Siamese Transformer Pyramid Networks for Real-Time UAV Tracking

Daitao Xing
New York University , USA
daitao.xing@nyu.edu

Nikolaos Evangeliou
New York University Abu Dhabi, UAE
nikolaos.evangeliou@nyu.edu

Athanasios Tsoukalas
New York University Abu Dhabi, UAE
athanasios.tsoukalas@nyu.edu

Anthony Tzes
New York University Abu Dhabi, UAE
anthony.tzes@nyu.edu

Abstract

Recent object tracking methods depend upon deep networks or convoluted architectures. Most of those trackers can hardly meet real-time processing requirements on mobile platforms with limited computing resources. In this work, we introduce the Siamese Transformer Pyramid Network (SiamTPN), which inherits the advantages from both CNN and Transformer architectures. Specifically, we exploit the inherent feature pyramid of a lightweight network (ShuffleNetV2) and reinforce it with a Transformer to construct a robust target-specific appearance model. A centralized architecture with lateral cross attention is developed for building augmented high-level feature maps. To avoid the computation and memory intensity while fusing pyramid representations with the Transformer, we further introduce the pooling attention module, which significantly reduces memory and time complexity while improving the robustness. Comprehensive experiments on both aerial and prevalent tracking benchmarks achieve competitive results while operating at high speed, demonstrating the effectiveness of SiamTPN. Moreover, our fastest variant tracker operates over 30 Hz on a single CPU-core and obtaining an AUC score of 58.1% *on the LaSOT dataset. Source codes are available at https://github.com/RISC-NYUAD/SiamTPNTracker*

1. Introduction

Unmanned Aerial Vehicle (UAV) tracking has drawn increasing attention in recent years given its enormous potential in diverse fields such as path planning [25], visual surveillance [43], and border security [44]. While extensive advancements have been made towards powerful visual object tracking methods, the problem of real-time tracking has been overlooked. Moreover, the inherently limited power resources on lower performance compact devices further constraint the development of UAV tracking.

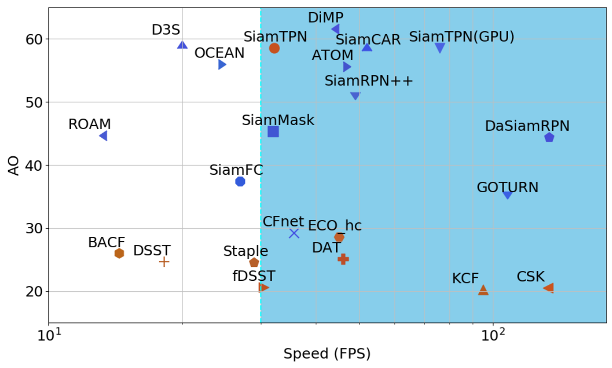


Figure 1: A comparison of the quality and the speed on CPU (dark red) or GPU (blue) of tracking methods on Got10K test set. The Average Overlap (AO) with respect to the Frames-Per-Seconds (FPS) is presented. The blue area corresponds to the trackers running in real-time speed (above 30 FPS).

Due to the optimization of both software and hardware on mobile devices and the progress of the lightweight but powerful backbone networks [24, 36, 41], the real-time applications based on visual classification, object detection, instance segmentation have been implemented on the CPU end. However, designing an efficient and effective object tracker for UAVs with limited computing resources, such as a single CPU-core, remains challenging. The lightweight backbones are insufficient for extracting robust discriminative features, which is vital for the tracking performance, especially under uncertainty scenarios. Thus, previous trackers try to address this problem by employing deeper networks [26], designing complex structures [50], or online updaters [2], which sacrifice the inference speed.

In this work, we alleviate the aforementioned problems,

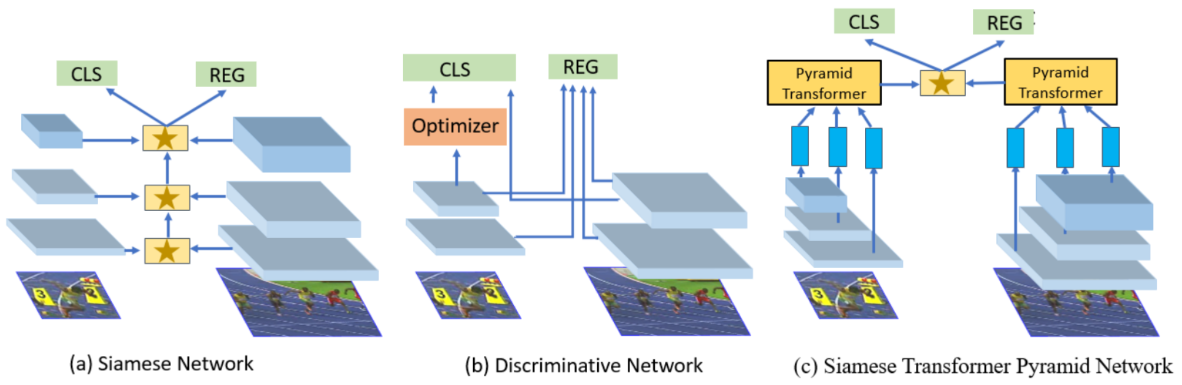


Figure 2: Object tracking architecture comparison. (a) Siamese based tracking network which operates cross correlation on pyramid layers separately. (b) Discriminative network that uses pyramid features for different tasks. (c) The proposed SiamTPN where features are first fused by the pyramid transformer module before used for both classification and regression.

accommodate the lightweight backbone and build a real-time CPU-based tracker. Firstly, to complement the representative ability of lightweight backbone network, we integrate the Feature Pyramid Network (FPN) [30] into the tracking pipeline. Although existing trackers [7, 15, 27] also employ multi-scale features, most of them resort to a simple combination or use features for different tasks. We claim that this is fundamentally limited since a discriminative representation requires combining the contexts from multiple scales. Even though, FPN encodes the pyramid information from low/high level semantics, it only exploits contexts from local neighborhoods rather than explicitly modeling the global interactions. The perception of the FPN is constrained by the receptive field, which is limited on the shallower networks. Inspired by the development of Transformer [5] and its ability to model global dependencies, recent works [13, 49] introduce attention-based modules and achieve profound results. However, the complexity of these models may cause computation/memory overhead which is not suited for pyramid architecture. Instead, we design a lightweight Transformer attention layer and embed it into pyramid network. The proposed Siamese Transformer Pyramid Network (named SiamTPN) augments the target features with lateral cross attention between pyramid features, producing robust target-specific appearance representation. Figure 2 illustrates the main difference between our tracker and existing ones. Moreover, our tracker based on a lightweight backbone network achieves state-of-the-art results while running at real-time speed on both GPU and CPU end, as shown in Figure 1. Our main contributions are summarized as follows:

1. We introduce a novel Transformer-based tracking framework for systems with limited computational resouces. These systems are typical encountered in UAVs with only CPU-support. To the best of our knowledge, this is the first deep learning based visual tracker running at real-time speed on UAVs using CPUs.

2. We propose a lightweight Transformer layer and integrate it into pyramid networks to build an efficient and effective framework.

3. Superior performance on multiple benchmarks as well as extensive ablation studies demonstrate the effectiveness of the proposed method. Particularly, our approach achieves state-of-the-art results and an AUC score of 58.1 on LaSOT [14] with only a lightweight backbone while running at over 30 FPS on the CPU end. The field tests further validate the efficiency of SiamTPN in real world applications.

2. Related Work

2.1. Lightweight Network

With the requirement of running neural networks on mobile platforms, a series of lightweight models are proposed [24, 36, 41]. AlexNet [24] utilizes fully convolutional operations and achieves profound results on ImageNet [12] classification tasks. MobileNet [41] family proposes inverted residual block, depthwise separate convolution to save computation cost. The ShuffleNet [36] family is another series of lightweight deep neural networks which introduce channel shuffle operation and optimize the network design for the target hardware.

Feature Pyramid Network The feature pyramid (i.e. bottom-up feature pyramid) is the most common architecture in modern neural network design. The hierarchical structure of CNN encodes the contexts in the gradually increased receptive field. The Feature Pyramid Network (FPN) [30] and Path Aggregation Network (PANet) [32] are commonly used for the cross-scale feature interaction and multi-scale feature fusion. FPN includes a bottom-up as well as a top-down path to propagate semantic information into multi-level features.

2.2. Object Tracking

Discriminative correlation filter (DCF). DCFs have shown promising results for object tracking since MOSSE [3] and KCF [19]. After that, multi-channel features, color names and multi scale features are used [9, 39] to improve the tracking robustness. Further improvements are achieved with non-linear kernels [10, 28], long-term memory [8] and deep features [11, 17]. [21, 29] further improves the robustness and optimized DCF for UAV tracking.

Deep learning based object tracking The popular Siamese network family based trackers address object tracking via similarity learning. SiamRPN [27] introduces the region proposal network to jointly perform classification and regression. DaSiamRPN [51] improves the discrimination power of the model with a distractor-aware module and SiamRPN++ [26] further improves the performance with more powerful deep architectures. Recent works like SiamBAN [6], SiamFC++ [47] and Ocean [50] replace the RPN with an anchor-free mechanism and achieve faster tracking speed. DiMP [2] and ATOM [7] learn a discriminative classifier online to distinguish the target from the background. These methods require intense calculation which is not suitable for CPU-based tracking.

Transformer. Transformer was first proposed for machine translation in [45] and shows great potential in many sequential tasks. DETR [5] first migrated Transformer into object detection tasks and achieves remarkable results. Recent works [13, 49] introduce an attention mechanism for improving the tracking performance. Motivated by DETR, [4] make use of transformer to directly fuse correlation maps from different levels and obtains remarkable accuracy and speed for object tracking on UAVs. Instead of migrating the complex transformer encoder and decoder paradigm, in this work, we exploit the transformer encoder and design an attention based feature pyramid fusion network to learn the target-specific model more efficiently.

3. Proposed Method

As shown in Figure 2, the proposed SiamTPN, consists of three modules: one Siamese backbone network for feature extraction, one Transformer based feature pyramid network and one prediction head for per-pixel classification and regression.

3.1. Feature Extraction Network

Similar to a Siamese tracking framework, the proposed SiamTPN consists of two branches: the template branch, which takes cropped image z of size $W_z \times H_z$ from the initial frame as reference, and the search branch, which takes the cropped image x of size $W_x \times H_x$ from the current frame for tracking. The two inputs are processed by the same backbone network, obtaining pyramid feature maps $P_i \in \mathbb{R}^{C_i \times \frac{W}{R} \times \frac{H}{R}}$, where $i \in \{3, 4, 5\}$ is the stage number of feature extraction and R is spatial reduction ratios, $R_i \in \{8, 16, 32\}$.

Instead of performing cross-correlation directly on feature maps pairs, we first feed the feature pyramid into the TPN (details in Section 3.3), shared between template branch and search branch. Specifically, TPN takes pyramid features P_3, P_4, P_5 as input and output the blend representation with the same size of P_4 for correlation purpose. Then, a depth-wise correlation is performed between outputs from reference branch and search branch as:

$$M = \Gamma(P_3^x, P_4^x, P_5^x) \star \Gamma(P_3^z, P_4^z, P_5^z)\,, \tag{1}$$

where Γ is the TPN module, and M is a multi-channel correlation map and is adopted as the input to the classification and regression head. The overall architecture is shown in Figure 2.

3.2. Feature Fusion Network

Multi-head Attention. Generally, a Transformer has several encoder layers, and each encoder layer is composed of Multi-head attention (MHA) module and a multilayer perceptron (MLP) module. The attention function is operated on queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ in the scale dot-production way, which can be expressed as:

$$\text{Attention}\,(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{(\mathbf{Q}+\mathbf{Pos})(\mathbf{K}+\mathbf{Pos})^\top}{\sqrt{C}}\right)\mathbf{V} \tag{2}$$

where the C is the key dimensionality to normalize the attention , and $\mathbf{Pos}$ is the positional encoding that are added to the input of each attention layer. The positional embedding in Transformer architectures is a location-dependent trainable parameter vector that is added to the token embeddings prior to inputting them to the Transformer blocks. The model representation capability is enhanced when extending the attention mechanism into multiple head way, which can be formulated as follows:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\mathbf{H}_1, \ldots, \mathbf{H}_N\right) W^O, \tag{3}$$

$$\mathbf{H}_i = \text{Attention}\left(\mathbf{Q}W_i^Q, \mathbf{K}W_i^K, \mathbf{V}W_i^V\right), \tag{4}$$

where $W_i^Q \in \mathbb{R}^{C\times d_{head}}, W_i^K \in \mathbb{R}^{C\times d_{\text{head}}}, W_i^V \in \mathbb{R}^{C\times d_{\text{head}}}$ and $W^O \in \mathbb{R}^{C\times C}$ are parameters of linear projections, Concat refers to concatenation operation, N is the number of attention head, and d_{head} is the dimension of each head equal to $\frac{C}{N}$.

Pooling Attention. MHA made the model assign importance to different aspects of information and learns a robust representation. However, the complexity increases with the power of input size. The computational cost for MHA is:

$$\mathcal{O}(MHA) = 2 \times n_q n_{kv} C + n_q C^2 + n_{kv} C^2, \tag{5}$$

where $n_q = h_q w_q, n_{kv} = h_{kv} w_{kv}$, w, h is the resolution of input feature map. There exist three ways to reduce the computation cost: (1) reduce the query size, (2) reduce the dimension of C, or (3) reduce the key and value size. However, reducing the query size also reduces the number of points for the prediction head, which eventually affects tracking accuracy. The same situation happens with the reduction of feature dimensionality. Since the feature maps with variable resolution are used as keys and values for fusion in TPN, we propose a pooling attention (RA) layer to reduce the spatial scale of $\mathbf{K}$ and $\mathbf{V}$. Specifically, the $\mathbf{K}$ and $\mathbf{V}$ are fed into a pooling layer with both pooling and stride size of R.

To further reduce the computation cost of attention module, we remove the position encoding in original MHA for the following reasons: (1) the permutation of the input tokens is constrained by the final cross correlation. (2) Accessing and storage of the position embedding for each feature maps costs extra resources which is not suited for mobile devices. Overall, the mechanism of PA block (PAB) can be summarized as:

$$\mathrm{PAB}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, R) = \mathrm{Norm}(\mathbf{F} + \mathrm{MLP}\,(\mathbf{F})), \tag{6}$$

$$\mathbf{F} = \mathrm{Norm}(\mathbf{Q} + \mathrm{PA}(R)\,(\mathbf{Q}, \mathbf{K}, \mathbf{V})), \tag{7}$$

where MLP is a fully connected feed-forward network, and Norm is the LayerNorm to smooth the input feature. The structure comparison between MHA and PA module is shown in Figure 3.

3.3. Transformer Pyramid Network

To leverage the pyramid feature hierarchy P_i, $i \in \{3, 4, 5\}$, which has both low-level information and high-level semantics, a Transformer Pyramid Network (TPN) is proposed to build a blend feature with high-level semantics throughout. The TPN consists of stacked TPN blocks which takes pyramid features $\{P_3, P_4, P_5\}$ and output new fusion feature $\{P_3^{'}, P_4^{'}, P_5^{'}\}$, as shown in Figure 4. The pyramid features are fed into a 1×1 convolution layer for dimension reduction, following a flatten operation before processing in the TPN. We fix the feature dimension (numbers of channels), denoted as C in all the feature maps.

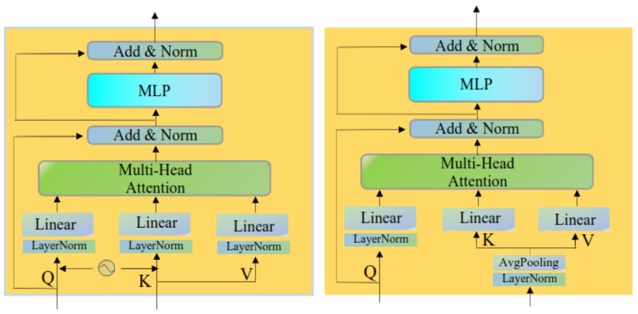


Figure 3: **Multi-head attention module versus Pooling Attention (PA) module.** Compared with the original attention block, the memory and time complexity in PA module is independent to the size of input features and controlled by the pooling operation

The construction of the pyramid features involves a bottom-up pathway and centralized pathway. The bottom-up pathway is the feed-forward convolution from the backbone architecture and produces feature hierarchy $\{P_3, P_4, P_5\}$. Then a centralized pathway merges the feature hierarchy into a unified feature. Specifically, we use P_4 as query for all feature hierarchy, yielding 3 combinations with different pooling scales which are processed by three parallel PAB locks. The outputs are directly added and fed into two self-attention PAB blocks to get the final semantic feature. The whole processing can be formulated as:

$$\begin{gathered} P_4^{'} = \mathrm{PAB}(P_4, P_3, P_3, R = 4) + \\ \mathrm{PAB}(P_4, P_4, P_4, R = 2) + \mathrm{PAB}(P_4, P_5, P_5, R = 1) \end{gathered} \tag{8}$$

$$P_4^{'} = \{\mathrm{PAB}(P_4^{'}, P_4^{'}, P_4^{'}, R = 2)\}_{n=2} \tag{9}$$

$$P_5^{'} = P_5; P_3^{'} = P_3. \tag{10}$$

P_3 and P_5 are set as identity ones to avoid computation/memory overhead. Moreover, PA block design guarantee that the interdependencies among hierarchical features can be raised efficiently. The TPN Block repeats B times and produces the final representation for cross-correlation and the final prediction. Simplicity is central to our design and we have found that our model is robust to various design choices.

3.4. Prediction Head

The fusion features P_4^x and P_4^z are reshaped back to the original size before fed into the prediction head. Following [26], the Depth-wise Cross Correlation is performed between the search map and the template kernel to get a mult-channel correlation map. The correlation maps are fed into two separate branches. Each branch consists of 3 stacked convolution blocks to generate final outputs $A^{cls}_{w\times h\times 2}$ and

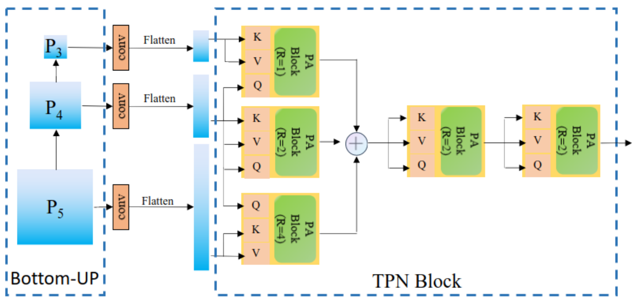


Figure 4: **Transformer Pyramid Network (TPN).** Features from different levels $P_3 - P_5$ are flattened and fed into TPN blocks. Each TPN block consists of 5 PA layers. Hierarchical information are extracted by 3 separate PA layers and further distilled by 2 additional PA layers. Variable stride pooling ratios R are assigned for each layer for efficiency purposes.

$A^{reg}_{w\times h\times 2}$. $A^{cls}_{w\times h\times 2}$ represents the foreground and background scores for each point on feature maps and $A^{reg}_{w\times h\times 2}$ predicts the distances from each feature point to the four sides of the bounding box. Overall, the objective function is

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{iou}\mathcal{L}_{iou} + \lambda_{reg}\mathcal{L}_{reg}, \tag{11}$$

where $\mathcal{L}_{cls}$ is the cross-entropy loss for classification, $\mathcal{L}_{iou}$ is GIOU [40] loss between prediction boxes and ground truth box and $\mathcal{L}_{reg}$ is the $L1$ loss for regression. Constants λ_{cls}, λ_{reg} and λ_{iou} weight the losses.

4. Experimental Studies

This section first presents the implementation details and the comparisons between variants of the SiamTPN tracker, with the cross-correlation visualization results. Then, ablation studies are presented to analyze the effects of the key components. We further compare our method with the state-of-the-art methods both on aerial and prevalent benchmarks. Finally, we deployed our tracker on a UAV platform to test its effectiveness in real-world applications.

4.1. Implementation Details

Model We apply our SiamTPN to three representative lightweight backbones, namely AlexNet [24], MobileNetV2 [41], ShuffleNetV2 [36]. Using those networks as backbones enables us to adequately compare the effectiveness of proposed method. All backbones are pretrained on Imagenet. The details of backbone configuration for the different backbones are shown in Table 1. For ShuffleNet and MibileNet, we extract that the stages of spatial ratio equal to $\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ respectively. For AlexNet, the last three layers are used for building feature pyramid.

Backbone	AlexNet [24]		MobileNet [41]		ShuffleNet [36]	
	stage	C	stage	C	stage	C
P_3	3	384	2	32	2	116
P_4	4	384	4	96	3	232
P_5	5	256	6	320	4	464
#Param (M)	3.1		1.81		0.8	
GFLOPs	4.33		0.39		0.16	

Table 1: Backbone configurations. #Param refer to the number of parameters. Multi-Adds GFLOPS is calculated under the input size of 256×256 for feature extraction. C is the dimension of the stage.

Training Like the Siamese approaches, the network is trained offline with image pairs. The training data consists of the training splits from LaSOT [14], GOT10K [20], COCO [31] and TrackingNet [38] dataset. The image pairs are sampled from the videos with a maximum gap of 100 frames. The sizes of search images and templates are 256 × 256 pixels and 80 × 80 pixels respectively, corresponding to 4^2 and 1.5^2 times of the target box area, resulting in pyramid features $\{h^x_3 = h^x_3 = 32, h^x_4 = h^x_4 = 16, h^x_5 = h^x_5 = 8\}$ and $\{h^z_3 = h^z_3 = 10, h^z_4 = h^z_4 = 5, h^z_5 = h^z_5 = 3\}$. Even though the lower input resolution brings additional speed increment, it is not the focus of this paper, so we set the aforementioned sizes for all the following experiments. The test images are augmented with some perturbation in the position and scale.

For all backbones, the first layer and all BatchNorm layers are frozen during training. All experiments are trained for 100 epochs with 64 image pairs per batch. We use the ADAMW [33] optimizer with initial learning rate of 10^{-5} for the backbone and 10^{-4} for the rest of the parts. The learning rate drops by a factor 0.1 decay on 90 epochs and the loss terms are weights with $\lambda_{cls} = 5$,$\lambda_{iou} = 5$,$\lambda_{reg} = 2$ respectively. During tracking, the scale penalty and Hanning windows [18] is performed before selecting best prediction point from classification map $A^{cls}_{w\times h\times 2}$. The final bounding box is given by adding the offsets predicted in $A^{reg}_{w\times h\times 2}$ to the coordinates of the best prediction point.

4.2. Ablation Study

In this section, we verify the effectiveness of the proposed tracker from the following aspects: backbones choice, comparison with original Transformer and Convolution, the impact of TPN hyperparameters and the attention visualization. We follow the one-pass evaluation (Success and Precision) to compare different tracking configurations on the LaSOT [14] test set and report the Success (AUC) scores. LaSOT [14] is a large-scale long-term tracking benchmark which contains 280 videos for testing.

Backbone	Neck Type	C	N	B	Depth	AUC	Δ_{AUC}	#Params (M)	GFLOPs	FPS (CPU)	Δ_{FPS}
AlexNet [24]	Identity	192			0	31.6	-	3.94	5.73	13.3	-
	Conv	192		2	6	34.7	+3.1	9.62	8.31	4.5	-8.8
MobileNetV2 [41]	Identity	192			0	33.9	+2.3	2.58	0.95	43.3	+30.0
	Conv	192		2	6	39.2	+7.6	5.04	1.75	24.3	+11.0
ShuffleNetV2 [36]	Identity	192			0	34.1	+2.5	1.57	0.6	48.1	+34.8
	Conv	192		2	6	39.5	+7.9	3.56	1.4	31.2	+17.9
	FPN	192		2	6	47.5	+15.9	3.85	1.62	26.9	+13.3
	Trans	192	6	2	6	53.5	+21.9	4.24	1.79	22	+8.7
	TPNwoPA	192	6	2	6	58.7	+27.1	4.84	2.05	17.7	+4.4
	TPN	192	6	1	3	52.8	+21.2	3.92	1.08	33.2	+19.9
	TPN	128	4	2	6	46.2	+14.6	2.7	0.88	37.1	+23.8
	TPN	192	6	2	6	58.1	+26.5	4.24	1.31	32.1	+18.8
	TPN	256	8	2	6	58.4	+26.8	10.77	3.73	15.2	+1.9

Table 2: **Comparison with different backbones and fusion configuration.** "Identity" means no feature encoding between pyramid features and cross-correlation. "Conv" and "Trans" refer to the use of Convolution layer or original Transformer to encode features. In those cases, only P_4 is used since the is no pyramid information to merge. For "TPN" and "TPNwoPA" share the same setting except the PA blocks are replaced with the original Transformer. C is feature channel, N is the number of attention head and B is the repeat number of TPN blocks. The AUC score are tested on LaSOT [14] test set. Since a TPN block consists of 3 layers processing P_4, for fair comparison, each block in "Conv" and "Trans" represents 3 stacked corresponding layers.

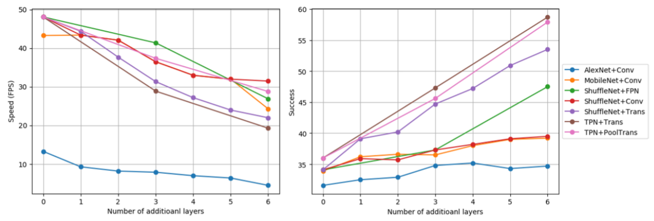


Figure 5: Speed and AUC score for different configurations.

Backbones. The backbone network has the dominant impact on inference speed and accuracy. Modern architectures make use of residual skip connection, group/depthwise convolution to design a competent network to learn more representative features, with even higher inference speed. We first compared the performance using different backbones. Similar to SiamFC [1], we remove all the feature fusion modules and predict results directly from P_4. We set C=192 for all prediction layers. As shown in Table 2, the tracker with a simple backbone with prediction head achieves appreciable AUC scores on LaSOT with an average high inference speed on CPU end. Specifically, ShuffleNetV2 achieves AUC score of 34.1 with 48.1 FPS. A straight forward question is: Will more attached convolution layers help with tracking performance? We then stack additional convolution layers following P_4 and Figure 5 shows the AUC changes along with the number of additional layers. Stacking more convolution layers improves the accuracy inefficiently and is worthless when compared with the speed drop. For ShufflenetV2, the speed drops over 30% at a 15% improvement on AUC score. We see that AlexNet is not suited for edge computing and ShuffleNetV2 and MobileNetV2 give comparable results both on accuracy and speed test. For the following experiments, we choose ShuffleNetV2 as the backbone.

Comparison with original Transformer. To show the effect of our proposed TPN module and PA block, we design a tracker using the original Transformer. Similar to the setting of stacked convolution, we attach additional Transformer layers behind P_4. As shown in Figure 5, without the fusion of pyramid features, the tracker with only one additional transformer layer achieves better results than the tracker with six additional convolution layers. Moreover, the tracker with six transformer layers achieves an AUC score of 53.5 on LaSOT. Next, we implement an FPN using the same settings as TPN, but replacing the transformer layers with convolution and interpolation layers. The tracker with two stacked FPN learns more comprehensive representations from the interactions inside the feature pyramid and gets an AUC score of 47.2, demonstrating its advantage over the single layer architecture. However, the lack of the global dependencies become the bottleneck of improving accuracy. We further integrate Transformer layer into TPN blocks without using Pooling Attention layer. With the high-level semantics aggregated from pyramid features, the tracker achieves an state-of-the-art performance on LaSOT with an AUC score of 58.7. However, we see that the speed of tracker drops below 20 FPS which is not applicable for real-time tracking requirement. Finally, we test the results of TPN model with PA layers instead of transformer layer.

Even the input size of queries and keys shrink with scale R, the tracker still achieves state-of-the-art performance . Nevertheless, the speed boosts up to 32.1 FPS with only 0.6 AUC score loss on LaSOT dataset, demonstrating the superiority of our method on both robustness and efficiency.

Impact of TPN hyperparameters. We discuss some architecture hyper parameters of the TPN model. Firstly, we examine the impact of the number of TPN blocks. With only one TPN block, the tracker produces a slight speed increment but suffers from AUC score drop from 58.1 to 52.8. Since the original transformer use 6 layers depth for both the encoder and decoder, we argue that 2 TPN blocks (depth=6) are enough for achieving robust tracking results. The number of heads in PA layer also plays an important role in tracking stability. For simplicity, we fix the head dimension=32, so we can test the input dimension $C = \{128, 192, 256\}$ and head number $N = \{4, 6, 8\}$ simultaneously. The tracker with 8 heads yields the best AUC score albeit at a cost of reducing in half of the inference time (FPS from 32.1 to 15.2). On the other hand, only using 4 heads is inefficient to learn an effective representation and only gives an AUC score of 46.2 on LaSOT. In practice, C=192, N=6, B=2 gives best balance between speed and accuracy.

Attention Visualization. The first three columns in Figure 6 show the response maps from the classification head with or without TPN module. Without TPN to learn discriminative features, the correlation results become dispersed and much easier to shift to distractors. The last three columns illustrate the attention maps between pyramid features. The attention between lower levels (P_3 to P_4, P_4 to P_4) distill more local information across the search area, while attention from high level (P_5 to P_4) is more centralized on the semantics of the object target. All attention maps are calculated from the central feature point inside the bounding box with the whole key inputs.

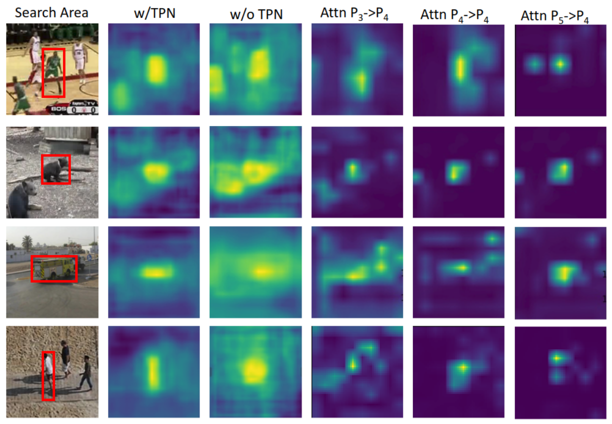


Figure 6: The visualization of response map with TPN (second column), without TPN (third column) and attention map between P_i and P_4.

4.3. Comparison with State-Of-The-Art Trackers

In this section, we compare our approach with 22 SOTA trackers. There are 4 anchor-based Siamese methods (SiamRPN [27], SiamRPN++ [26], DaSiamRPN [51], HiFT [4]), 5 anchor-free Siamese methods (SiamFC [1], SiamBAN [6], SiamCar [15], SiamFC++ [47], Ocean [50]), 10 DCF based methods (ECO [8], CCOT [11], KCF [19], ARCF [21], BACF [22], AutoTrack [29], CSRDCF [35], ROAM [48], DiMP [2], ATOM [7]), 2 attention based methods (CGACD [13], SiamAttn [49]) and 1 segmentaion based method, D3S [34].

UAV123 [37]. UAV123 is one of the largest UAV tracking benchmarks and adopts success and precision metrics for evaluation. As shown in Table 3, all trackers which achieve real-speed time on CPU are based on DCF, which rely on the handcraft features. This becomes a bottleneck of designing high-accuracy trackers. On the other hand, trackers relying on deeper networks like Resnet-50 can achieve high performance but are only applicable on GPU devices. Instead, our SiamTPN runs at real-time speed on the CPU while obtaining SOTA results. Specifically, SiamTPN gains a precision score of 85.8 and an AUC score of 66.04, outperforming the recent SOTA Siamese tracker SiamAttn. For a fair comparison, we develop a variant tracker based on AlexNet. While the AlexNet is not friendly on the CPU end, our tracker could run on the GPU at over 100 FPS while achieving consistent results with SiamRPN++.

VOT2018 [23] and OTB [46] The VOT2018 dataset consists of 60 sequences with different challenge factors. The performance is compared in terms of EAO (Expected Average Overlap). OTB contains 100 sequences and evaluates performance with AUC score. Table 4 shows that our method achieves comparable results with SOTA algorithms on both VOT (second row) and OTB (third row) datasets.

LaSOT [14]. Figure 7 shows our SiamTPN achieves best results on the LaSOT test set, with an AUC score of 58.1 and beats all trackers based on deep Resnet trackers (DiMP, ATOM, OCEAN).

Got10K [20] is another large-scale dataset and employs Average Overlap (AO) as measurement. Following the requirement of generic object tracking, there is no overlap in object categories between the training set and test set, which is more challenging and requires a tracker with a powerful generalization ability. We follow their protocol and train the network with a training split. As demonstrated in Figure 1, SiamTPN achieves a relative 12% higher performance on AO compared with the SOTA Siamese based tracker SiamRPN++ [26]. On the other hand, our method exceeds all DCF based trackers while keeping the real-time

Trackers	KCF [19]	BACF [22]	CSRDCF [35]	ARCF [21]	Auto Track [29]	ECO [8]	Siam RPN++ [26]	DaSiam [51]	HiFT [4]	Siam BAN [6]	Siam CAR [15]	Siam Attn [49]	DiMP [2]	ATOM [7]	Siam TPN	Siam TPN
Feat	HF	HF	HF	HF	HF	VGG	R50	Alex	R50	R50	R50	R50	R50	R18	Alex	Shuffle
Prec.	52.3	66.2	67.6	67.1	68.9	75.2	76.9	60.8	78.7	83.3	76	84.5	84.9	83.7	79	85.83
Succ.	33.1	46.1	48.1	46.8	47.2	52.2	57.9	40	58.9	63.1	61.4	65	65.4	65	59.3	66.04
FPS	95	14.4	58	15.3	65.4	45	35*	134*	130*	40*	52*	45*	45*	46*	105*	32.1

Table 3: Comparison results on UAV123 dataset [37] in terms of precision (Prec.), success (Succ.) and speed (FPS). HF refers to handcraft features, R50 (18) is Resnet-50 (18) [16], Alex, Shuffle, VGG represents AlexNet [24], ShuffleNet [36], VGGNet [42] respectively. GPU speeds are mark with *, Our SiamTPN based on AlexNet and ShuffleNet exhibit promising results. The top three trackers are shown in red, green and blue fonts.

	Siam RPN++ [26]	ATOM [7]	Dimp [2]	Siam FC++ [47]	CGACD [13]	SiamAttn [49]	SiamBAN [6]	Ocean [50]	Ours
EAO	41.4	40.1	44	42.6	44.9	47.0	45.2	48.9	46.2
AUC	69.6	66.7	68.4	68.3	71.3	71.2	69.6	68.4	71.0

Table 4: Evaluation on VOT and OTB datasets

inference speed on the CPU.

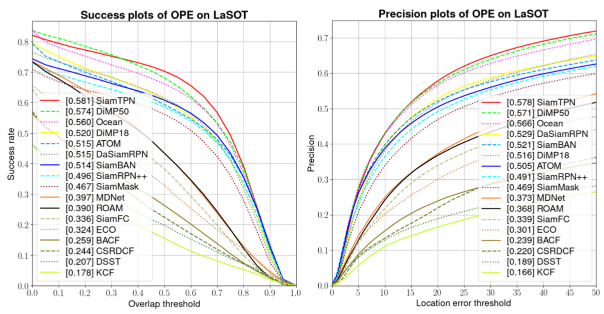
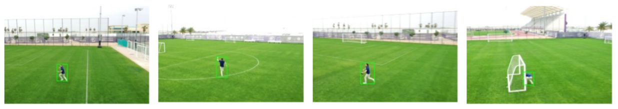


Figure 7: Evaluation results of trackers on LaSOT [14]

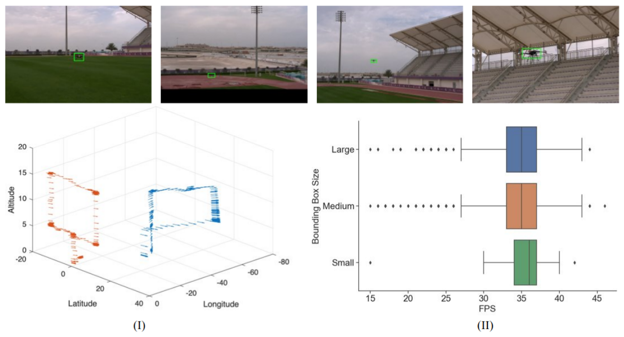
4.4. Real World Experimental Test

In this section, we verify the reliability of the proposed tracker in real-world UAV tracking. The hardware setup consists of a multi-copter UAV, an Embedded PC, a 3 axis Gimbal and a visual PTZ (pan-tilt-zoom) camera. We set up three different tracking scenarios to validate the tracking speed, generalization ability and robustness of SiamTPN. Specifically, the field tests include: (1) drone tracking with a ground stationary PTZ camera, as shown in Figure 8a. (2) tracking and following a moving person with a drone and keeping the target within the field of view, as shown in Figure 8b. (3) drone (evader) tracking with another drone (pursuer) with PTZ camera embedded, where two drones fly with custom trajectories but the parameters of PTZ camera are adjusted adaptively based on the position of evader, as shown in Figure 8c. The position of drones are recorded with two GPS devices and shown in Figure 8c(I), where the red (blue) dots correspond to the pursuer (evader). Figure 8 shows the precise tracking results obtained under complex environments, exhibiting the robustness and practicability of tracker in real-world applications. We also compare the tracking speed variance under different bounding boxes size. Empirically, we split the bounding boxes into three categories based on pixel numbers, which is small (< 1600), medium (< 10000) and large (> 10000) ones. Figure 8c(II) demonstrates the steady inference speed under varies circumstances.

(a) Drone tracking with ground PTZ camera



(b) Moving person tracking with a flying drone



(c) Drone tracking with PTZ camera mounted on a flying drone

Figure 8: Visualization of real-world tracking on drones

5. Conclusion

In this work, we propose a transformer pyramid network which aggregate semantics from different levels. The local interactions as well as the global dependencies are distilled from the cross attention among pyramid features. A pooling attention is further introduced to prevent the computation overhead. The comprehensive experiments demonstrate that our approach significantly improves the tracking results, while running at real-time speed on the CPU end.

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional Siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865. Springer, 2016.

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6182–6191, 2019.

[3] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.

[4] Ziang Cao, Changhong Fu, Junjie Ye, Bowen Li, and Yiming Li. Hift: Hierarchical feature transformer for aerial tracking. *arXiv preprint arXiv:2108.00202*, 2021.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[6] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6668–6677, 2020.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4660–4669, 2019.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6638–6646, 2017.

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2016.

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

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

[12] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009.

[13] Fei Du, Peng Liu, Wei Zhao, and Xianglong Tang. Correlation-guided attention for corner detection based visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6836–6845, 2020.

[14] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5374–5383, 2019.

[15] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6269–6277, 2020.

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

[17] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[18] Joao F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European Conference on Computer Vision*, pages 702–715. Springer, 2012.

[19] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2014.

[20] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[21] Ziyuan Huang, Changhong Fu, Yiming Li, Fuling Lin, and Peng Lu. Learning aberrance repressed correlation filters for real-time uav tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2891–2900, 2019.

[22] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1135–1143, 2017.

[23] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka ˇCehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al. The sixth visual object tracking VOT2018 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[25] Kuan-Hui Lee and Jenq-Neng Hwang. On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, 17(9):1429–1438, 2015.

[26] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of Siamese visual

tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

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

[28] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming-Hsuan Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4904–4913, 2018.

[29] Yiming Li, Changhong Fu, Fangqiang Ding, Ziyuan Huang, and Geng Lu. AutoTrack: Towards High-Performance Visual Tracking for UAV With Automatic Spatio-Temporal Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11920–11929, 2020.

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[31] 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 *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[32] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[34] Alan Lukezic, Jiri Matas, and Matej Kristan. D3s - a discriminative single shot segmentation tracker. In *CVPR*, 2020.

[35] Alan Lukezic, Tomas Vojir, Luka ˇCehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017.

[36] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient CNN architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[37] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *European Conference on Computer Vision*, pages 445–461. Springer, 2016.

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

[39] Horst Possegger, Thomas Mauthner, and Horst Bischof. In defense of color-based model-free tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[40] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on CVPR*, pages 658–666, 2019.

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on CVPR*, pages 4510–4520, 2018.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on CVPR*, pages 3539–3548, 2017.

[44] Athanasios Tsoukalas, Daitao Xing, Nikolaos Evangeliou, Nikolaos Giakoumidis, and Anthony Tzes. Deep learning assisted visual tracking of evader-UAV. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 252–257. IEEE, 2021.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[46] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013.

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

[48] Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B Chan. ROAM: Recurrently Optimizing Tracking Model. In *CVPR*, 2020.

[49] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6728–6737, 2020.

[50] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *The European Conference on Computer Vision (ECCV)*, August 2020.

[51] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware Siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.