Global Filter Pruning with Self-Attention for Real-Time UAV Tracking

Mengyuan Liu[1]
mengyuaner1122@qq.com
Yuelong Wang[1]
wang_yuelong@139.com
Qiangyu Sun[2]
centuryqsun@163.com
Shuiwang Li(✉)[1,3]
lishuiwang0721@163.com

[1] Guilin University of Technology, China
[2] Hubei Enshi College, China
[3] Guangxi Key Laboratory of Embedded Technology and Intelligent System, China

Abstract

Unmanned aerial vehicle (UAV)-based tracking has wide perspective applications, however, due to the limitations of computing resources, battery capacity, and maximum load of UAVs, efficiency seems more thorny and imperative as an issue than precision in UAV tracking, which explains why discriminative correlation filters (DCF)- instead of deep learning (DL)- based trackers are usually preferred in this field, as the former are known for high efficiency, whereas, the latter hardly achieve real-time tracking on a single CPU. However, without deep representation learning the precision of DCF-based trackers is extremely limited. This paper dedicates to boost the efficiency of DL-based trackers for UAV tracking by presenting a global filter pruning and proposes a self-attention module, which seeks to learn backbone features that highlight meaningful visual inter-dependencies, in order to combat the precision drop, and, importantly, to avoid the arduous process of determining layer-wise pruning ratios in the original ranked-based filter pruning method. Remarkably, self-attention is utilized here to guide the training without introducing any extra computational burden into the inference phase. Extensive experiments on four UAV benchmarks show that the proposed tracker strikes a remarkable balance between efficiency and precision and achieves state-of-the-art performance in UAV tracking.

1 Introduction

Unmanned aerial vehicle (UAV)-based tracking has wide perspective applications in navigation, aviation, transportation, agriculture, public security and many other fields, and is developing rapidly recently [5, 28, 30, 46]. Undoubtedly, their wide and mature applications rely heavily on the tacking precision and efficiency. However, UAV tracking faces more severe challenges compared with visual tracking in general scenes [26, 28, 30, 31]. On the one hand, great challenges are posed to the *precision* of the UAV tracking algorithms by e.g. extreme viewing angle, motion blur, scale changes, and severe occlusion; on the other hand, huge challenges are raised to their *efficiency* by e.g. limited computing resources, battery capacity limitations, low power consumption requirements, and UAV's maximum load [26, 31, 32]. Nevertheless, at the current technical level, efficiency seems more thorny and imperative in UAV tracking, which explains why discriminative correlation filters (DCF)- instead of deep learning (DL)- based trackers are usually preferred in the UAV tracking community, since the former are known for high efficiency, whereas, the latter, especially state-of-the-art ones, hardly achieve real-time tracking on a single CPU despite their relatively higher precision, hindering their deployment on UAVs to a great extent. Although their efficiency is more favourable, without the great power of deep representation learning the precision of DCF-based trackers is so extremely limited that they hardly remain robustness under very challenging conditions. Very recently, in [5] an efficient and effective deep tracker for UAV tracking was proposed, uses a lightweight backbone for efficiency and a hierarchical feature transformer to combine features from shallow and deep layers for robust representation learning. Although it has obtained a good balance between efficiency and precision, and demonstrated remarkable performance in UAV tracking, this tracker is not yet real-time on a single CPU. But importantly, it suggests that better balance between efficiency and precision may be more easily achieved by effective and lightweight DL-based trackers than fiddling with DCF-based ones, which motivates us to explore implementing real-time yet effective DL-based trackers with model compression techniques.

Model compression aims to reduce the cost of large models by representing the model in a more efficient format with minimal impact on its performance, which are usually used to deploy deep networks in resource-constrained and low-power edge devices [8]. However, the selection of DL tracker and compression method makes a huge difference to our purpose of implementing real-time yet effective DL-based trackers. Considering DL-based trackers, SiamFC [1] is a very efficient DL-based tracker, based upon which SiamFC++ [47] demonstrates state-of-the-art performance in both precision and speed with the proposed regression branch and center-ness branch, which is chosen as the baseline DL-based tracker for compression. Regarding model compression methods, the rank-based filter pruning method proposed in [33] is straightforward yet efficient in training as no cumbersome retraining is required. But there is a shortcoming: layer-wise pruning ratios are difficult and time-consuming to decide. Adopting a **global pruning ratio** is a simple solution to this problem, which, however, may compromise the precision to a great extent. In view of recent advances in Natural Language Processing (NLP) and vision tasks (such as the Transformer architecture [37], BERT [13] and ViT [14]) are largely attributed to the attention mechanism, and, particularly, self-attention has shown great power in image classification and object detection tasks thanks to its capacity to learn meaningful visual inter-dependencies, in this paper, we use the self-attention mechanism to mitigate accuracy loss when pruning SiamFC++ [47] with a global pruning ratio. The proposed self-attention module seeks to learn backbone features that highlight meaningful visual inter-dependencies through guiding the finetuning process. We name the proposed method 'PS-SiamFC++', since our tracker is obtained by applying pruning with self-attention on SiamFC++. Our contributions can be summarized as follows:

- Our work provides a fresh perspective to improve efficiency and precision of UAV tracking by developing DL-based trackers with filter pruning method, which has not been well explored before.
- We proposed a method of global filter pruning with self-attention for real-time UAV tracking, with which the proposed PS-SimaFC++ can globally compress the baseline

SiamFC++ to about 60% of its original model size, meanwhile maintaining and even significantly boosting precision simultaneously.

- We evaluate our PS-SiamFC++ on four public UAV benchmarks, i.e., UAV123@10fps [35], DTB70 [29], UAVDT [15] and VisDrone2018 [45]. Experimental results show that the proposed PS-SiamFC++ achieves state-of-the-art performance.

2 Related Works

2.1 Visual Tracking

Modern trackers are classified as DCF-based trackers or DL-based trackers. Using DCF for visual tracking starts with the minimum output sum of squared error (MOSSE) filter [4]. Afterwards, great progresses have been witnessed [11, 17, 20, 21, 24, 25, 27, 51]. DCF-based trackers usually adopt handcrafted features and can be calculated in the Fourier domain, which leads to competitive performance with high efficiencies. Since efficiency is a critical aspect in UAV tracking, DCF-based trackers, therefore, dominate the UAV tracking community currently. However, because of the restricted representation capability of handcrafted features, DCF-based trackers frequently fail to retain robustness in complex situations. Deep learning for visual tracking has shown to be quite effective in recent years, dramatically improving tracking precision and robustness. SiamFC [1] employed the Siamese network to quantify the similarity between the target and search pictures, making it one of the first attempts to consider visual tracking as a generic similarity-learning issue. Many DL-based trackers using Siamese topologies have been presented since then. Recently, SiamRPN++ [23] and SiamBAN [7] use deeper architectures to further improve tracking precision. However, their tracking efficiency has dropped significantly. In contrast, SiamFC++ [47] is a simple yet powerful framework as it has a lightweight backbone and a quality assessment branch that is effective for enhancing performance. Unfortunately, despite its excellent GPU speed, its CPU speed appears to be too slow to fulfill strict real-time requirements (i.e., with a speed of $\gg 30$ FPS). In this work, we attempt to increase the efficiency of SiamFC++ while keeping as much precision as possible for UAV tracking.

2.2 Filter Pruning

Pruning is a common technique for compressing neural networks, which are classified as weight pruning and filter pruning. The former usually removes neurons or weights, but its acceleration on general-purpose hardware is hard to achieve. [3]. While filter pruning removes the entire filters or channels, it is much easier to achieve considerable speed-up [33]. The pruning ratio determines how many weights to eliminate, and it is generally settled in one of two methods. The first is a specified global ratio or a series of layer-wise ratios. The second option is to alter the pruning ratio indirectly, for example by employing a regularization-based pruning approach. However, the second method necessitates considerable technical modifications to attain specific ratios [40]. The pruning criterion determines which weights should be pruned. For filter pruning, Frobenius norm or sparsity of the filter response, and the scaling factor of the Batch Normalization layer are commonly used criteria [41]. Last but not least, to specify how the sparsity of the network changes from zero to the target number, i.e., pruning schedule, there are two typical choices [41, 44]: (1) a single step (one-shot), then finetune, (2) progressive pruning and training are interleaved. Although the progressive approach is better to the one-shot approach since it provides for more time for training, the latter is more efficient and can alleviate the effort of developing complex training strategies. By and large, filter pruning so far remains an open problem. Recently, Lin et al. [33] proposed an effective and efficient filter pruning approach. They scheduled the pruning in a one-shot manner, using the rank of the feature map in each layer as the pruning criterion, which simplifies the process of pruning to a great extent. However, this approach requires a laborious and time-consuming process to determine the layer-wise pruning ratios. We propose to use a global pruning ratio to get around this issue. Furthermore, to prevent a potential precise decrease, we utilize self-attention to guide the fintuning process seeking to learn backbone features that highlight meaningful visual inter-dependencies.

2.3 Self-Attention in Vision

Attention mechanism is an attempt to mimic the human brain action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks [37]. As a special case, self-attention at the outset is the primary workhorse in NLP as it is an effective and computationally efficient mechanism for capturing global interactions between words in a sentence. But self-attention has properties, such as content-based interactions, ability to capture long-range dependencies, flexibility to handle multiple types of data and etc, that make it a good fit for vision tasks as well [38]. For instance, Wang et al. [43] presented non-local operations for capturing long-range dependencies for video understanding, Fu et al. [16] proposed DANet for semantic segmentation, Zhang et al. [50] demonstrated the effectiveness of the self-attention in image generation, and Zhao et al. [53] explored two forms of self-attention for image recognition. The usefulness of self-attention in many NLP and computer vision tasks has already gotten the extensive identification. Although self-attention has spawned the rise of so many recent breakthroughs in NLP and computer vision, including the Transformer architecture [37], BERT [13] and ViT [14], they come at a cost considering the computing and memory overheads involved. In view of this and our goal of real-time DL-based trackers, in this paper we avoid using self-attention in inference, but instead exploit it to guide our tracker in the training phase, which enables us to boosts tracking precision without introducing additional computation overhead in the inference phase.

3 Proposed Method

3.1 PS-SiamFC++ Overview

The overview of the proposed PS-SiamFC++ is shown in Fig. 1. It consists of a backbone, a neck, a head network, and a self-attention module. The target patch Z and the search patch X are the inputs for the template branch and the search branch, respectively. The shared backbone network of the two branches is denoted by $\phi(\cdot)$. The cross-correlation operation is conducted to the output backbone features of the two branches before they are passed to subsequent classification and regression tasks. The features produced by the cross-correlation operation are formulated by:

$$f_l(Z,X) = \psi_l(\phi(Z)) \star \psi_l(\phi(X)), \psi_l \in \{\psi_{cls}, \psi_{reg}\}, \tag{1}$$

where $\psi_{cls}(\cdot)$ and $\psi_{reg}(\cdot)$ denote the layer that is specifically designed for the tasks of classification and regression, respectively. $\star$ represents the cross-correlation operation. The classification branch predicts the category for each location, and its output is denoted by

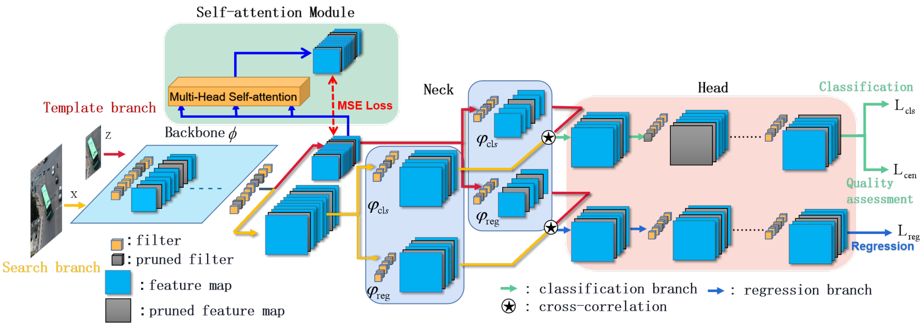

Figure 1: An illustration of the proposed PS-SiamFC++ method. The network structure is inherited from that of SiamFC++.

$O^{cls}_{h\times w\times 2}$; the regression branch is to compute the target bounding box at this location, and its output is denoted by $O^{reg}_{h\times w\times 4}$, where w and h are the width and height of the outputs, respectively. More specific, $O^{cls}_{h\times w\times 2}(i,j,:)$ is a 2D vector to record foreground and background scores of the location (i,j), while $O^{reg}_{h\times w\times 4}(i,j,:)$ is a 4D vector that shows how far the corresponding location is from the bounding box's four sides. The purpose of the centerness branch is to evaluate classification qualities. Its output is denoted by $O^{cen}_{h\times w\times 1}$, which is finally used to reweight the classification scores. The self-attention module is exploited to guide the finetuning process, which will be detailed in the following subsection.

3.2 Filter Pruning with Self-Attention

PS-SiamFC++ inherits the pipeline of SiamFC++ with the difference that the filters considered less important are pruned and a self-attention module is incorporated to guide the finetuning process. Let's first describe the rank-based filter pruning. We denote the i-th ($1 \leqslant i \leqslant K$) convolutional layer C^i of SiamFC++ by a set of 3-D filters $W_{C^i} = \{w^i_1, w^i_2, ..., w^i_m\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i}$, where n_i is the number of filters in C^i, k_i denotes the kernel size, and the j-th filter is $w^i_j \in \mathbb{R}^{n_{i-1}\times k_i \times k_i}$. The filters' output feature maps are denoted by $O_{C^i} = \{o^i_1, o^i_2, ..., o^i_m\} \in \mathbb{R}^{n_i \times g \times h_i \times w_i}$, where $o^i_j \in \mathbb{R}^{g\times h_i \times w_i}$ is associated with w^i_j, g is the number of input images, h_i and w_i denote the height and width of the feature maps, respectively. The rank-based filter pruning aims to minimize the following objective function:

$$\min_{\delta_{i,j}} \sum_{i=1}^{K}\sum_{j=1}^{n_i} \delta_{i,j}\mathbb{E}_{I\sim P(I)}[\mathcal{R}(o^i_j(I))], \quad s.t \sum_{j=1}^{n_i}\delta_{i,j} = n^i_p, \tag{2}$$

where I follows the $P(I)$ distribution representing an input image, n^i_p represents the number of filters to be pruned in C^i. $\delta_{i,j} \in \{0,1\}$ indicates whether or not the filter is pruned, $\delta_{i,j} = 1$ if it is, otherwise $\delta_{i,j} = 0$. $\mathcal{R}(\cdot)$ calculates a feature map's rank as a measure of how rich its information is. The expectation of the rank generated by a single filter is empirically proved to be robust to the input images [33], by which Eq. (2) is approximated by

$$\min_{\delta_{i,j}} \sum_{i=1}^{K}\sum_{j=1}^{n_i} \delta_{i,j} \sum_{t=1}^{g}\mathcal{R}(o^i_j(I_t)), \quad s.t \sum_{j=1}^{n_i}\delta_{i,j} = n^i_p, \tag{3}$$

where t indexes the input images. Eq. (3) is readily minimized by pruning npi filters that have the lowest average rank of feature maps.

After pruning the less important filters, the compressed network will be finetuned to optimize the parameters for the compressed model. To make the finetuning more productive, we utilize the self-attention mechanism to draw dependencies between spatial features, seeking to enhance significant parts while diminishing less informative parts of the features output by the pruned backbone for our tracking task. How the self-attention module plays a part is illustrated in Fig. 1. The backbone output in the template branch, denoted by f_Z, will be fed into a multi-head self-attention module to generate an enhanced feature representation f_Z^*, which is used in turn to supervise f_Z with the mean squared error (MSE) loss L_{mse}. The self-attention module consists of a Multi-Head Self-Attention layer (refer to supplementary material for its concrete structure). Intuitively, the attention mechanism describes a weighted average of (sequence) elements with the weights dynamically computed based on an input query and elements' keys. In our implementation, f_Z is encoded in a pixel-wise manner, i.e., the spatial coordinates of f_Z index the tokens, and the query, key and value are initially the same, for simplicity. Note that the output of the self-attention module is used for finetuning only, the module plays no part in the inference phase.

We now formulate the losses for finetuning the PS-SiamFC++. Let (x_0, y_0) and (x_1, y_1) denote the ground truth bounding box's left-top and right-bottom coordinates, and (x, y) denote the corresponding location of point (i, j), then the regression target $\hat{t}_{(i,j)} = \{\hat{t}^k_{(i,j)}\}_{k=0}^3$ of $O^{reg}_{h\times w\times 4}(i, j, :)$ is

$$\hat{t}^0_{(i,j)} = \hat{l} = x - x_0, \hat{t}^1_{(i,j)} = \hat{t} = y - y_0, \hat{t}^2_{(i,j)} = \hat{r} = x_1 - x, \hat{t}^3_{(i,j)} = \hat{b} = y_1 - y. \tag{4}$$

The differences between $O^{reg}_{h\times w\times 4}(i, j, :)$ and the regression target is penalized by the loss

$$L_{reg} = \frac{1}{\sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)})} \sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)}) L_{IOU}(O^{reg}_{(i,j,:)}, \hat{t}_{(i,j)}), \tag{5}$$

where L_{IOU} is the IOU loss as defined in [49], $\mathbb{I}(\cdot)$ is the indicator function defined as follow

$$\mathbb{I}(\hat{t}_{(i,j)}) = \begin{cases} 1 & if \quad \hat{t}^k_{(i,j)} > 0, k = 0,2,2,3 \\ 0 & \quad otherwise. \end{cases} \tag{6}$$

Denote $O^{cen}_{h\times w\times 1}(i, j)$, i.e., the centerness score at (i, j), by $\mathfrak{c}(i, j)$ as follows,

$$\mathfrak{c}(i, j) = \mathbb{I}(\hat{t}_{(i,j)}) * \sqrt{\frac{min(\hat{l}, \hat{r})}{max(\hat{l}, \hat{r})} \times \frac{min(\hat{t}, \hat{b})}{max(\hat{t}, \hat{b})}}. \tag{7}$$

The centerness loss is defined by

$$L_{cen} = \frac{-1}{\sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)})} \sum_{\mathbb{I}(\hat{t}_{(i,j)})=1} \mathfrak{c}(i, j) * log(O^{cen}_{h\times w\times 1}(i, j)) + (1 - \mathfrak{c}(i, j)) * log(1 - O^{cen}_{h\times w\times 1}(i, j)). \tag{8}$$

Finally, the overall loss for fintuning PS-SiamFC++ is:

$$L = L_{cls} + \lambda_1 L_{reg} + \lambda_2 L_{cen} + \lambda_3 L_{mse}(f_Z, f_Z^*), \tag{9}$$

where L_{cls} is the cross-entropy loss for classification, λ_1, λ_2, and λ_3 are predefined constants.

3.3 Pruning Schedule

The pipeline of pruning is: First, calculate the average rank of the feature map of any filter in each layer to obtain the rank sets $\{R^i\}_{i=1}^K = \{\{r^i_1, r^i_2, ..., r^i_{n_i}\}\}_{i=1}^K$. Second, each set R^i is sorted in decreasing order, resulting in $\bar{R}^i = \{r^i_{s^i_1}, r^i_{s^i_2}, ..., r^i_{s^i_{n_i}}\}$, where s^i_j denotes the index of the j-th top value in R^i. Third, perform filter pruning with a predefined global pruning ratio ρ, after which R^i turns to $\hat{R}^i = \{r^i_{s^i_1}, r^i_{s^i_2}, ..., r^i_{s^i_{\hat{n}_i}}\}$, $\hat{n}_i = n_i - n^p_i$ and incorporate the self-attention module to obtain the PS-SiamFC++ model. Finally, PS-SiamFC++ is finetuned after the remained filters are initialized with the original weights in the trained SiamFC++.

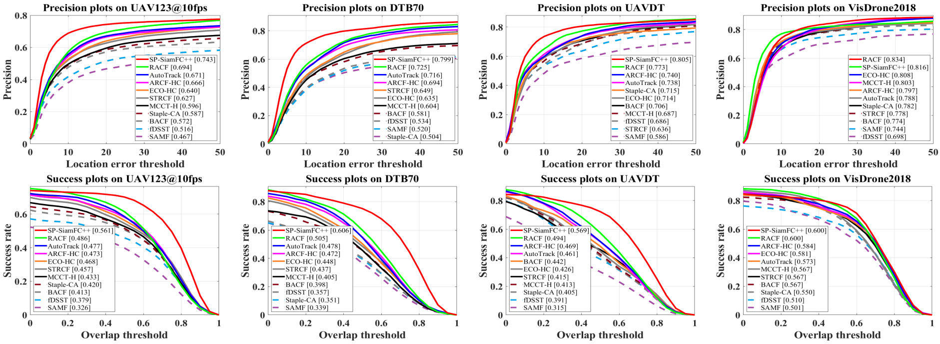

Figure 2: Overall performance of ten hand-crafted based trackers and our PS-SiamFC++ on UAV123@10fps, DTB70, UAVDT and VisDrone2018. For evaluation, precision and success rate for one-pass evaluation (OPE) are used. Precision at 20 pixels and area under curve (AUC) are utilized for ranking, respectively.

4 Experiments

4.1 Experiment Settings

Our experiments are conducted on four challenging UAV benchmarks, i.e., UAV123@10fps [35], DTB70 [29], UAVDT [13] and VisDrone2018 [45]. All evaluation experiments are conducted on a PC (with i9-10850K processor (3.6GHz), 16GB RAM, and an NVIDIA TitanX GPU) and on a tiny mini PC, i.e., Intel NUC (with an i5-1135G7 processor, 16GB RAM). Our PS-SiamFC++ is set with a global pruning ratio of 0.4. Other settings for training and inference follow SiamFC++ [47]. Code is available on: https://github.com/PS-SiamFCpp/PS-SiamFCpp.

4.2 Comparison with DCF-based Trackers

Ten state-of-the-art trackers with handcrafted features are used for comparison, including KCF [21], fDSST [9], Staple-CA [36], BACF [17], ECO-HC [10], MCCT-H [42], STRCF [24], ARCF-HC [22], AutoTrack [30], and RACF [26]. Fig. 2 demonstrates the overall performance of PS-SiamFC++ with the competing trackers on the four benchmarks. PS-SiamFC++ outperformed all other trackers by a significant margin on three benchmarks, namely UAV123@10fps, DTB70, and UAVDT. On the three benchmarks, in terms of precision[1] and area under curve (AUC), PS-SiamFC++ exceeds the second tracker RACF by (4.9%, 7.5%), (7.4%, 10.1%), and (3.2%, 7.5%), respectively. Although PS-SiamFC++ is inferior to the first tracker RACF in precision (by 1.8%), we get the best AUC when combined with RACF for VisDrone2018. It's worth noting that the settings of are dataset specific but ours are not. We evaluate the average FPS over the four benchmarks on the CPU of the PC and the NUC, respectively. Table 1 shows the average precision and FPS produced by different trackers. PS-SiamFC++ is the best real-time tracker (speed of >30FPS) on the PC and the NUC and it beats all rival trackers in terms of precision. Specifically, PS-SiamFC++ achieves 79.1% in precision at a frame rate of 71.3 FPS and 62.4 FPS on the PC and the NUC, respectively. Qualitative comparison can be found in the supplementary material.

[1]Unless otherwise specified, the precision metric in our experiment refers to distance precision at 20 pixels.

Table 1: Comparision of average precision and speed (FPS) between PS-SiamFC++ and hand-crafted based trackers on the four benchmarks. The reported FPSs are evaluated on a single CPU. Red, blue and green respectively show the first, second and third places.

	KCF TPAMI 15	fDSST CVPR 16	Staple-CA CVPR 17	BACF ICCV 17	ECO-HC CVPR 17	MCCT-H CVPR 18	STRCF CVPR 18	ARCF-HC ICCV 19	AutoTrack CVPR 20	RACF 3DV 21	**PS-SiamFC++ Ours**
Precision	53.3	60.4	64.2	65.3	68.8	66.8	67.1	71.9	**72.3**	**75.7**	**79.1**
FPS (PC)	**655.6**	**203.6**	67.7	57.0	**88.9**	66.7	29.9	36.0	61.8	37.5	71.3
FPS (NUC)	**572.1**	**177.6**	59.4	50.1	**77.8**	58.4	26.3	31.2	54.2	32.9	62.4

Table 2: Precision and speed (FPS) (evaluated on a single GPU) comparison between PS-SiamFC++ and deep-based trackers on UAVDT [15].

	SiamR-CNN CVPR 20	D3S CVPR 20	PrDimp18 CVPR 20	KYS ECCV 20	SiamGAT CVPR 21	LightTrank CVPR 21	TransT CVPR 21	HiFT ICCV 21	SOAT ICCV 21	AutoMatch ICCV 21	**PS-SiamFC++ Ours**
Precision	66.5	72.2	73.2	79.8	76.4	80.4	**82.6**	65.2	**82.1**	**82.1**	**80.5**
FPS (GPU)	7.2	44.6	48.5	30.2	74.8	84.8	42.1	**135.3**	29.4	50.4	**291.9**

In Fig. 3, we compare our method's qualitative tracking results to four top CPU-based trackers. As can be seen, when the objects are subjected to substantial deformations (i.e., BMX5), pose changes (i.e., truck1 and S0309), and partial occlusion (i.e., uav0000294 00000 s), the four CPU-based trackers eventually fail to accurately track the objects. However, thanks to the strong deep representation learning, our PS-SiamFC++ performs better and is visually more pleasing. It implies that building more efficient DL-based trackers, for example, by model compression, might be more beneficial in improving UAV tracking precision.

4.3 Comparison with DL-based Trackers

Ten state-of-the-art DL-based trackers are also compared with the proposed PS-SiamFC++, including PrDiMP18 [12], SiamR-CNN [39], D3S [34], KYS [2], SiamGAT [19], LightTrack [48], TransT [6], HiFT [5], SOAT [54], and AutoMatch [52]. Table 2 displays the FPSs and precisions of the trackers on UAVDT. As is shown, although PS-SiamFC++ (pruning ratio $\rho = 0.4$) is not as accurate as TransT, SOAT, AutoMatch and LightTrank, the gap is not more than 2.1% . Moreover, PS-SiamFC++ has a GPU performance about 7 times that of the first tracker TransT and 10 times that of the second tracker SOAT. PS-SiamFC++, in particular, achieves a precision of 80.5 percent and a GPU performance of 291.9 FPS. It achieves a fantastic balance between precision and efficiency when compared to the first tracker TransT, which achieves 82.6 percent precision and 42.1 FPS GPU speed (i.e., speed).

4.4 Ablation Study

Effect of pruning with self-attention: We integrate the proposed method into two baseline trackers, i.e., SiamCAR [18] and SiamFC++, and evaluate their performance with and without the proposed components (i.e., filter pruning and self-attention) on the four benchmarks. Table 3 shows the precisions and speeds evaluated on the PC. As can be seen, when applying filter pruning with a global pruning ratio of 0.4, resulting in P-SiamCAR and P-SiamFC++, the model size of both baselines are reduced to 60.0% of the original size, i.e., 5.1M and 5.8M, respectively. Their speeds are thus increase significantly. Specifically, from 40.7 FPS to 79.4 FPS and from 36.5 FPS to 71.3 FPS, respectively. However, all their precisions decrease, except the ones of SiamFC++ on UAVDT and VisDrone2018. This suggests that filter pruning, especially with a global pruning ratio, can either improve or decrease the precision of the model to be compressed, which depends on both the model itself and the dataset for evaluation. Remarkably, when the self-attention component is integrated into

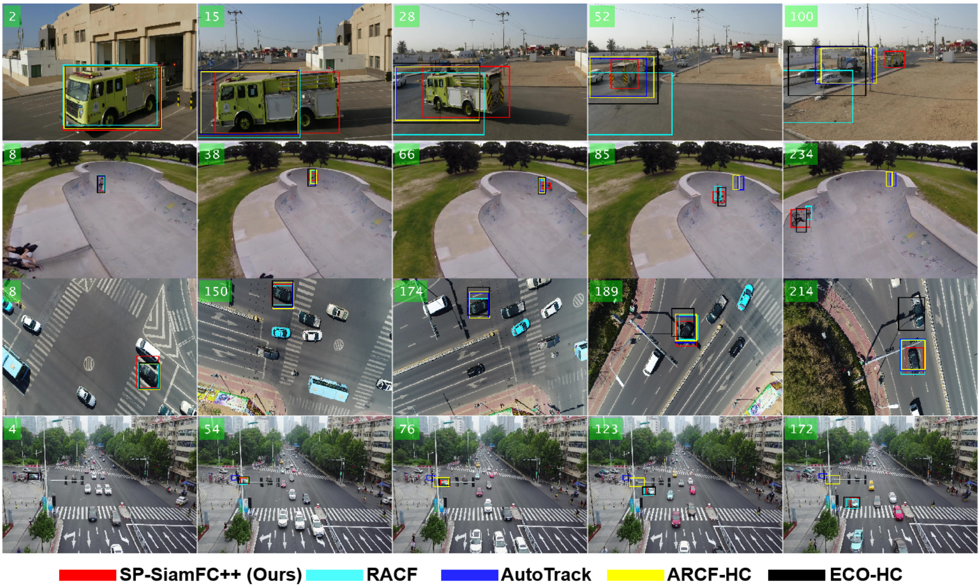

Figure 3: Qualitative evaluation on 4 sequences from UAV123@10fps, DTB70, UAVDT and VisDrone2018 (i.e. truck1, BMX5, S0309 and uav0000294_00000_s), respectively. The results of different methods have been shown with different colors.

Table 3: Comparison of the proposed PS-SiamFC++ tracker with two baseline trackers in terms of model size (Parameters), precision (DP), and tracking speed (FPS) on the PC CPU.

Methods	Parameters	Pruning	Self-attention	UAV123@10fps	DTB70	UAVDT	VisDrone2018	Avg. Precision	Avg. FPS (CPU)
SiamCAR [ ]	8.5M			**73.7**	76.6	76.1	**80.3**	**76.7**	40.7
P-SiamCAR	5.1M	✓		70.9	68.2	72.7	74.6	71.6	**79.4**
PS-SiamCAR	5.1M	✓	✓	71.0	73.2	77.1	75.8	74.1	**79.3**
SiamFC++ [ ]	9.7M			**72.8**	**80.5**	**76.2**	72.5	75.5	36.5
P-SiamFC++	5.8M	✓		71.9	**79.5**	**78.8**	**79.3**	**77.4**	**71.3**
PS-SiamFC++	5.8M	✓	✓	**74.3**	**79.9**	**80.5**	**81.6**	**79.1**	71.1

P-SiamCAR and P-SiamFC++, resulting in PS-SiamCAR and PS-SiamFC++, all the precisions of both compressed models are improved. For example, the precisions of P-SiamCAR on DTB70 and UAVDT are raised by 5.0% and 4.4% and the precisions of P-SiamFC++ on UAV123@10fps and VisDrone2018 are improved by 2.4% and 2.3%, respectively. In average, PS-SiamCAR and PS-SiamFC++ achieve an improvement of 2.5% and 1.7% in precision over P-SiamCAR and P-SiamFC++, respectively. Note that speeds of the models with and without the self-attention component are very close since the self-attention impacts the training only but not the inference of the model. These results justify the effectiveness of the proposed method, which can be attributed to the effectiveness of filter pruning for improving efficiency and the effectiveness of self-attention in highlighting relevant visual inter-dependencies thus providing more effective feature representations for tracking.

Impact of the global pruning ratio: To see how the global pruning ratio affects the final precision, S-SiamFC++ was trained and evaluated with different global pruning ratios. For further comparison, it was also trained and evaluated without the proposed self-attention module (i.e., P-SiamFC++). The global ratio ρ ranges from 0.1 to 0.8. Note that the higher the ratio, the more filters will be removed. Table 4 shows the precisions of PS-SiamFC++

Table 4: Illustration of how the precision on DTB70 of PS-SiamFC++ varies with the global pruning ratio, with or without the self-attention module. The precisions that have been improved by the self-attention component are marked in bold.

ρ	UAV123@10fps		DTB70		UAVDT		VisDrone2018	
	w/o	w/	w/o	w/	w/o	w/	w/o	w/
0.1	70.8	**72.2**	**79.6**	79.4	**81.4**	81.3	79.6	**83.1**
0.2	71.6	**72.3**	80.0	**80.1**	76.9	**77.2**	**80.2**	77.7
0.3	71.3	**72.4**	81.0	**81.5**	**83.9**	80.3	75.6	**79.3**
0.4	71.9	**74.3**	79.5	**79.9**	78.8	**80.5**	79.3	**81.6**
0.5	**71.1**	69.2	77.6	**79.2**	77.5	**78.0**	72.7	**76.4**
0.6	68.7	**69.2**	**78.6**	77.1	76.6	**78.3**	75.2	**76.9**
0.7	**69.1**	66.0	**77.9**	74.9	77.8	**80.7**	76.7	**79.4**
0.8	65.2	**66.0**	**76.4**	70.1	74.6	**77.0**	71.8	**74.9**

with and without the self-attention module with respect to the global pruning ratio. As can be seen, the highest precisions are primarily reached at ρ less than 0.5, which is consistent with our expectation for filter pruning techniques. Remarkably, incorporating the proposed self-attention improves most precisions. On UAV123@10fps and UAVDT, for example, six out of eight precisions are improved, and on VisDrone2018 seven out of eight precisions is raised. This allows us to maintain highly favorable precisions with larger pruning ratios, particularly when the global prunning ratio is set to the default value of 0.4, where the increases in precision are 2.4%, 0.4%, 1.7%, and 2.3%, on UAV123@10fps, DTB70, UAVDT, and VisDrone2018, respectively. This demonstrates the remarkable balance between prcision and efficiency achieved by our method, justifying the its effectiveness for real-time UAV tracking.

5 Conclusion

In this work, we present a method of global filter pruning with self-attention for real-time UAV tracking and achieve state-of-the-art performance on four public UAV tracking benchmarks. When using the proposed method to improve UAV tracking efficiency, experimental results reveal that the proposed method is quite effective at maintaining and even improving precision. Surprisingly, the proposed PS-SiamFC++ not only outperforms the baseline SiamFC++ in terms of efficiency (PS-SiamFC++ can run at and more than 62 FPS on a single CPU of a mini PC, i.e., Intel NUC), but it also outperforms the baseline in terms of precision on UAVDT and VisDrone2018, well combating the adverse effects of filter pruning.

6 acknowledgement

Thanks to the support by Guangxi Science and Technology Base and Talent Special Project (No. Guike AD22035127)

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, and et al. Fully-convolutional siamese networks for object tracking. In *ECCV,2016*, pages 850–865. Springer, 2016.

[2] Goutam Bhat, Martin Danelljan, Van Gool, and et al. Exploiting scene information for object tracking. In *ECCV*, pages 205–221. Springer, 2020.

[3] Davis Blalock, Jose Javier Gonzalez Ortiz, and et al. What is the state of neural network pruning? *arXiv:2003.03033*, 2020.

[4] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE CVPR*, pages 2544–2550. IEEE, 2010.

[5] Ziang Cao, Changhong Fu, Junjie Ye, and et al. Hift: Hierarchical feature transformer for aerial tracking. In *ICCV*, pages 15457–15466, 2021.

[6] Xin Chen, Bin Yan, Jiawen Zhu, and et al. Transformer tracking. In *CVPR*, pages 8126–8135, 2021.

[7] Zedu Chen, Bineng Zhong, Guorong Li, and et al. Siamese box adaptive network for visual tracking. In *CVPR,2020*, pages 6668–6677, 2020.

[8] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and et al. A comprehensive survey on model compression and acceleration. *Artif Intell Rev*, 53(7):5113–5155, 2020.

[9] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and et al. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*, pages 1430–1438, 2016.

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, pages 6931–6939, 2017.

[11] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2017.

[12] Martin Danelljan, LucVanGool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2021.

[15] Dawei Du, Yuankai Qi, Hongyang Yu, and et al. The unmanned aerial vehicle benchmark: Object detection and tracking. In *ECCV*, pages 375–391, 2018.

[16] J. Fu, J. Liu, Haijie Tian, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3141–3149, 2019.

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

[18] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6268–6276, 2020.

[19] Dongyan Guo, Yanyan Shao, Ying Cui, and et al. Graph attention tracking. In *CVPR*, pages 9543–9552, 2021.

[20] Yujie He, Changhong Fu, Fuling Lin, Yiming Li, and Peng Lu. Towards robust visual tracking for unmanned aerial vehicle with tri-attentional correlation filters. *arXiv preprint arXiv:2008.00528*, 2020.

[21] Joao F. Henriques, Rui Caseiro, Pedro Martins, and et al. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 37(3):583–596, 2015.

[22] Ziyuan Huang, Changhong Fu, Yiming Li, and et al. Learning aberrance repressed correlation filters for real-time uav tracking. In *ICCV*, pages 2891–2900, 2019.

[23] Bo Li, Wei Wu, Qiang Wang, and et al. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR,2020*, pages 4282–4291, 2019.

[24] Feng Li, Cheng Tian, Wangmeng Zuo, and et al. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, pages 4904–4913, 2018.

[25] Shui-wang Li, Qian-bo Jiang, Qi-jun Zhao, Li Lu, and Zi-liang Feng. Asymmetric discriminative correlation filters for visual tracking. *Frontiers of Information Technology & Electronic Engineering*, 21(10):1467–1484, 2020.

[26] Shuiwang Li, Yuting Liu, Qijun Zhao, and et al. Learning residue-aware correlation filters and refining scale estimates with the grabcut for real-time uav tracking. In *3DV*, pages 1238–1248, 2021.

[27] Shuiwang Li, Qijun Zhao, Ziliang Feng, and Li Lu. Equivalence of correlation filter and convolution filter in visual tracking. *ArXiv*, abs/2105.00158, 2021.

[28] Shuiwang Li, Yuting Liu, Qijun Zhao, and Ziliang Feng. Learning residue-aware correlation filters and refining scale for real-time uav tracking. *Pattern Recognition*, 2022.

[29] Siyi Li and Dit Yan Yeung. Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In *AAAI*, pages 4140–4146, 2017.

[30] Yiming Li, Changhong Fu, Fangqiang Ding, and et al. Autotrack: Towards high-performance visual tracking for uav with automatic spatio-temporal regularization. In *CVPR*, pages 11923–11932, 2020.

[31] Yiming Li, Changhong Fu, Ziyuan Huang, Yinqiang Zhang, and Jia Pan. Keyfilter-aware real-time uav object tracking. In *2020 IEEE ICRA*, pages 193–199, 2020.

[32] Fuling Lin, Changhong Fu, Yujie He, Fuyu Guo, and Qian Tang. Bicf: Learning bidirectional incongruity-aware correlation filter for efficient uav object tracking. In *2020 IEEE ICRA*, pages 2365–2371, 2020.

[33] Mingbao Lin, Rongrong Ji, Yan Wang, and et al. Hrank: Filter pruning using high-rank feature map. In *CVPR*, pages 1529–1538, 2020.

[34] Alan Lukezic, Jiri Matas, and Matej Kristan. D3s – a discriminative single shot segmentation tracker. In *CVPR,2020*, pages 7133–7142, 2020.

[35] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. *FJMS*, 2(2):445–461, 2016.

[36] Matthias Mueller, Neil Smith, and Bernard Ghanem. Context-aware correlation filter tracking. In *CVPR*, pages 1387–1395, 2017.

[37] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[38] Ashish Vaswani, Prajit Ramachandran, A. Srinivas, Niki Parmar, Blake A. Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12889–12899, 2021.

[39] Paul Voigtlaender, Jonathon Luiten, Philip H.S. Torr, and et al. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, pages 6578–6588, 2020.

[40] Huan Wang, Can Qin, Yulun Zhang, and et al. Neural pruning via growing regularization. *arXiv:2012.09243*, 2020.

[41] Huan Wang, Can Qin, Yulun Zhang, and et at. Emerging paradigms of neural network pruning. *arXiv:2103.06460*, 2021.

[42] Ning Wang, Wengang Zhou, Qi Tian, and et al. Multi-cue correlation filters for robust visual tracking. In *CVPR*, pages 4844–4853, 2018.

[43] X. Wang, Ross B. Girshick, Abhinav Kumar Gupta, and Kaiming He. Non-local neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[44] Xucheng Wang, Dan Zeng, Qijun Zhao, and Shuiwang Li. Rank-based filter pruning for real-time uav tracking. *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2022.

[45] Longyin Wen, Pengfei Zhu, Dawei Du, and et al. Visdrone-sot2018: The vision meets drone single-object tracking challenge results. In *ECCV*, pages 469–495, 2018.

[46] Wanying Wu, Pengzhi Zhong, and Shuiwang Li. Fisher pruning for real-time uav tracking. *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2022.

[47] Yinda Xu, Zeyu Wang, and Zuoxin andet al. Li. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, volume 34, pages 12549–12556, 2020.

[48] Bin Yan, Houwen Peng, Kan Wu, and et al. Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In *CVPR*, pages 15180–15189, 2021.

[49] Jiahui Yu, Yuning Jiang, Zhangyang Wang, and et al. Unitbox: An advanced object detection network. *Proceedings of the 24th ACM international conference on Multimedia*, 2016.

[50] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019.

[51] Zhewen Zhang, Fuliang Wu, Yuming Qiu, Jingdong Liang, and Shuiwang Li. Tracking small and fast moving objects: A benchmark. *ArXiv*, abs/2209.04284, 2022.

[52] Zhipeng Zhang, Yihao Liu, Xiao Wang, and et al. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, pages 13339–13348, 2021.

[53] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10073–10082, 2020.

[54] Zikun Zhou, Wenjie Pei, Xin Li, and et al. Saliency-associated object tracking. In *ICCV*, pages 9866–9875, 2021.