agnostic objective is effective to train the model, we investigate how to incorporate task knowledge to improve performance. Specifically, we introduce the SIoU loss [22] to better measure the spatial correlation between the predicted and ground truth bounding boxes. We first get the coordinate token from the estimated probability distribution. As sampling is not differentiable, we apply the expectation of distribution to express the coordinate. We then get the predicted bounding box and calculate its SIoU with the ground truth. The whole loss function can be written as:

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \lambda \mathcal{L}_{\text{SIoU}}, \quad (3)$$

where $\mathcal{L}_{\text{ce}}$ and $\mathcal{L}_{\text{SIoU}}$ are the cross-entropy loss and SIoU loss, respectively, and $\lambda$ is a weight to balance the two loss terms.

**Inference.** At inference time, we sample tokens from the model likelihood $P\left(\boldsymbol{Y}^t | \boldsymbol{Y}^{t-N:t-1}, (\boldsymbol{C}, \boldsymbol{Z}, \boldsymbol{X}^t)\right)$ using the `argmax` sampling. We find that other stochastic sampling techniques [30] or the expectation perform comparably with `argmax` sampling. There is no need to end the sequence prediction with an additional EOS token, since the sequence length is fixed in our problem. After obtaining the discrete tokens, we de-quantize them to get continuous coordinates.

## 4. Experiments

### 4.1. Implementation Details

**Model variants.** We train three variants of ARTrack with different configurations as follows:

- **ARTrack$_{256}$.** Backbone: ViT-Base; Template size: [128×128]; Search region size: [256×256];
- **ARTrack$_{384}$.** Backbone: ViT-Base; Template size: [192×192]; Search region size: [384×384];

| Methods | GOT-10k* | | | TrackingNet | | | LaSOT | | | LaSOText | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AO(%) | $SR_{0.5}$(%) | $SR_{0.75}$(%) | AUC(%) | $P_{Norm}$(%) | $P$(%) | AUC(%) | $P_{Norm}$(%) | $P$(%) | AUC(%) | $P_{Norm}$(%) | $P$(%) |
| SiamFC$_{255}$ [2] | 34.8 | 35.3 | 9.8 | 57.1 | 66.3 | 53.3 | 33.6 | 42.0 | 33.9 | 23.0 | 31.1 | 26.9 |
| MDNet$_{107}$ [48] | 29.9 | 30.3 | 9.9 | 60.6 | 70.5 | 56.5 | 39.7 | 46.0 | 37.3 | 27.9 | 34.9 | 31.8 |
| ECO$_{224}$ [16] | 31.6 | 30.9 | 11.1 | 55.4 | 61.8 | 49.2 | 32.4 | 33.8 | 30.1 | 22.0 | 25.2 | 24.0 |
| SiamRPN++$_{255}$ [36] | 51.7 | 61.6 | 32.5 | 73.3 | 80.0 | 69.4 | 49.6 | 56.9 | 49.1 | 34.0 | 41.6 | 39.6 |
| DiMP$_{288}$ [44] | 61.1 | 71.7 | 49.2 | 74.0 | 80.1 | 68.7 | 56.9 | 65.0 | 56.7 | 39.2 | 47.6 | 45.1 |
| SiamR-CNN$_{255}$ [54] | 64.9 | 72.8 | 59.7 | 81.2 | 85.4 | 80.0 | 64.8 | 72.2 | - | - | - | - |
| MAMLTrack$_{263}$ [55] | - | - | - | 75.7 | 82.2 | 72.5 | 52.3 | - | - | - | - | - |
| LTMU$_{288}$ [14] | - | - | - | - | - | - | 57.2 | - | 57.2 | 41.4 | 49.9 | 47.3 |
| Ocean$_{255}$ [68] | 61.1 | 72.1 | 47.3 | - | - | - | 56.0 | 65.1 | 56.6 | - | - | - |
| TrDiMP$_{352}$ [56] | 67.1 | 77.7 | 58.3 | 78.4 | 83.3 | 73.1 | 63.9 | - | 61.4 | - | - | - |
| SLT-TrDiMP$_{352}$ [35] | 67.5 | 78.8 | 58.7 | 78.1 | 83.1 | 73.1 | 64.4 | 73.5 | - | - | - | - |
| TransT$_{256}$ [10] | 67.1 | 76.8 | 60.9 | 81.4 | 86.7 | 80.3 | 64.9 | 73.8 | 69.0 | - | - | - |
| AutoMatch$_{255}$ [67] | 65.2 | 76.6 | 54.3 | 76.0 | - | 72.6 | 58.3 | - | 59.9 | - | - | - |
| KeepTrack$_{352}$ [45] | - | - | - | - | - | - | 67.1 | 77.2 | 70.2 | 48.2 | - | - |
| STARK$_{320}$ [61] | 68.8 | 78.1 | 64.1 | 82.0 | 86.9 | - | 67.1 | 77.0 | - | - | - | - |
| SwinTrack-T$_{224}$ [40] | 71.3 | 81.9 | 64.5 | 81.1 | - | 78.4 | 67.2 | - | 70.8 | 47.6 | - | 53.9 |
| SwinTrack-B$_{384}$ [40] | 72.4 | 80.5 | 67.8 | 84.0 | - | 82.8 | 71.3 | - | 76.5 | 49.1 | - | 55.6 |
| MixFormer-22k$_{320}$ [13] | 70.7 | 80.0 | 67.8 | 83.1 | 88.1 | 81.6 | 69.2 | 78.7 | 74.7 | - | - | - |
| MixFormer-L$_{320}$ [13] | - | - | - | 83.9 | 88.9 | 83.1 | 70.1 | 79.9 | 76.3 | - | - | - |
| OSTrack$_{256}$ [64] | 71.0 | 80.4 | 68.2 | 83.1 | 87.8 | 82.0 | 69.1 | 78.7 | 75.2 | 47.4 | 57.3 | 53.3 |
| OSTrack$_{384}$ [64] | 73.7 | 83.2 | 70.8 | 83.9 | 88.5 | 83.2 | 71.1 | 81.1 | 77.6 | 50.5 | 61.3 | 57.6 |
| **ARTrack$_{256}$** | 73.5 | 82.2 | 70.9 | 84.2 | 88.7 | 83.5 | 70.4 | 79.5 | 76.6 | 46.4 | 56.5 | 52.3 |
| **ARTrack$_{384}$** | 75.5 | 84.3 | 74.3 | 85.1 | 89.1 | 84.8 | 72.6 | 81.7 | 79.1 | 51.9 | 62.0 | 58.5 |
| **ARTrack-L$_{384}$** | **78.5** | **87.4** | **77.8** | **85.6** | **89.6** | **86.0** | **73.1** | **82.2** | **80.3** | **52.8** | **62.9** | **59.7** |

Table 1. State-of-the-art comparison on GOT-10k [31], TrackingNet [47], LaSOT [19] and LaSOText [18]. Where * denotes for tracker only trained on GOT-10k. The number in subscript denotes the search region resolution. Best in **bold**, second best underlined.

- **ARTrack-L$_{384}$.** Backbone: ViT-Large; Template size: [192×192]; Search region size: [384×384];

**Training strategy.** We follow the conventional protocols to train our models. The training set consists of GOT-10k [31] (we removed 1k sequences in GOT-10k train split according to [61]), LaSOT [19] and TrackingNet [47]. For performance evaluation on GOT-10k especially, the models are trained on the full GOT-10k training split. Unlike the traditional per-frame training which uses random translation and scale transformation for ground truth to simulate spatial dithering, our sequential training allows us to interpret a sequence of coordinates tracing the target frame by frame without any augmentations. We optimized the model with AdamW [42], the learning rate of the backbone is $4 \times 10^{-7}$ and $4 \times 10^{-6}$ for other parameters.We train the network for 60 epochs with 960 video sequences every epoch. Each sequence contains 16 frames due to the GPU memory constraint.

More about, to compare with mainstream trackers fairly, we first pre-train the AR(0) model that can leverage image datasets such as COCO2017 [41] to align with other per-frame trained trackers. The AR(0) training set consists of four datasets, which are the same as DiMP [44] and STARK [61]. We utilize the same data augmentations as OSTrack [64], including horizontal flip and brightness jittering. With optimized by AdamW, the learning rate of the backbone is $8 \times 10^{-6}$ and $8 \times 10^{-5}$ for other parameters. Our AR(0) model is trained with 240 epochs and 60k matching pairs per epoch.

## 4.2. Main Results

We evaluate the performance of our proposed ARTrack$_{256}$, ARTrack$_{384}$ and ARTrack-L$_{384}$ on several benchmarks, including GOT-10k [31], TrackingNet [47], LaSOT [19], LaSOText [18], TNL2K [57], UAV123 [46] and NFS [33].

**GOT-10k [31].** GOT-10k is a large-scale dataset that contains over 10,000 video sequences of frames with high-precision bounding boxes. It promotes the one-shot tracking rule, which means that the classes between the training and test sets do not overlap. In accordance with this protocol, we trained our ARTrack solely on the GOT-10k train split and evaluated the test results. As reported in Table 1, our ARTrack$_{384}$ outperformed the state-of-the-art trackers on all indicators. Moreover, our ARTrack-L$_{384}$ model, trained only on the GOT-10k train split, has achieved better performance than other mainstream trackers trained on additional datasets. This strongly verifies that our tracker has strong generalization and is not sensitive to categories. It also demonstrates that our model can capture more precise details, which enables it to perform well in one-shot datasets.

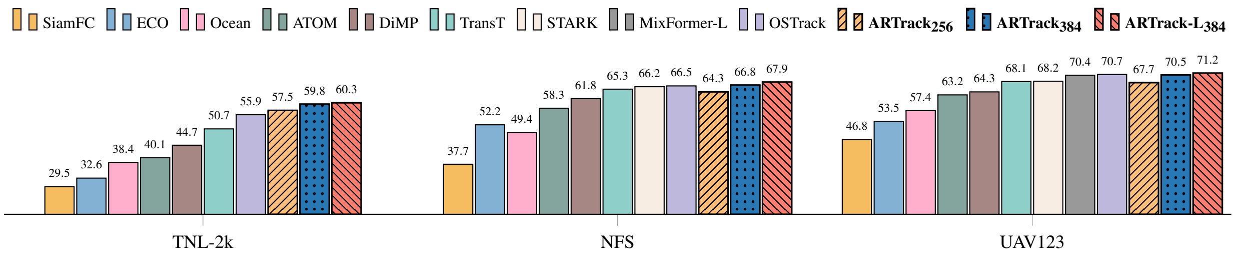


Figure 4. State-of-the-art comparison on TNL2K [57], NFS [33] and UAV123 [46].

**TrackingNet [47].** TrackingNet is a dataset for tracking that covers a diverse range of object classes and scenes in the real world. Its test set contains 511 sequences for which only the primary frame's annotations are provided. We evaluated our trackers on the TrackingNet test set and submitted the results to the official evaluation server. The results are presented in Table 1, and we observed that even our ARTrack$_{256}$ achieved a new state-of-the-art performance on this large-scale benchmark. When using higher input resolution, our ARTrack$_{384}$ and ARTrack-L$_{384}$ models surpassed all other trackers and achieved the best performance.

**LaSOT [19].** LaSOT is a large-scale benchmark consisting of 280 videos in its test set, which is effective in detecting the performance of long-term tracking. We evaluated our ARTrack on the test set and the results are shown in Table 1. Our ARTrack$_{256}$ already outperforms MixFormer-L, which has a larger backbone. Furthermore, our ARTrack$_{384}$ achieved the top-ranked performance on AUC, reaching 72.6%, surpassing all other trackers without the use of any online template update strategy or post-processing.

**LaSOT$_{ext}$ [18].** LaSOT$_{ext}$ is an extended subset of LaSOT that includes 150 additional videos from 15 new categories. These videos feature numerous similar interfering objects and fast-moving small objects, which significantly increase the tracking difficulty. Although our ARTrack$_{256}$'s performance is lower than the state-of-the-art trackers, we believe that due to the lower resolution causes the lack of utilizing motion information. We can observe our ARTrack$_{384}$ and ARTrack-L$_{384}$ outperforms OSTrack$_{384}$ by 1.4% AUC and 2.3% AUC with higher resolution.

**TNL2K [57], NFS [33] and UAV123 [46].** In order to demonstrate the robustness of our model to deal with complex scenarios, we evaluated on other three benchmarks: (i) TNL2K is a high quality and multimodal dataset with natural language tagging, (ii) NFS is a dataset with higher frame rate (240fps) video, and (iii) UAV123 composed of complex scene video clips shot by different UAVs. Figure 4 shows that our ARTrack-L$_{384}$ (▧) and ARTrack$_{384}$ (▦) perform better than other trackers in general.

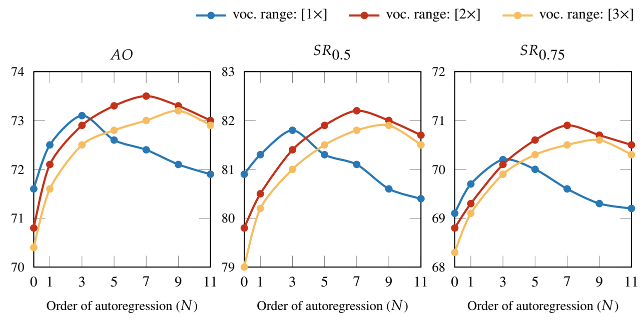


Figure 5. **Performance *vs.* the order of autoregression**. The —●—, —●—, and —●— curves denote the representation ranges of vocabulary, which are 1×, 2×, and 3× long to the search region, respectively.

## 4.3. Analysis of the Autoregressive Model

We analyze the main properties of the ARTrack framework. For the following experimental studies, we follow GOT-10k test protocol unless otherwise noted. Default settings are marked in gray.

**Order of autoregression.** The core of ARTrack is autoregression, which is controlled by the length of the spatio-temporal prompts, or the order ($N$). This parameter determines how much preceding trajectory information can be utilized. For instance, with $N = 1$, we can infer the target scale and aspect ratio based on the previous time step, and with $N = 2$, we can also learn a coarse moving direction. Increasing $N$ provides more motion information. We experiment with different values of $N$ to examine its impact on the model.

One way to set the range of vocabulary representation is to use the same range as the search region, which is illustrated by the (—●—) curves (voc. range: [1×]) in Figure 5. As shown, incorporating spatio-temporal prompts through ARTrack improves the AO score by almost 1.0% compared to using $N = 0$, which is a pure per-frame model. Moreover, increasing $N$ leads to a significant boost in AO score from 71.6% to 73.1%. However, when $N > 3$, the precision declines due to a higher number of invalid coordinates falling outside the representation range.

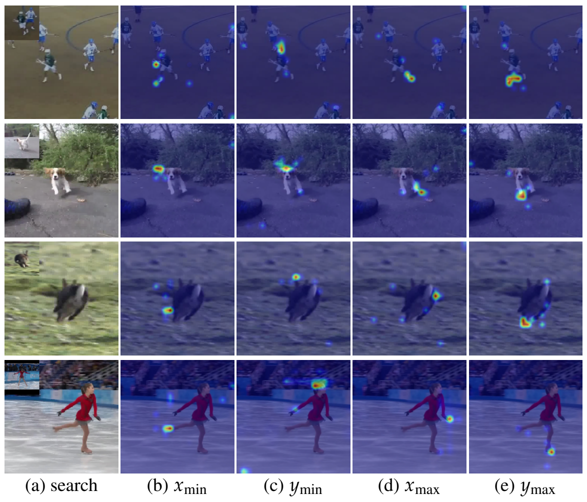


Figure 6. **Decoder's cross-attention**. (a): Search region and template image (in the top-left corner). (b)-(e): Corresponding coordinates token-to-search attention maps in the last layer of decoder.

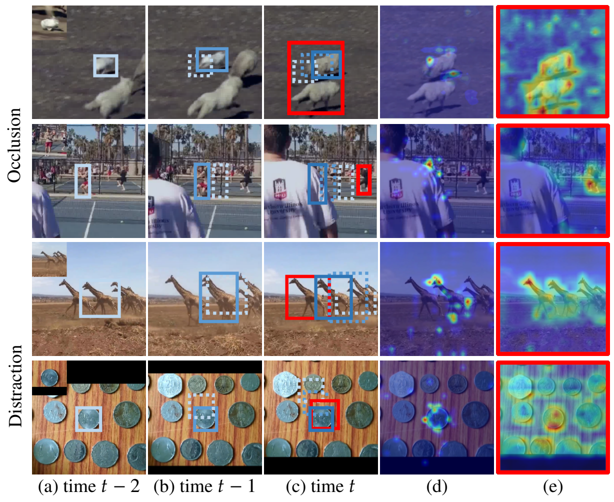


Figure 7. (a)-(c): Search region and predicted boxes. The blue and red boxes denote the predictions of ours and OSTrack, respectively. (d): Attention map of ARTrack. (e): Attention map of OSTrack.

As suggested in Section 3.1, we expand the vocabulary range appropriately to mitigate the truncation of trajectories caused by coordinates beyond the representation range. The effect of this expansion is demonstrated in Figure 5 with the (●) curves (voc. range: [2×]). By doing so, the model is not only able to capture more preceding motion cues for more coherent tracking results but also predict bounding boxes that exceed the search region. This approach proves to be effective and outperforms the naive [1×] setting by 0.4% (73.1% *vs.* 73.5%).

However, expanding the representation range poses a challenge to the localization of bounding boxes. As the range increases, it becomes increasingly difficult to assign the appropriate $bin$ to its corresponding coordinate accurately. This is why the [2×] setting results in a lower AO score when $N$ is small. Similarly, although the precision of the [3×] setting improves with increased synchronization with $N$, as depicted in the (●) curves, it still falls short of the best performance. Unfortunately, due to hardware memory constraints, we were unable to train using a larger $N$.

**Qualitative comparison.** To gain a better understanding of our time-autoregressive model, we generate cross-attention maps while predicting coordinate tokens sequentially. In order to test the robustness of our model, we use complex scenarios encountered in real-world tracking, such as motion blur, rotation, aspect change, and camera motion, as shown in Figure 6. Interestingly, in each scenario, our tracker focuses on the appropriate *extremities* when predicting each coordinate, demonstrating our model's ability for precise localization.

When faced with more challenging scenarios like occlusion and distraction, per-frame template matching can be unreliable. The target in the former may become invisible, while the presence of various similar objects in the latter can confuse the tracker. To overcome these problems, our method leverages preceding motion cues to generate a reasonable prediction in situations where visual features are not discriminative.

In Figure 7, we present the cross-attention maps generated by ARTrack frame-by-frame and compare them with the attention map estimated by OSTrack [64]. To obtain an instance-level visualization, we sum over the cross-attention maps at the last layer when predicting each coordinate. The first two rows of the figure demonstrate the occlusion scenario. Our method can predict a reasonable target bounding box even when encountering full occlusion by conditioning on the preceding trajectory sequence. On the other hand, the attention in OSTrack is wrongly assigned to other instances, which is understandable since it is difficult for humans to locate targets without observing them. However, humans can track invisible objects given the preceding trajectory sequence of the target. Similar findings can be inferred in the case of the distraction scenario, which is depicted in the last two rows. When there are numerous similar objects in the search image, the attention of OSTrack gets distracted, leading to erroneous tracking. In contrast, ARTrack can maintain the focus on the target by considering the prior states. This supports our claim that our method can effectively model the sequential evolution of object trajectories across frames.

**Bins per pixel.** To investigate the impact of the resolution of bins (i.e., bins per pixel) on performance, we fixed the resolution of the search region to 256 pixels and used a vocabulary with a representation range twice as long as the range of the search region. We then varied the number of

| Bins per pixel | AO(%) | $SR_{0.5}$(%) | $SR_{0.75}$(%) |
|---|---|---|---|
| 512/[2 × 256] | 71.8 | 80.2 | 68.3 |
| **800/[2 × 256]** | **73.5** | **82.2** | **70.9** |
| 1200/[2 × 256] | 72.6 | 80.8 | 69.3 |
| 1600/[2 × 256] | 71.9 | 80.1 | 68.5 |

Table 2. Experimental studies on **bins per pixel**.

bins per pixel, as shown in Table 2.

As there is no significant error caused by quantization when the number of bins is larger than the longer side of the cropped image, we first use 512 bins (512/[2×256] bins per pixel) and then increase the number. As shown in Table 2, increasing the number of bins that enable subpixel quantization accuracy can improve performance. However, the conclusion differs slightly from that of [8], which suggests that a smaller number of bins is sufficient for accurate object detection. We believe that this difference may be due to the need for more quantization accuracy for precise motion modeling. Using much more bins (*e.g.* 1600) can significantly increase the vocabulary size and slow down training.

| $\lambda_{\text{CE}}$ | $\lambda_{\text{SIoU}}$ | AO(%) | $SR_{0.5}$(%) | $SR_{0.75}$(%) |
|---|---|---|---|---|
| ✓ | | 72.6 | 81.2 | 69.5 |
| | ✓ | 72.2 | 81.1 | 67.6 |
| ✓ | ✓ | **73.5** | **82.2** | **70.9** |

Table 3. A task-specific **loss function** can improve performance.

**Loss function.** Table 3 showcases the effectiveness of integrating specific tracking knowledge with task-agnostic objectives. We observed that combining SIoU and CE loss resulted in better performance than using either of them alone. This can be attributed to the fact that when computing SIoU, we consider the expected bounding box location, which accounts for the spatial relationships, thereby enhancing the robustness of supervision. Using only the SIoU loss without CE led to a significant decrease in $SR_{0.75}$, but $SR_{0.5}$ remained the same as when using the CE loss. We speculate that this is because the model was solely supervised by the expected coarse-grained location and lacked the ability to generate more precise bounding boxes.

## 4.4. Limitation Analysis

| Architecture | AO(%) | $SR_{0.5}$(%) | $SR_{0.75}$(%) | FPS |
|---|---|---|---|---|
| **default decoder** | **73.5** | **82.2** | **70.9** | **26** |
| altered decoder | 73.2 | 81.7 | 70.6 | 45 |

Table 4. Performance comparison of **decoder variants.** The altered decoder can improve tracking speed significantly.

**Speed analysis and architecture variant.** A major limitation of the ARTrack framework is that it is not as efficient as recently proposed trackers, due to its serial computation in the decoder. We investigate an altered decoder composed of self- and cross- attention layers stacked individually. Specifically, several self-attention layers process coordinate tokens in an autoregressive manner, followed by parallel cross-attention layers to aggregate visual features. The altered decoder can improve inference speed significantly (73% speed-up) with a bit of sacrifice on the accuracy (diminished by 0.3% on AO score), as reported in Table 4.

| Training Strategy | LaSOT | | |
|---|---|---|---|
| | AUC(%) | $P_{norm}$(%) | $P$(%) |
| **w/ AR(0) pre-train** | **70.4** | **79.5** | **76.6** |
| w/o AR(0) pre-train | 69.2 | 78.6 | 75.5 |

Table 5. **Training strategy**. Pre-train obtains better performance.

**Training strategy and command token analysis.** In order to make a fair comparison with previous trackers [10, 13, 40, 61, 64] that have been trained on diverse image datasets such as COCO2017 [41], we first pre-trained our model with $N = 0$. This allowed our time-autoregressive model to function like a per-frame model temporarily, without relying on previous states. We then tested our model on the LaSOT benchmark, and the results are shown in Table 5, which reported a 1.2% improvement on the AUC score with pre-training. However, the cost is that we needed to use a learnable command token to initiate the autoregressive process, and this token had to be preserved to ensure consistency between the per-frame and sequential training.

## 5. Conclusion

We propose ARTrack, a simple and direct end-to-end autoregressive framework for visual object tracking. We regard visual tracking as a coordinate sequence interpretation task, thus we employ language modeling for simultaneous visual template matching and motion information modeling. The tracker is a general encoder-decoder architecture that eliminates customized heads and post-processings to simplify the tracking pipeline. More about, we present spatio-temporal prompts modeling the sequential evolution of trajectory propagating motion cues for more coherent tracking results. Extensive experiments prove our tracker outperforms other mainstream trackers and achieves state-of-the-art on prevailing benchmark datasets. In the future, we hope this framework could be extended to other video tasks.

**Acknowledgements.** This work was funded by the National Natural Science Foundation of China under Grant No. 62006183 and No. U21B2048, and by the National Key Research and Development Project of China under Grant No. 2020AAA0105600.

## References

[1] Shuai Bai, Zhedong Zheng, Xiaohan Wang, Junyang Lin, Zhu Zhang, Chang Zhou, Hongxia Yang, and Yi Yang. Connecting language and vision for natural language-based vehicle retrieval. In *CVPR*, 2021. 2

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016. 1, 2, 5

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016. 2

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 1, 2

[5] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 1

[6] Ziang Cao, Changhong Fu, Junjie Ye, Bowen Li, and Yiming Li. Hift: Hierarchical feature transformer for aerial tracking. In *ICCV*, 2021. 2

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[8] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2022. 2, 3, 4, 8

[9] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. In *NeurIPS*, 2022. 2

[10] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 1, 2, 3, 5, 8

[11] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *ICCV*, 2017. 2

[12] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Fully convolutional online tracking. *CVIU*, 2022. 2

[13] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *CVPR*, 2022. 1, 2, 3, 5, 8

[14] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-performance long-term tracking with meta-updater. In *CVPR*, 2020. 5

[15] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, 2019. 2

[16] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017. 2, 5

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[18] Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Mingzhen Huang, Juehuan Liu, Yong Xu, Chunyuan Liao, Lin Yuan, and Haibin Ling. Lasot: A high-quality large-scale single object tracking benchmark. *IJCV*, 2021. 5, 6

[19] 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*, 2019. 5, 6

[20] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019. 1

[21] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *WACV*, 2018. 2

[22] Zhora Gevorgyan. Siou loss: More powerful learning for bounding box regression. *arXiv:2205.12740*, 2022. 4

[23] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *CVPR*, 2020. 2

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[26] Yuhang He, Xing Wei, Xiaopeng Hong, Wei Ke, and Yihong Gong. Identity-quantity harmonic multi-object tracking. *TIP*, 2022. 2

[27] Yuhang He, Xing Wei, Xiaopeng Hong, Weiwei Shi, and Yihong Gong. Multi-target multi-camera tracking by tracklet-to-target assignment. *TIP*, 2020. 2

[28] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 3

[29] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 2014. 2

[30] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020. 4

[31] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2019. 5

[32] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020. 2

[33] 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. 5, 6

[34] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017. 1, 2

[35] Minji Kim, Seungkwan Lee, Jungseul Ok, Bohyung Han, and Minsu Cho. Towards sequence-level training for visual tracking. In *ECCV*, 2022. 1, 5

[36] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 2, 5

[37] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 2

[38] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming-Hsuan Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, 2018. 1, 2

[39] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. In *ACL*, 2020. 2

[40] Liting Lin, Heng Fan, Yong Xu, and Haibin Ling. Swintrack: A simple and strong baseline for transformer tracking. In *NeurIPS*, 2022. 2, 3, 5, 8

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5, 8

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[43] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019. 2

[44] Alan Lukežic, Tomáš Vojír, Luka Cehovin Zajc, Jirí Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017. 2, 5

[45] 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*, 2021. 1, 5

[46] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 5, 6

[47] 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*, 2018. 5, 6

[48] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 5

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020. 3

[50] Dahu Shi, Xing Wei, Liangqi Li, Ye Ren, and Wenming Tan. End-to-end multi-person pose estimation with transformers. In *CVPR*, 2022. 3

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3

[52] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *CVPR*, 2018. 1

[53] 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*, 2017. 3

[54] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, 2020. 2, 5

[55] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *CVPR*, 2020. 5

[56] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, 2021. 1, 2, 5

[57] 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*, 2021. 5, 6

[58] Xing Wei, Yuanrui Kang, Jihao Yang, Yunfeng Qiu, Dahu Shi, Wenming Tan, and Yihong Gong. Scene-adaptive attention network for crowd counting. *arXiv:2112.15509*, 2021. 3

[59] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 2

[60] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021. 3

[61] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021. 1, 2, 3, 5, 8

[62] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018. 1

[63] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019. 2

[64] Botao Ye, Hong Chang, Bingpeng Ma, and Shiguang Shan. Joint feature learning and relation modeling for tracking: A one-stream framework. In *ECCV*, 2022. 1, 2, 3, 5, 7, 8

[65] Xiaodong Yu, Dahu Shi, Xing Wei, Ye Ren, Tingqun Ye, and Wenming Tan. Soit: Segmenting objects with instance-aware transformers. In *AAAI*, 2022. 3

[66] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *ICCV*, 2019. 1

[67] Zhipeng Zhang, Yihao Liu, Xiao Wang, Bing Li, and Weiming Hu. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, 2021. 5

[68] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020. 1, 2, 5

[69] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 2