Backbone is All Your Need: A Simplified Architecture for Visual Object Tracking

Boyu Chen[1,*], Peixia Li[1,*], Lei Bai[2,†], Lei Qiao[3], Qiuhong Shen[3], Bo Li[3], Weihao Gan[3], Wei Wu[3], Wanli Ouyang[2,1]

[1] The University of Sydney, SenseTime Computer Vision Group, Australia
[2] Shanghai AI Laboratory, Shanghai, China
[3] SenseTime, China
(*) equal contribution; (†) corresponding author
bailei@pjlab.org.cn

Abstract. Exploiting a general-purpose neural architecture to replace hand-wired designs or inductive biases has recently drawn extensive interest. However, existing tracking approaches rely on customized submodules and need prior knowledge for architecture selection, hindering the development of tracking in a more general system. This paper presents a Simplified Tracking architecture (SimTrack) by leveraging a transformer backbone for joint feature extraction and interaction. Unlike existing Siamese trackers, we serialize the input images and concatenate them directly before the one-branch backbone. Feature interaction in the backbone helps to remove well-designed interaction modules and produce a more efficient and effective framework. To reduce the information loss from down-sampling in vision transformers, we further propose a foveal window strategy, providing more diverse input patches with acceptable computational costs. Our SimTrack improves the baseline with 2.5%/2.6% AUC gains on LaSOT/TNL2K and gets results competitive with other specialized tracking algorithms without bells and whistles. The source codes are available at https://github.com/LPXTT/SimTrack.

1 Introduction

Visual Object Tracking (VOT) [11,52,7,29] aims to localize the specified target in a video, which is a fundamental yet challenging task in computer vision. Siamese network is a representative paradigm in visual object tracking [1,27,26,51], which usually consists of a Siamese backbone for feature extraction, an interactive head (e.g., naive correlation [1]) for modeling the relationship between the *exemplar* and *search*, and a predictor for generating the target localization. Recently, transformer [9,44,51] has been introduced as a more powerful interactive head to Siamese-based trackers for providing information interaction, as shown in Fig. 1(a), and pushes the accuracy to a new level.

While effective, these transformer heads are highly customized and meticulously designed, making it difficult to incorporate them into a more general system or generalize to a wide variety of intelligence tasks. On the other hand,

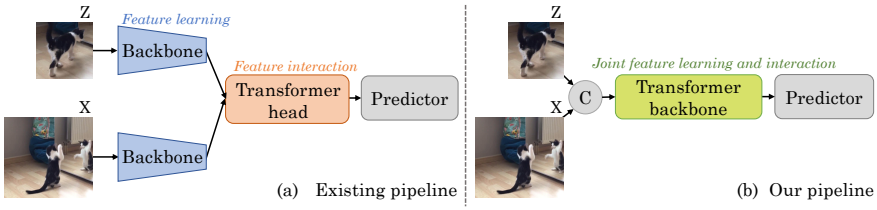

Fig. 1: The pipeline of existing transformer trackers (a) and ours (b). A transformer backbone is used to create a simple and generic framework for tracking.

transformers have recently shown an excellent capability to simplify frameworks for computer vision tasks, like object detection [8] and object segmentation [56]. Owning to the superior model capacity of transformers, the sub-modules and processes with task-specific prior knowledge can be removed by adequately leveraging transformers to a specific task. Producing a task-agnostic network can not only get a more simplified framework but also help the community move towards a general-purpose neural architecture, which is an appealing trend [23,58]. However, as observed in this paper, exploiting the transformer to produce a simple and generic framework is not investigated in existing VOT approaches.

With the observation above, this paper advocates a Simplified Tracking (SimTrack) paradigm by leveraging a transformer backbone for joint feature learning and interaction, shown as Fig.1(b). Specifically, we serialize the *exemplar* (Z) and *search* (X) images as multiple tokens at the beginning and send them together to our transformer backbone. Then, the *search* features from the transformer backbone are directly used for target localization through the predictor without any interaction module. Like existing backbones, our transformer backbone can also be pre-trained on other vision tasks, *e.g.* classification, providing stronger initialization for VOT. Moreover, our SimTrack brings multiple new benefits for visual object tracking. (1) Our SimTrack is a simpler and more generic framework with fewer sub-modules and less reliance on prior knowledge about the VOT task. The transformer backbone is a one-branch backbone instead of a Siamese network, consistent with the backbones used in many vision tasks, e.g., image classification [21,15,40,49], object detection [38], semantic segmentation [20,53], depth estimation [25,43], *etc.* (2) The attention mechanism in our transformer backbone facilitates a multi-level and more comprehensive interaction between the *exemplar* and *search* features. In this way, the backbone features for the *search* and *exemplar* image will be dependent on each other in every transformer block, resulting in a designated *examplar*(*search*)-sensitive rather than general *search*(*examplar*) feature, which is the hidden factor for the effectiveness of the seemingly simple transformer backbone. (3) Removing transformer head reduces training expenses. On one hand, the SimTrack can reach the same training loss or testing accuracy with only half training epochs as the baseline model because information interaction happens in a well-initialized transformer backbone instead of a randomly-initialized transformer head. On

the other hand, although adding information interaction in backbone will bring additional computation, the additional computation is generally smaller than that from a transformer head. (4) According to extensive experiments, SimTrack can get more accurate results with appropriate initialization than other transformer-based trackers using the same transformer as Siamese backbone.

While the transformer-based backbone is capable of achieving sufficient feature learning and interaction between the *exemplar* and *search* jointly, the down-sampling operation may cause unavoidable information loss for VOT, which is a localization task and requires more object visual details instead of only abstract/semantic visual concepts. To reduce the adverse effects of down-sampling, we further present a foveal window strategy inspired by fovea centralis. The fovea centralis is a small central region in the eyes, enabling human eyes to capture more useful information from the central part of vision area. In our paper, the centre area in the *exemplar* image contains more target-relevant information and needs more attention accordingly. Therefore, we add a foveal window at the central area to produce more diverse target patches, making the patch sampling frequencies around the image centre higher than those around the image border and improving the tracking performance.

In conclusion, our contributions are summarized as follows:

- We propose SimTrack, a Simplified Tracking architecture that feeds the serialized *exemplar* and *search* into a transformer backbone for joint feature learning and interaction. Compared with the existing Siamese tracking architecture, SimTrack only has the one-branch backbone and removes the existing interaction head, leading to a simpler framework with more powerful learning ability.
- We propose a foveal window strategy to remedy the information loss caused by the down-sampling in SimTrack, which helps the transformer backbone capture more details in important *exemplar* image areas.
- Extensive experiments on multiple datasets show the effectiveness of our method. Our SimTrack achieves state-of-the-art performances with 70.5% AUC on LaSOT [12], 55.6% AUC on TNL2K [48], 83.4% AUC on TrackingNet [35], 69.8% AO on GOT-10k [22] and 71.2% on UAV123 [34].

2 Related Work

2.1 Vision Transformer

Vaswani *et.al.* [42] originally proposed transformer and applied it in the machine translation task. The key character of the transformer is the self-attention mechanism which learns the dependencies of all input tokens and captures the global information in sequential data. Thanks to significantly more parallelization and competitive performance, transformer becomes a prevailing architecture in both language modeling [14,37] and vision community [15,41,6,5]. The first convolution-free vision transformer, ViT [15], splits input images into fixed-size

patches, which are converted to multiple 1D input tokens. All these tokens are concatenated with a class token and sent into a transformer encoder. After the encoder, the class token is used for image classification. Later, DeiT [41] introduces a distillation strategy to help transformers reduce the reliance on huge training data. For object detection, DETR [4] treats the task as a sequential prediction problem and achieves promising performance. To reduce the long training time of DETR, deformable DETR [57] replaces the global attention to adaptive local attention and speeds up the training process. Besides, transformer has also shown their powerful potential in other research topics like self-supervised learning [10,33], multi-module learning [36,24], *etc.*

2.2 Visual Object Tracking

Siamese networks is a widely-used two-branch architecture in a surge of tracking algorithms. Previous works [1,27,50,59,28,18,11,46,39] based on Siamese Networks [3] formulate VOT as a similarity matching problem and conduct the interaction through cross-correlation. Concretely, SiameseFC [1] utilize the response map from cross-correlation between the *exemplar* and *search* features for target localization. The highest score on the response map generally indicts the target position. In stead of directly getting the target position through the response map, SiamRPN [27] and the follow-ups [59,18,11,52] send the response map to Region Proposal Network (RPN) [38] to get a more accurate localization and scale estimation. Later, GAT [16] and AutoMatch [54] tried to replace the global cross-correlation with more effective structure to improve model performance. Recently, there have been several notable transformer trackers [44,9,51] which introduce the transformer to tracking framework for stronger information interaction and achieve compelling results.

All the above-mentioned works introduce interaction between the *exemplar* and *search* frames after the backbones. A recent work [17] adds multiple interaction modellers inside the backbone through hand-designed sub-modules. Our SimTrack also moves information interaction to the backbone but has the following fundamental differences. First, our SimTrack is a more generic and straightforward framework without using Siamese architecture or well-designed interaction modules, which are both used in [17] and all above Siamese-based methods. Second, our SimTrack utilizes pre-trained vision transformers for the interaction instead of training the interaction module from scratch. Third, the interaction between the *exemplar* and *search* exists in each block of our backbone. In contrast, the interaction modules are only added at the end of several blocks in [17]. Fourth, there is only information flow from the *exemplar* feature to the *search* feature in [17], while ours has bidirectional information interaction between the *exemplar* and *search* features.

3 Proposed Method

Our SimTrack consists of a transformer backbone and a predictor, as shown in Fig. 2 (b). The transformer backbone is used for feature extraction and in-

formation interaction between the *exemplar* and *search* features, guiding the network to learn a target-relevant *search* feature. After passing the backbone, the output features corresponding to the *search* area are sent to a corner predictor for target localization. For better understanding, we will first introduce our baseline model in Sec. 3.1, which replaces the CNN backbone of STARK-S [51] with a transformer backbone, and then show details of our SimTrack in Sec. 3.2 and the foveal window strategy for improving SimTrack in Sec. 3.3.

3.1 Baseline Model

STARK-S has no extra post-processing during inference, which is consistent with our initial purpose to simplify the tracking framework. We replace the backbone of STARK-S [51] from Res50 [21] to ViT [15] to get our baseline model STARK-SV. Like other transformer-based trackers, the pipeline of STARK-SV is shown in Fig. 1 (a). Given a video, we treat the first frame with ground truth target box as *exemplar* frame. According to the target box, we crop an *exemplar* $\mathbf{Z} \in \mathbb{R}^{H_z \times W_z \times 3}$ from the first frame, where (H_z, W_z) is the input resolution of $\mathbf{Z}$. All following frames $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times 3}$ are the *search* frames.

Image serialization. The two input images are serialized into input sequences before the backbone. Specifically, similar to current vision transformers [15,41], we reshape the images $\mathbf{Z} \in \mathbb{R}^{H_z \times W_z \times 3}$ and $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times 3}$ into two sequences of flattened 2D patches $\mathbf{Z_p} \in \mathbb{R}^{N_z \times (P^2 \cdot 3)}$ and $\mathbf{X_p} \in \mathbb{R}^{N_x \times (P^2 \cdot 3)}$, where (P, P) is the patch resolution, $N_z = H_z W_z / P^2$ and $N_x = H_x W_x / P^2$ are patch number of the *exemplar* and *search* images. The 2D patches are mapped to 1D tokens with C dimensions through a linear projection. After adding the 1D tokens with positional embedding [42], we get the input sequences of the backbone, including the *exemplar* sequence $e^0 \in \mathbb{R}^{N_z \times C}$ and the *search* sequence $s^0 \in \mathbb{R}^{N_x \times C}$.

Feature extraction with backbone. The transformer backbone consists of L layers. We utilize e^l and s^l to represent the input *exemplar* and *search* sequences of the $(l+1)_{th}$ layer, $l = 0, ..., L-1$. The forward process of the *exemplar* feature in one layer can be written as:

$$\begin{aligned} e^* &= e^l + Att(LN(e^l)), \\ e^{l+1} &= e^* + FFN(LN(e^*)), \end{aligned} \tag{1}$$

where FFN is a feed forward network, LN denotes Layernorm and Att is self-attention module [42] (we remove LN in the following functions for simplify),

$$Att(e^l) = softmax\left(\frac{(e^l W_Q)(e^l W_K)^T}{\sqrt{d}}\right)\left(e^l W_V\right), \tag{2}$$

where $1/\sqrt{d}$ is the scaling factor, $W_Q \in \mathbb{R}^{C \times D}$, $W_K \in \mathbb{R}^{C \times D}$, $W_V \in \mathbb{R}^{C \times D}$ are project metrics to convert input sequence to *query*, *key* and *value*. Generally, multi-head self-attention [42] is adopted to replace self-attention in Equ.(1). For

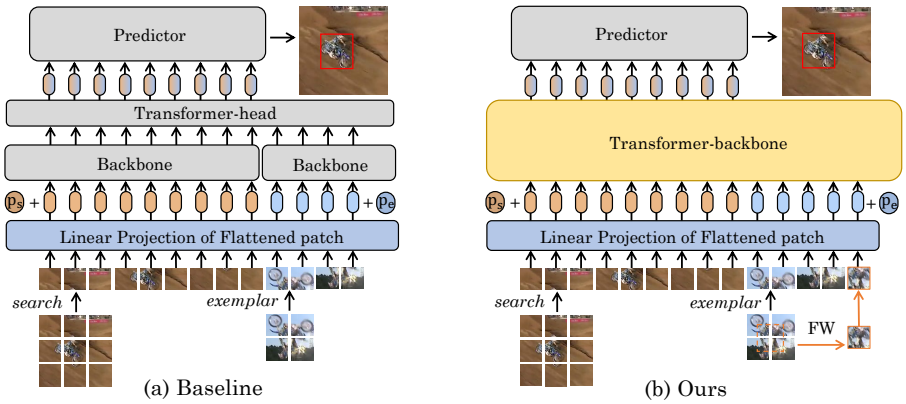

Fig. 2: The pipeline of the baseline model (a) and our proposed SimTrack (b). 'FW' in (b) denotes foveal window, p_s and p_e are position embedding of the *search* and *exemplar* tokens. In (b), a transformer backbone is utilized to replace the Siamese backbone and transformer head in (a). Both *exemplar* and *search* images in (b) are serialized into input sequences, which are sent to the transformer backbone for joint feature extraction and interaction. Finally, the target-relevant search feature is used for target localization through a predictor.

simplicity and better understanding, we use the self-attention module in our descriptions. As we can see, the feature extraction of e^l only considers *exemplar* information. The feed forward process of s^l is the same as e^l. After passing the input into the backbone, we get the output *exemplar* sequence e^L and the output *search* sequence s^L.

Feature interaction with transformer head. The features $e^L \in \mathbb{R}^{N_z \times D}$ and $s^L \in \mathbb{R}^{N_x \times D}$ interact with each other in the transformer head. We refer readers to STARK-S [51] for more details of the transformer head in our baseline models.

Target localization with predictor. After transformer head, we get a target-relevant *search* feature $s^{L*} \in \mathbb{R}^{N_x \times D^*}$, which is reshaped to $\frac{H_x}{s} \times \frac{W_x}{s} \times D^*$ and sent to a corner predictor. The corner predictor outputs two probability maps for the top-left and bottom-right corners of the target box.

During offline training, a pair of images within a pre-defined frame range in a video are randomly selected to serve as the *exemplar* and *search* frame. After getting the predicted box b_i on the *search* frame, the whole network is trained through ℓ_1 loss and generalized IoU loss [4],

$$L = \lambda_{iou} L_{iou}(b_i, b_i^*) + \lambda_{L_1} L_1(b_i, b_i^*), \tag{3}$$

where b_i^* is the ground truth box, λ_{iou} and λ_{L_1} are loss weights, L_{iou} is generalized IoU loss and L_1 is the ℓ_1 loss.

3.2 Simplified Tracking Framework

Our key idea is replacing the Siamese backbone and transformer head in the baseline model with a unified transformer backbone, as shown in Fig. 2 (b). For STARK-S, the function of the backbone is to provide a strong feature extraction. The transformer head is responsible for information interaction between the *exemplar* and *search* features. In our SimTrack, only a transformer backbone is needed for joint feature and interaction learning. In the following, we show how to apply vision transformer as a powerful backbone to VOT successfully and create a more simplified framework. The input of our transformer backbone is also a pair of images, the *exemplar* image $\mathbf{Z} \in \mathbb{R}^{H_z \times W_z \times 3}$ and the *search* image $\mathbf{X} \in \mathbb{R}^{H_x \times W_x \times 3}$. Similarly, we first serialize the two images to input sequences $e^0 \in \mathbb{R}^{N_z \times C}$ and $s^0 \in \mathbb{R}^{N_x \times C}$ as mentioned above.

Joint feature extraction and interaction with transformer backbone. Different from the baseline model, we directly concatenate e^0 and s^0 along the first dimension and send them to the transformer backbone together. The feed forward process of $(l+1)_{th}$ layer is:

$$\begin{aligned} \begin{bmatrix} e^* \\ s^* \end{bmatrix} &= \begin{bmatrix} e^l \\ s^l \end{bmatrix} + Att\left(\begin{bmatrix} e^l \\ s^l \end{bmatrix}\right), \\ \begin{bmatrix} e^{l+1} \\ s^{l+1} \end{bmatrix} &= \begin{bmatrix} e^* \\ s^* \end{bmatrix} + FFN\left(\begin{bmatrix} e^* \\ s^* \end{bmatrix}\right). \end{aligned} \tag{4}$$

The symbol of layer normalization is removed in Equ.(4) for simplify. The main difference between Equ.(1) and Equ.(4) is the computation in $Att(.)$,

$$Att(\begin{bmatrix} e^l \\ s^l \end{bmatrix}) = softmax\left(\begin{bmatrix} a(e^l, e^l),\, a(e^l, s^l) \\ a(s^l, e^l),\, a(s^l, s^l) \end{bmatrix}\right)\left(\begin{bmatrix} e^l W_V \\ s^l W_V \end{bmatrix}\right), \tag{5}$$

where $a(x, y) = (xW_Q)(yW_K)^T/\sqrt{d}$. After converting Equ.(5), the *exemplar* attention $Att(e^l)$ and the *search* attention $Att(s^l)$ are,

$$\begin{aligned} Att(e^l) &= softmax\left(\left[a(e^l, e^l), a(e^l, s^l)\right]\right)\left[e^l W_V, s^l W_V\right]^T, \\ Att(s^l) &= softmax\left(\left[a(s^l, e^l), a(s^l, s^l)\right]\right)\left[e^l W_V, s^l W_V\right]^T. \end{aligned} \tag{6}$$

In the baseline model, the feature extraction of the *exemplar* and *search* features are independent with each other as shown in Equ.(2). While, in our transformer backbone, the feature learning of *exemplar* and *search* images influence each other through $a(e^l, s^l)$ and $a(s^l, e^l)$ in Equ.(6). $Att(e^l)$ contains information from s^l and vice verse. The information interaction between the *exemplar* and *search* features exists in every layer of our transformer backbone, so there is no need to add additional interaction module after the backbone. We directly send the output *search* feature s^L to the predictor for target localization.

Distinguishable position embedding. It is a general paradigm to seamlessly transfer networks pre-trained from the classification task to provide a

stronger initialization for VOT. In our method, we also initialize our transformer backbone with pre-trained parameters. For the *search* image, the input size (224×224) is the same with that in general vision transformers [15,41], so the pre-trained position embedding p_0 can be directly used for the *search* image ($p_s = p_0$). However, the *exemplar* image is smaller than the *search* image, so the pre-trained position embedding can not fit well for the *exemplar* image. Besides, using the same pre-trained position embedding for both images provides the backbone with no information to distinguish the two images. To solve the issue, we add a learnable position embedding $p_e \in \mathbb{R}^{N_z \times D}$ to the *exemplar* feature, which is calculated by the spatial position (i, j) of the patch and the ratio R_{ij} of the target area in this patch (as depicted in Fig. 3 (b)),

$$p_e = FCs(i, j, R_{ij}), \tag{7}$$

where p_e denotes the position embedding of the *exemplar* feature, FCs are two fully connected layers. After obtaining the position embedding p_e and p_s, we add them to the embedding vectors. The resulting sequences of embedding vectors serve as inputs to the transformer backbone.

3.3 Foveal Window Strategy

The *exemplar* image contains the target in the center and a small amount of background around the target. The down-sampling process may divide the important target region into different parts. To provide the transformer backbone with more detailed target information, we further propose a foveal window strategy on the *exemplar* image to produce more diverse target patches with acceptable computational costs. As shown in the second row of Fig. 3(a), we crop a smaller region $\mathbf{Z}^* \in \mathbb{R}^{H_z^* \times W_z^* \times 3}$ in the center of the *exemplar* image and serialize $\mathbf{Z}^*$ into image patches $\mathbf{Z}_\mathbf{p}^* \in \mathbb{R}^{N_z^* \times (P^2 \cdot 3)}$, where $N_x^* = H_x^* W_x^* / P^2$. The partitioning lines on $\mathbf{Z}^*$ are located in the center of those on the *exemplar* image $\mathbf{Z}$, so as to ensure that the foveal patches $\mathbf{Z}_\mathbf{p}^*$ contain different target information with the original patches $\mathbf{Z}_\mathbf{p}$. After getting the foveal patches $\mathbf{Z}_\mathbf{p}^*$, we calculate their position embedding according to Equ.(7). Then, we map $\mathbf{Z}_\mathbf{p}^*$ with the same linear projection as $\mathbf{Z}_\mathbf{p}$ and add the mapped feature with the position embedding to get the foveal sequence

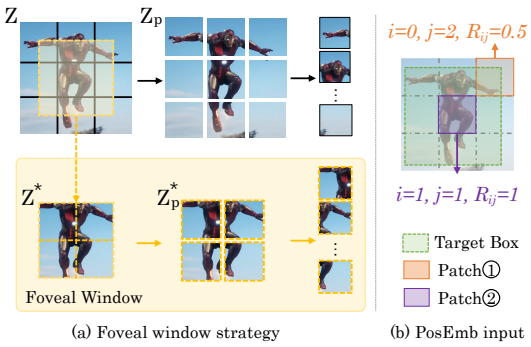

Fig. 3: (a) the foveal window strategy and (b) getting the inputs of FCs in Equ.(7).

e^{0*}. Finally, the input of transformer backbone includes the *search* sequence s^0, the *exemplar* sequence e^0 and the foveal sequence e^{0*}. The *exemplar* image is small in VOT, so the token number in e^0 and e^{0*} are modest as well.

4 Experiments

4.1 Implementation Details

Model. We evaluate our method on vision transformer [36] and produce three variants of SimTrack: Sim-B/32, Sim-B/16, and Sim-L/14 with the ViT base, base, and large model [15] as the backbone, respectively, where input images are split into 32×32, 16×16 and 14×14 patches, correspondingly. All parameters in the backbone are initialized with pre-trained parameters from the vision branch of CLIP [36]. For better comparison with other trackers, we add another variant Sim-B/16* with fewer FLOPs than Sim-B/16. In Sim-B/16*, we remove the last four layers in the transformer backbone to reduce computation costs. The predictor is exactly the same as that in STARK-S [51].

Training. Our SimTrack is implemented with Python 3.6.9 on PyTorch 1.8.1. All experiments are conducted on a server with 8 16GB V100 GPUs. The same as STARK-S, we train our models with training-splits of LaSOT [12], GOT-10K [22], COCO2017 [30], and TrackingNet [35] for experiments on all testing datasets except for GOT-10k_Test. For GOT-10k_Test, we follow the official requirements and only use the *train* set of GOT-10k for model training. In Sim-B/32, we set the input sizes of *exemplar* and *search* images as 128×128 and 320×320, corresponding to 2^2 and 5^2 times of the target bounding box, because the larger stride 32 makes the output features having a smaller size. Too small output size has a negative effect on target localization. In Sim-B/16, the input sizes are 112×112 and 224×224, corresponding to 2^2 and 4^2 times of the target bounding box. For Sim-L/14, the *exemplar* input size is reduced to 84×84 (1.5^2 times of target bounding box) to reduce computation costs. Without the special declaration, all other experiments use the same input sizes as Sim-B/16. The size of the cropped image for the foveal window is 64×64. All other training details are the same with STARK-S [51] and shown in the supplementary materials.

Inference. Like STARK-S [51], there is no extra post-processing for all SimTrack models. The inference pipeline only consists of a forward pass and coordinate transformation process. The input sizes of *exemplar* and *search* images are consistent with those during offline training. Our Sim-B/16 can run in real-time at more than 40 fps.

4.2 State-of-the-art Comparisons

We compare our SimTrack with other trackers on five datasets, including LaSOT [12], TNL2K [48], TrackingNet [35], UAV123 [34] and GOT-10k [22].

LaSOT is a large-scale dataset with 1400 long videos in total. The *test* set of LaSOT [12] consists of 280 sequences. Table 1 shows the AUC and normalized precision scores (P_{norm}) of all compared trackers. Our SimTrack can get a

Methods	Net	Size	FLOPs	LaSOT		TNL2K		TrackingNet	
				AUC	P_{norm}	AUC	P	AUC	P
SiamFC [1]	AlexNet	255	4.9G	33.6	42.0	29.5	28.6	57.1	66.3
ATOM [13] ◇	ResNet18	288	3.0G	51.5	57.6	40.1	39.2	70.3	64.8
DiMP [2] ◇	ResNet50	288	5.4G	56.9	65.0	44.7	43.4	74.0	68.7
SiamRPN++ [26]	ResNet50	255	7.8G	49.6	56.9	41.3	41.2	73.3	69.4
SiamFC++ [50]	GoogleNet	303	15.8G	54.4	56.9	38.6	36.9	75.4	70.5
Ocean [55] ◇	ResNet50	255	7.8G	56.0	65.0	38.4	37.7	70.3	68.8
SiamBAN [11]	ResNet50	255	12.1G	51.4	52.1	41.0	41.7	-	-
SiamAtt [52]	ResNet50	255	7.8G	56.0	64.8	-	-	75.2	-
TransT [9]	ResNet50	256	29.3G	64.9	73.8	50.7	51.7	81.4	80.3
TrDiMP [45] ◇	ResNet50	352	18.2G	63.9	-	-	-	78.4	73.1
KeepTrack [32] ◇	ResNet50	464	28.7G	67.1	77.2	-	-	-	-
AutoMatch [54]	ResNet50	-	-	58.3	-	47.2	43.5	76.0	72.6
TransInMo* [17]	ResNet50	255	16.9G	65.7	76.0	52.0	52.7	-	-
STARK-S [51]	ResNet50	320	15.6G	65.8	-	-	-	80.3	-
STARK-ST [51] ◇	ResNet101	320	28.0G	67.1	77.0	-	-	82.0	86.9
Sim-B/32	ViT-B/32	320	11.5G	66.2	76.1	51.1	48.1	79.1	83.9
Sim-B/16*	ViT-B/16*	224	14.7G	68.7	77.5	53.7	52.6	81.5	86.0
Sim-B/16	ViT-B/16	224	25.0G	69.3	78.5	54.8	53.8	82.3	86.5
Sim-L/14	ViT-L/14	224	95.4G	70.5	79.7	55.6	55.7	83.4	87.4

Table 1: Performance comparisons with state-of-the-art trackers on the *test* set of LaSOT [12], TNL2K [48] and TrackingNet [35]. 'Size' means the size of *search* image, 'FLOPs' shows the computation costs of backbone and transformer head. For methods without transformer head, 'FLOPs' shows the computation costs from the backbone. AUC, P_{norm} and P are AUC, normalized precision and precision. Sim-B/16* denotes removing the last four layers of the transformer-backbone in Sim-B/16 to reduce FLOPs. Trackers shown with ◇ have online update modules. Red, green and blue fonts indicate the top-3 methods.

competitive or even better performance compared with state-of-the-art trackers. Our Sim-B/16* outperforms all compared trackers with a simpler framework and lower computation costs. Our Sim-B/16 achieves a new state-of-the-art result, 69.3% AUC score and 78.5% normalized precision score, with acceptable computation costs. After using the larger model ViT-L/14, our Sim-L/14 can get a much higher performance, 70.5% AUC score and 79.7% normalized precision score. We are the first to exploit such a large model and demonstrate its effectiveness in visual object tracking.

TNL2K is a recently published datasets which composes of 3000 sequences. We evaluate our SimTrack on the *test* set with 700 videos. From Tab. 1, SimTrack performs the best among all compared trackers. The model with ViT-B/16 exceeds 2.8 AUC points than the highest AUC score (52.0%) of all compared trackers. Leveraging a larger model can further improve the AUC score to 55.6%.

TrackingNet is another large-scale dataset consists of 511 videos in the *test* set. The *test* dataset is not publicly available, so results should be submitted to

	SiamFC [1]	SiamRPN [27]	SiamFC++ [50]	DiMP [2]	TrDiMP [45]	TransT [9]	**Ours** ViT-B/16	**Ours** ViT-L/14
AUC↑	48.5	55.7	63.1	65.4	67.5	68.1	69.8	71.2
Pre↑	64.8	71.0	76.9	85.6	87.2	87.6	89.6	91.6

Table 2: Performance comparisons on UAV123 [34] dataset. Red, green and blue fonts indicate the top-3 methods.

	SiamFC [1]	SiamRPN [27]	SiamFC++ [50]	DiMP [2]	TrDiMP [45]	STARK-s [51]	**Ours** ViT-B/16	**Ours** ViT-L/14
AO↑	34.8	46.3	59.5	61.1	67.1	67.2	68.6	69.8
$SR_{0.5}$ ↑	35.3	40.4	69.5	71.7	77.7	76.1	78.9	78.8
$SR_{0.75}$ ↑	9.8	14.4	47.9	49.2	58.3	61.2	62.4	66.0

Table 3: Experimental results on GOT-10k_Test [22] dataset.

an online server for performance evaluation. Compared with the other trackers with complicated interaction modules, our SimTrack is a more simple and generic framework, yet achieves competitive performance. By leveraging a larger model, Sim-L/14 outperforms all compared trackers including those with online update.

UAV123 provides 123 aerial videos captured from a UAV platform. In Table 2, two versions of our method both achieve better AUC scores (69.8 and 71.2) than the highest AUC score (68.1) of all compared algorithms.

GOT-10k requires training trackers with only the *train* subset and testing models through an evaluation server. We follow this policy for all experiments on GOT-10k. As shown in Table 3, our tracker with ViT-B/16 obtains the best performance. When leveraging a larger model ViT-L/14, our model can further improve the performance to 69.8 AUC score.

4.3 Ablation Study and Analysis

Simplified Framework *vs.* **STARK-SV.** To remove concerns about backbone, we compare our method with the baseline tracker STARK-SV [51] using the same backbone architecture. In Table 4, our design can consistently get significant performance gains with similar or even fewer computation costs. Our three variations with ViT-B/32, ViT-B/16 and ViT-L/14 as backbone outperforms STARK-SV for 3.7/3.1, 2.5/2.6 and 1.3/1.6 AUC points on LaSOT/TNL2K dataset, respectively, demonstrating the effectiveness and efficiency of our method.

Training Loss & **Accuracy.** In Fig. 6, we show the training losses and AUC scores of the baseline model STARK-SV and our method 'Ours' on the LaSOT dataset. Both the two trackers utilize ViT-B/16 as the backbone. We can see that 'Ours' uses fewer training epochs to get the same training loss with STARK-SV. When training models for the same epochs, 'Ours' can get lower training losses than STARK-SV. In terms of testing accuracy, training our model for

	Backbone	FLOPs	LaSOT			TNL2K	
			AUC↑	P_{norm}↑	P↑	AUC↑	P↑
STARK-SV	ViT-B/32	13.3G	62.5	72.1	64.0	48.0	44.0
Ours	ViT-B/32	11.5G	66.2 (+3.7)	76.1 (+4.0)	68.8 (+4.8)	51.1 (+3.1)	48.1 (+4.1)
STARK-SV	ViT-B/16	25.6G	66.8	75.7	70.6	52.2	51.1
Ours	ViT-B/16	23.4G	69.3 (+2.5)	78.5 (+2.8)	74.0 (+3.4)	54.8 (+2.6)	53.8 (+2.7)
STARK-SV	ViT-L/14	95.6G	69.2	78.2	74.3	54.0	54.1
Ours	ViT-L/14	95.4G	70.5 (+1.3)	79.7 (+1.5)	76.2 (+1.9)	55.6 (+1.6)	55.7 (+1.6)

Table 4: Ablation study about our simplified framework and the baseline model STARK-S [51]. 'FLOPs' shows computation costs of different methods, AUC, P_{norm} and P respectively denote AUC, normalized precision and precision.

#Num		①	②	③	④	⑤
Pretrain		DeiT	Moco	SLIP	CLIP	MAE
LaSOT	AUC	66.9	66.4	67.6	69.3	70.3
	Prec	70.3	69.4	71.0	74.0	75.5
TNL2K	AUC	51.9	51.9	53.4	54.8	55.7
	Prec	49.6	49.4	51.8	53.8	55.8

Table 5: The AUC/Pre scores of SimTrack (with ViT-B/16 as backbone) when using different pre-training weights.

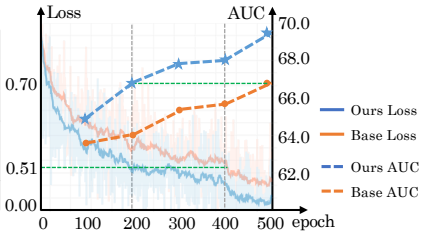

Table 6: The training loss and AUC (on LaSOT) in the Y-axis for different training epochs (X-axis).

200 epochs is enough to get the same AUC score (66.8% *vs.* 66.8%) with the baseline model trained for 500 epochs. We think the main reason is 'Ours' does not have a randomly initialized transformer head. The transformer head without pre-training needs more training epochs to get a good performance.

Results with Other Transformer Backbones. We evaluate our framework with Swin Transformer [31] and Pyramid Vision Transformer (PVT) [47]. For Swin Transformer, we made a necessary adaption, considering the shifted window strategy. We remove the $a(e^l, s^l)$ and $s^l W_V$ in the first function of Eq.(6), which has less influence according to our experiments (from 69.3% to 69.1% AUC score on LaSOT for SimTrack-ViT). The attention of each *search* token is calculated with the tokens inside the local window and those from *exemplar* features. During attention calculation, the *exemplar* features are pooled to the size of the local window. For PVT, we reduce the reduction ratio of SRA module for the *exemplar* by half, to keep a reasonable *exemplar* size. In the Table 7, SimTrack with PVT-Medium is denoted as PVT-M and SimTrack with Swin-Base is denoted as Swin-B. PVT-M gets comparable AUC scores with fewer FLOPs, and Swin-B has higher AUC scores with similar FLOPs to STARK-S on both datasets, demonstrating the good generalization of our SimTrack.

Different Pre-training. We evaluate our SimTrack when using ViT-B/16 as backbone and initializing the backbone with parameters pre-trained with several

	DiMP	TrDiMP	TransT	STARK-S	**PVT-M**	**Swin-B**
FLOPs	5.4G	18.2G	29.3G	15.6G	**8.9G**	**15.0G**
LaSOT	56.9	63.9	64.9	65.8	**66.6**	**68.3**
UAV123	65.4	67.5	68.1	68.2	**68.5**	**69.4**

Table 7: The AUC scores and FLOPs of SimTrack using PVT and Swin-Transformer as backbone on LaSOT and UAV123 dataset.

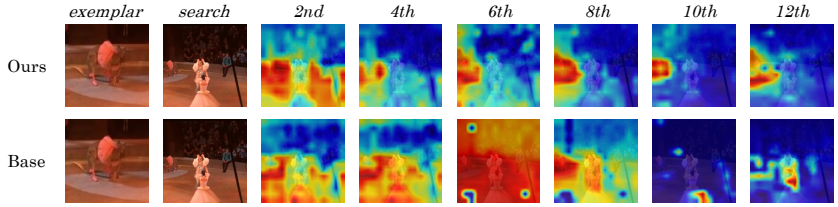

Fig. 4: The images in different columns are the *exemplar* image, *search* image, target-relevant attention maps from the $2nd, 4th, 6th, 8th, 10th, 12th(last)$ layer of the transformer backbone. Details can be found in supplementary materials.

recent methods, including DeiT [41], MOCO-V3 [10], SLIP [33], CLIP [36], and MAE [19]. From Table 5, all of these versions achieve competitive performance with state-of-the-art trackers on the two datasets. However, the pre-trained parameters from MAE show the best performance, suggesting that appropriate parameter initialization is helpful to the training of SimTrack.

Component-wise Analysis. To prove the efficiency of our method, we perform a component-wise analysis on the TNL2K [48] benchmark, as shown in Table 8. The 'Base' means STARK-SV with ViT-B/16, which obtains an AUC score of 52.2. In ②, '+Sim' indicates using our SimTrack framework without adding the distinguishable position embedding or foveal window strategy. It brings significant gains, *i.e.* 1.3/1.4 point in terms of AUC/Pre score, and verifies the effectiveness of our framework. Adding our position embedding helps model performs slightly better (③ *vs.* ②). Furthermore, the foveal window strategy brings an improvement of 0.8 point on AUC score in ④. This shows using more detailed target patches at the beginning contributes to improving accuracy.

Decoder Number. We analyze the necessity of introducing transformer decoders in our SimTrack. Specifically, we add a transformer decoder at the end of our backbone for further information interaction. In the decoder, the *search* features from the backbone are used to get *query* values. The *exemplar* features are adopt to calculate *key* and *value*. Through changing the layer number of the decoder from 0 to 6, the performance changes less. This shows another information interaction module is unnecessary in our framework, because our transformer backbone can provide enough information interaction between the *search* and *exemplar* features.

#Num	Com	TNL2K↑
①	Base	52.2/51.1
②	+Sim	53.5/52.5
③	+PosEm	54.0/53.1
④	+FW	54.8/53.8

Table 8: Component-wise analysis. AUC/Pre scores are reported respectively. The results demonstrate that each component is important in our framework.

#Num	Dec	TNL2K↑
①	0	54.8/53.8
②	1	54.8/54.2
③	3	54.6/54.0
④	6	54.7/54.3

Table 9: The influence of introducing decoders in SimTrack. With sufficient interaction in the transformer backbone, decoder becomes redundant for SimTrack.

#Num	Ratio	TNL2K↑
①	100%	54.8/53.8
②	50%	52.3/50.4
③	25%	49.8/46.2

Table 10: Analysis of information interaction ratio in backbone. ① is ours with interaction in 100% blocks. ② and ③ reduce the number of interaction blocks to 50% and 25%.

Dense or Sparse Information Interaction. The information interaction between the *exemplar* and *search* features exist in all twelve blocks in our SimB/16, shown as ① in Table 10. In ②, we only enable the interaction in the $2nd, 4th, 6th, 8th, 10th$ and $12th$ block, removing half of the interaction in ①. As we can see, using less information interaction leads to 2.5 points AUC drop. When we further reduce half interaction in ②, the AUC score drops another 2.5 points in ③. The experiments show that comprehension information interaction helps to improve the tracking performance in SimTrack.

Visualization. Fig. 4 shows the target-relevant area in the search region for different layers. Our architecture can gradually and quickly focus on the designated target and keep following the target in the following layers. The visualization maps show that the Siamese backbone in 'base' tends to learn general-object sensitive features instead of designated-target sensitive features and no information interaction hinders the backbone from 'sensing' the target during feature learning. By contrast, 'Ours' can produce designated-target sensitive features thanks to the information interaction from the first block to the last block.

5 Conclusions

This work presents SimTrack, a simple yet effective framework for visual object tracking. By leveraging a transformer backbone for joint feature learning and information interaction, our approach streamlines the tracking pipeline and eliminates most of the specialization in current tracking methods. While it obtains compelling results against well-established baselines on five tracking benchmarks, both architecture and training techniques can be optimized for further performance improvements

Acknowledgement. This work was supported by the Australian Research Council Grant DP200103223, Australian Medical Research Future Fund MRFAI000085, CRC-P Smart Material Recovery Facility (SMRF) – Curby Soft Plastics, and CRC-P ARIA - Bionic Visual-Spatial Prosthesis for the Blind.

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV (2016)
2. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: ICCV (2019)
3. Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., Shah, R.: Signature verification using a siamese time delay neural network. In: NeurIPS. pp. 737–744 (1993)
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) ECCV (2020)
5. Chen, B., Li, P., Li, B., Li, C., Bai, L., Lin, C., Sun, M., Yan, J., Ouyang, W.: Psvit: Better vision transformer via token pooling and attention sharing. arXiv preprint arXiv:2108.03428 (2021)
6. Chen, B., Li, P., Li, C., Li, B., Bai, L., Lin, C., Sun, M., Yan, J., Ouyang, W.: Glit: Neural architecture search for global and local image transformer. In: ICCV (2021)
7. Chen, B., Wang, D., Li, P., Wang, S., Lu, H.: Real-time'actor-critic'tracking. In: ECCV (2018)
8. Chen, T., Saxena, S., Li, L., Fleet, D.J., Hinton, G.: Pix2seq: A language modeling framework for object detection. arXiv preprint arXiv:2109.10852 (2021)
9. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: CVPR (2021)
10. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021)
11. Chen, Z., Zhong, B., Li, G., Zhang, S., Ji, R.: Siamese box adaptive network for visual tracking. In: CVPR (2020)
12. Choi, J., Kwon, J., Lee, K.M.: Deep meta learning for real-time visual tracking based on target-specific feature space. CoRR **abs/1712.09153** (2017)
13. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ATOM: accurate tracking by overlap maximization. In: CVPR (2019)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
15. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
16. Guo, D., Shao, Y., Cui, Y., Wang, Z., Zhang, L., Shen, C.: Graph attention tracking. In: CVPR (2021)
17. Guo, M., Zhang, Z., Fan, H., Jing, L., Lyu, Y., Li, B., Hu, W.: Learning target-aware representation for visual tracking via informative interactions. arXiv preprint arXiv:2201.02526 (2022)
18. Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., Wang, S.: Learning dynamic siamese network for visual object tracking. In: Proceedings of the IEEE international conference on computer vision. pp. 1763–1771 (2017)
19. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022)
20. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)

21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
22. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. CoRR **abs/1810.11981** (2018)
23. Jaegle, A., Borgeaud, S., Alayrac, J.B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al.: Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795 (2021)
24. Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I., Carion, N.: Mdetr-modulated detection for end-to-end multi-modal understanding. In: ICCV (2021)
25. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth international conference on 3D vision (3DV). pp. 239–248. IEEE (2016)
26. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019)
27. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR (2018)
28. Li, P., Chen, B., Ouyang, W., Wang, D., Yang, X., Lu, H.: Gradnet: Gradient-guided network for visual object tracking. In: ICCV (2019)
29. Li, P., Wang, D., Wang, L., Lu, H.: Deep visual tracking: Review and experimental comparison. Pattern Recognition **76**, 323–338 (2018)
30. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV (2014)
31. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
32. Mayer, C., Danelljan, M., Paudel, D.P., Van Gool, L.: Learning target candidate association to keep track of what not to track. In: ICCV (2021)
33. Mu, N., Kirillov, A., Wagner, D., Xie, S.: Slip: Self-supervision meets language-image pre-training. arXiv preprint arXiv:2112.12750 (2021)
34. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for UAV tracking. In: ECCV (2016)
35. Müller, M., Bibi, A., Giancola, S., Al-Subaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018)
36. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
37. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
38. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
39. Shen, Q., Qiao, L., Guo, J., Li, P., Li, X., Li, B., Feng, W., Gan, W., Wu, W., Ouyang, W.: Unsupervised learning of accurate siamese tracking. In: CVPR (2022)
40. Tang, S., Chen, D., Bai, L., Liu, K., Ge, Y., Ouyang, W.: Mutual crf-gnn for few-shot learning. In: CVPR (2021)
41. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
42. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
43. Wang, L., Zhang, J., Wang, O., Lin, Z., Lu, H.: Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. In: CVPR (2020)

44. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: CVPR (2021)
45. Wang, N., Zhou, W., Wang, J., Li, H.: Transformer meets tracker: Exploiting temporal context for robust visual tracking. In: ICCV (2021)
46. Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H.S.: Fast online object tracking and segmentation: A unifying approach. In: CVPR (2019)
47. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021)
48. Wang, X., Shu, X., Zhang, Z., Jiang, B., Wang, Y., Tian, Y., Wu, F.: Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In: CVPR (2021)
49. Wang, Y., Tang, S., Zhu, F., Bai, L., Zhao, R., Qi, D., Ouyang, W.: Revisiting the transferability of supervised pretraining: an mlp perspective. In: CVPR (2022)
50. Xu, Y., Wang, Z., Li, Z., Ye, Y., Yu, G.: Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In: AAAI (2020)
51. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. arXiv preprint arXiv:2103.17154 (2021)
52. Yu, Y., Xiong, Y., Huang, W., Scott, M.R.: Deformable siamese attention networks for visual object tracking. In: CVPR (2020)
53. Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., Yang, J.: Joint task-recursive learning for semantic segmentation and depth estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 235–251 (2018)
54. Zhang, Z., Liu, Y., Wang, X., Li, B., Hu, W.: Learn to match: Automatic matching network design for visual tracking. In: ICCV (2021)
55. Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W.: Ocean: Object-aware anchor-free tracking. In: ECCV (2020)
56. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR (2021)
57. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
58. Zhu, X., Zhu, J., Li, H., Wu, X., Wang, X., Li, H., Wang, X., Dai, J.: Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. arXiv preprint arXiv:2112.01522 (2021)
59. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: ECCV (2018)