Rank-Based Filter Pruning for Real-Time Deep UAV Trackers

Xucheng Wang[a,1], Dan Zeng[b,1], Qijun Zhao[c], Shuiwang Li[a,*]

[a]*Guangxi Key Laboratory of Embedded Technology and Intelligent System, Guilin University of Technology, Guilin, 541001, China*
[b]*Research Institute of Trustworthy Autonomous Systems and Department of computer science and engineering, Southern University of Science and Technology, Shenzhen, 518000, China*
[c]*Sichuan University, Chengdu, 610000, China*

Abstract

UAV tracking is an emerging task and has wide potential applications in such as agriculture, navigation, entertainment and public security. However, the limitations of computing resources, battery capacity, and maximum load of UAV hinder the deployment of DL-based tracking algorithms on UAV. In contrast to deep learning trackers, discriminative correlation filters (DCF)-based trackers stand out in the UAV tracking community because of their high efficiency. However, their precision is usually much lower than trackers based on deep learning. Model compression is a promising way to narrow the gap (i.e., effciency, precision) between DCF- and deep learning- based trackers, which has not caught much attention in UAV tracking community. In this paper, We propose the P-SiamFC++ tracker, which is the first to use rank-based filter pruning to compress the SiamFC++ model, achieving a remarkable balance between efficiency and precision. Our method is general and may encourage further studies on UAV tracking with model compression in the future. Extensive experiments on four UAV benchmarks, including UAV123@10fps, DTB70, UAVDT and Vistrone2018, show that P-SiamFC++ tracker significantly outperforms state-of-the-art UAV tracking methods.

Keywords: filter pruning, UAV tracking, SiamFC++, real time
PACS: 0000, 1111

*Corresponding author.
E-mail address: xcwang@glut.edu.cn (Xucheng Wang), zengd@sustech.edu.cn (Dan Zeng), qjzhao@scu.edu.cn (Qijun Zhao), lishuiwang0721@163.com (Shuiwang Li)
[1]These authors contributed equally.

Electronic copy available at: https://ssrn.com/abstract=4159540

1. Introduction

With the deployment of unmanned aerial vehicles (UAVs) in numerous industries recently, unmanned aerial vehicle (UAV)-based tracking emerges as a new challenge and has drawn increasing interest in visual tracking. It has wide potential applications in such as agriculture, navigation, transportation, aircraft tracking, public security, autonomously landing, aerial refueling, disaster response, and etc. [1, 2, 3, 4, 5, 6, 7, 8, 9]. However, UAV tracking is not an easy thing and is even more difficult than general visual tracking since it faces more onerous challenges than in general scenes. On the one hand, motion blur, severe occlusion, extreme viewing angle, and scale changes have greatly challenged the *precision* of the UAV tracking algorithms. Among them, motion blur and scale changes are quite likely to occur when a drone flies very fast, causing the scene, the target appearance, and the target scale to significantly vary throughout a single camera exposure; while severe occlusion and extreme viewing angle are more of a phenomenon when the targets are captured from cameras onboard UAVs flying high in the sky. On the other hand, limited computing resources, low power consumption requirements, battery capacity limitations, and UAV's maximum load pose a big challenge to its *efficiency* as well [10, 4, 6].

At the current level of technology, efficiency is a fundamental problem of UAV tracking, since the limitations due to restricted resources onboard UAVs is very difficult to surmount without great progress in other fields. Therefore, discriminative correlation filters (DCF)-based trackers, thanks to the fast Fourier transform (FFT) [11, 12, 10], are usually preferred instead of deep learning (DL)-based trackers [4, 13, 6, 7, 14, 8, 15, 9]. Although tracking precisions of DCF-based trackers have been greatly improved, they are still not comparable to most state-of-the-art DL-based trackers. Very recently, Cao et al. [5] proposed an efficient and effective deep tracker for UAV tracking, which used a lightweight backbone for consideration of efficiency and utilized a hierarchical feature transformer to achieve interactive feature fusion of shallow layers and deep layers for robust representation learning [5]. This tracker, although having not yet achieved real-time tracking on a single CPU, has achieved a good balance between efficiency and precision and demonstrated state-of-the-art performance in UAV tracking, which suggests that an effective lightweight DL-based tracker may be a good alternative to a DCF-based tracker. We are thus inspired to exploit model compression to narrow the gap between DCF- and deep learning- based UAV trackers.

Electronic copy available at: https://ssrn.com/abstract=4159540

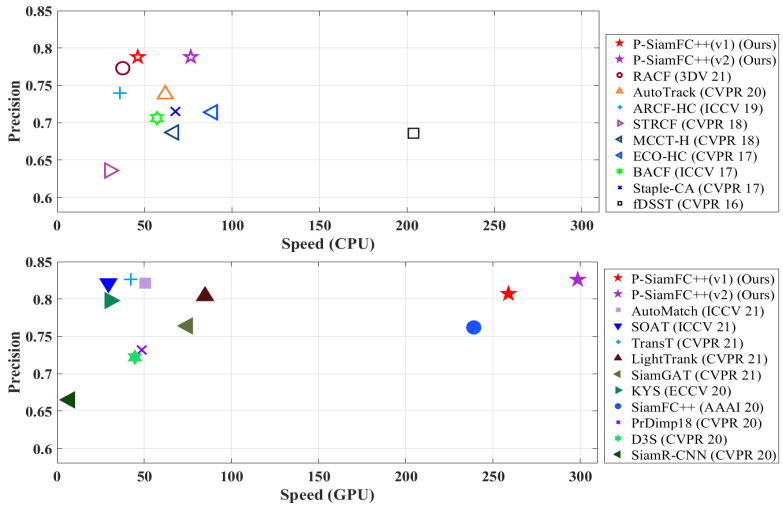


Figure 1: Comparing our method with CPU-based trackers and deep trackers in terms of precision and tracking speed (fps) on CPU and GPU, respectively.

Model compression is a technique usually used to deploy state-of-the-art deep networks in low-power and resource-constrained edge devices without compromising much on models' accuracy [16]. Popular and widely studied methods for model compression [17] include pruning, quantization, low-rank approximation, knowledge distillation, and etc. In this paper, we exploit the rank-based filter pruning method proposed in [18], which is efficient and straightforward, to compress the SiamFC++. We hence name the proposed method P-SiamFC++. The pruning method we used is very straightforward and training-efficient since it eliminates the need of introducing additional constraints and retraining the model. SiamFC++ is based on the efficient tracker SiamFC and shows state-of-the-art performance in both precision and speed by introducing a regression branch and a center-ness branch. To the best of our knowledge, we are the first to effectively use model compression to narrow the gap between DCF- and deep learning- based trackers in UAV tracking. We achieve a remarkable balance between efficiency and precision compared with existing CPU-based and DL-based trackers.We believe our work provides a fresh perspective for solving UAV tracking and will attract increasing attention in the UAV tracking community to this approach. Our contributions can be summarized as follows:

- We are the first to introduce model compression to UAV tracking to the best of our knowledge, which narrows the gap between DCF-based and

Electronic copy available at: https://ssrn.com/abstract=4159540

DL-based trackers. Surprisingly, the proposed method can improve both efficiency and tracking precision. The significant yet unexpected increase of the compressed model over the original one may encourage more work on this method.

- We propose the P-SiamFC++ tracker to exploit rank-based filter pruning for compressing the SiamFC++ model, achieving a remarkable balance between tracking efficiency and precision. Our method is general, real-time, and provides a fresh perspective to solve UAV tracking.
- We demonstrate the proposed method on four UAV benchmarks. Experimental results show that the proposed P-SiamFC++ tracker achieves state-of-the-art performance.

The rest of this paper is organized as follows. Section 2 gives an overview of the prior work relevant to this work. Section 3 describes the proposed method. In Section 4, we provide description and results of the performed experiments. Conclusions are finally given in Section 5.

2. Related Works

This paper extends our work [19]. In this paper, we improve the realization of the previous version and additionally perform a comprehensive analysis of the rank-based filter pruning for real-time UAV tracking. We extend our P-SiamFC++ by investigating block-wise pruning ratios, based upon which we can choose a better balance between tracking precision and efficiency. This enhancement enables us to obtain state-of-the-art real-time trackers, with an average speed above 70FPS on a single CPU. Note that the previous and the enhanced version are denoted by P-SiamFC++(v1) and P-SiamFC++(v2), respectively.

2.1. Visual Tracking Methods

Visual tracking is a fundamental task in computer vision. Nowadays, academics have presented a variety of novel tracking approaches in order to achieve performance perfection. Roughly speaking, modern trackers can be divided into two classes: DCF-based trackers and DL-based ones. DCF-based trackers start with a minimum output sum of squared error (MOSSE) filter [20]. After that, DCF-based trackers have made great progress in many variants by introducing kernel trick, discriminative scale estimation, continuous convolution, spatial and

Electronic copy available at: https://ssrn.com/abstract=4159540

temporal regularizations, training set management, deep features, attention, context and background information and so on [21, 10, 22, 23, 24]. Since DCF-based trackers usually use handcrafted features and can be calculated in the Fourier domain, they can achieve competitive performance with high efficiency. This is why they stood out in UAV tracking community given efficiency is a fundamental issue. However, they hardly maintain robustness under challenging conditions because of the poor representation ability of handcrafted features.

Thanks to the great success of deep learning, its application in visual tracking has proven to be very successful in recent years and it has significantly improved the precision and robustness of many trackers. As one of the pioneering works, SiamFC [25] considered visual tracking as a general similarity-learning problem and took advantage of the Siamese network to measure the similarity between target and search image. Since that, many DL-based trackers are based on Siamese architectures. Recently, the Siamese-based trackers are mainly divided into two categories, i.e., anchor-based and anchor-free trackers [26]. Regarding anchor-based methods, SiamRPN [27] applied a region proposal network (RPN) into Siamese networks and considered tracking as two subtasks performed by a classification and a regression branch; DaSiamRPN [28] proposed a distractor-aware module and an effective sampling strategy; SiamMask [29] added a new branch to produce a pixel-wise binary mask. More recently, deeper architectures are explored in such as SiamDW [30] and SiamRPN++ [31] to further improve tracking precision, but the efficiency is sacrificed largely. As to anchor-free trackers, SiamFC++ [32] introducing a novel quality assessment branch in classification is a simple but powerful framework, based upon which SiamCAR [33] achieved an impressive performance by redesigning a novel anchor-free structure and merging multi-layers features. Besides, SiamBAN [34] explores a new strategy of generating classification labels and regression targets. In addition to Siamese-based trackers, there are many DL-based trackers extending online discriminative framework with deep networks for end-to-end training, e.g., ATOM [35], DiMP [36], KYS [37] and KeepTrack [38].

Although deeper architectures have been developed to further improve tracking precision recently, the efficiency is sacrificed to a large extent. In contrast, SiamFC++ [32] is a simple but powerful framework that proposed an effective quality assessment branch to improve precision. Unfortunately, despite its excellent GPU speed, it cannot reach real-time speed (i.e., $\geqslant$ 30 FPS) on a single CPU. In this paper, we aim to improve the efficiency of SiamFC++ for real-time UAV tracking [32] using model compression techniques.

Electronic copy available at: https://ssrn.com/abstract=4159540

2.2. Model Compression by Pruning

In general the goal of model compression is to achieve a simplified model without significantly reducing the accuracy of the original model [16]. Pruning is a commonly used neural network compression technique that explores the redundancy in model weights and tries to remove/prune redundant and non-critical weights, which involves removing connections between neurons or entire neurons, channels, or filters from a trained network [18]. Pruning has been applied since the 1980s, but has seen an explosion of interest in the past decade thanks to the thriving of deep neural networks and their deployment to resource-constrained environments [16, 39]. Conventionally, a pruning pipeline is comprised of three steps: pretraining, pruning and finetuning. And there are four classic topics involved in the pipeline: pruning structure, pruning ratio, pruning criterion, and pruning schedule [17]. Pruning structure can be divided into two types: unstructured (weight) pruning and structured (filter) pruning. The former involves removing individual weights or neurons, which, however, is hard to leverage for acceleration on general-purpose hardware [39]. While the latter involves removing entire channels or filters, much easier to achieve considerable acceleration because of the regular arrangement of weights [18]. Pruning ratios indicate how many weights to remove. In general, there are two ways to adjust pruning ratio. The first is to pre-define one global ratio or many layer-wise ratios. The second approach is to adjust the pruning ratio indirectly, such as using a regularization-based pruning method, which removes weights by pushing them to zero with a penalty term. However, it demands much engineering tuning to achieve a specific ratio or group of ratios [40, 41]. Pruning criterion is used to select which weights to prune. For weight pruning, weight magnitude is the most simple criterion [42]. As to filter pruning, Frobenius norm (typically L_1-norm and L_2-norm) of a filter [43], sparsity of the filter response [44], and scaling factor of the Batch Normalization layer [45] are frequently used criteria. Besides, the idea of selecting the weights that induce the least loss increase and its variant are often practiced as well [46, 47]. Pruning schedule specifies how the network sparsity goes from zero to a target number, which has two typical choices [17]: (1) in a single step (one-shot), then finetune [18, 46], (2) progressively, pruning interleaved with training [48]. Although the progressive manner might outperform the one-shot way since more time is available for training, the latter is more efficient in training and frees one from designing complicated training strategy.

Overall, pruning remains an open problem so far. The HRank proposed recently in [18] is an effective and efficient filter pruning method that using the rank of the feature map in each layer as the pruning criterion and is scheduled in

Electronic copy available at: https://ssrn.com/abstract=4159540

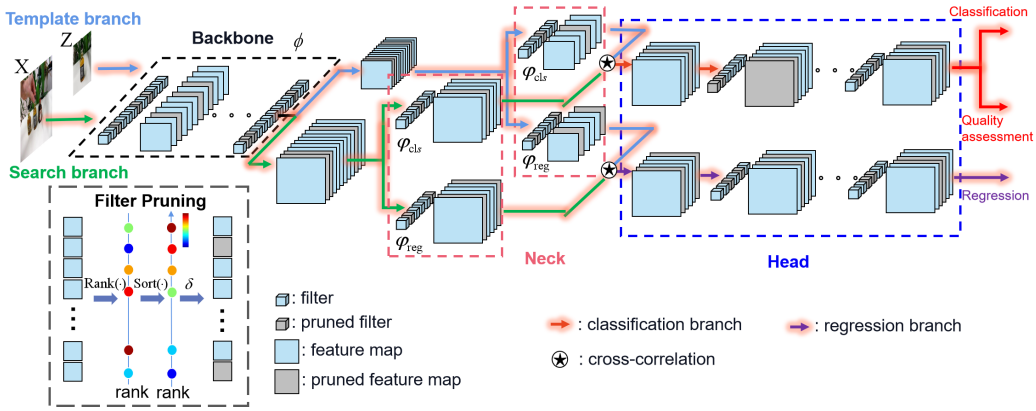


Figure 2: Our P-SiamFC++ pipeline. The pipeline is basically the same as that of SiamFC++. The difference lies in the pruned filters and feature maps. Note that the subsequent architectures connected to the head are neglected here since no pruning is involved.

one-shot way, which eliminates the need of introducing additional constraints or retraining, thus simplifying the pruning complexity a lot. We utilize this approach in this work to achieve our goal of model compression.

3. Proposed Approach

Our P-SiamFC++ is built up by pruning SiamFC++ [32] using the rank-based filter pruning method proposed in HRank [18], with the difference that we use block-wise pruning ratios to search for optimal ratios in P-SiamFC++(v2). The details are described as follows.

3.1. P-SiamFC++ Overview

As illustrated in Fig. 2, our P-SiamFC++ consists of a template branch, a search branch and three parts: backbone, neck and head. The two branches share the same backbone (i.e., Alexnet [49]) for feature extraction, which is denoted by the mapping $\phi(\cdot)$. The template branch generates features using the tracking target patch Z (as input). The search branch generates features using the search region X. The features of the two branches are coupled with cross-correlation before being used for subsequent classification and regression tasks. The coupled features are defined as follows:

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

Electronic copy available at: https://ssrn.com/abstract=4159540

where $\star$ denotes the cross-correlation operation, and $\psi_l(\cdot), l \in \{cls, reg\}$ denotes the task-specific layer ('cls' and 'reg' are short for classification and regression, respectively). Note that the output of ψ_{cls} and ψ_{reg} are of the same size. The classification branch, with the output being $O^{cls}_{h\times w\times 2}$, is used to predict the category for each location, while the regression branch, with the output being $O^{reg}_{h\times w\times 4}$, is to compute the target bounding box at this location, where w and h are the width and height of the outputs, respectively. $O^{cls}_{h\times w\times 2}(i,j,:)$ is a 2D vector representing the foreground and background scores of the location (i,j), while $O^{reg}_{h\times w\times 4}(i,j,:)$ is a 4D vector representing the distances from the corresponding location to the four sides of the bounding box. A center-ness branch, with the output being $O^{cen}_{h\times w\times 1}$, is in parallel with the classification branch to assess classification qualities, which is finally used to reweight the classification scores. The pipeline of our PW-SiamFC++ is the same as that of SiamFC++. The difference lies in the pruned feature maps determined by filter pruning, which will be explained in detail in the following subsection.

3.2. Rank-based Filter Pruning Criterion

Denote the i-th ($i \in [1, K]$) convolutional layer C^i of the 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 denotes the number of filters in C^i, k_i is 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 output feature maps of the filters 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 generated by 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 in [18] is formulated as the following optimization problem:

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

where I denotes an input image which follows the $P(I)$ distribution, n^i_p represents the number of filters to be pruned in C^i. $\delta_{i,j} \in \{0, 1\}$ indicates whether the filter w^i_j is pruned, $\delta_{i,j} = 1$ if it is, otherwise $\delta_{i,j} = 0$. $Rank(\cdot)$ computes the rank of a feature map , which is a measure of information richness.The expectation of ranks generated by a single filter is empirically proved to be robust to the input images [18], on the ground of which Eq. (2) is approximated by

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

Electronic copy available at: https://ssrn.com/abstract=4159540

where t indexes the input images. Eq. (3) be minimized by pruning n_p^i filters with the least average ranks of feature maps.

3.3. Losses for Finetuning

We now formulate the losses for finetuning P-SiamFC++ after the pruning. Let (x_0, y_0) and (x_1, y_1) represent the left-top and right-bottom coordinates of the ground truth bounding box, and (x, y) denote the corresponding location of the point (i, j), then the regression target $\hat{t}_{(i,j)} = \{\hat{t}^k_{(i,j)}\}_{k=0}^3$ at $O^{reg}_{h\times w\times 4}(i, j, :)$ is defined by

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

The IOU loss for regression is defined as follows

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

where L_{IOU} indicates the IOU loss as in [50], $\mathbb{I}(\cdot)$ is the indicator function defined by:

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

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

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

Then the centerness loss is defined by

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

The overall loss for finetuning P-SiamFC++ then is:

$$L = L_{cls} + \lambda_1 L_{reg} + \lambda_2 L_{cen}, \tag{9}$$

where L_{cls} is the cross-entropy loss [51] for classification, λ_1 and λ_2 are predefined constants to balance the losses.

Electronic copy available at: https://ssrn.com/abstract=4159540

3.4. Rank-based Filter Pruning Schedule

The pipeline of rank-based filter pruning is as follows: First, calculate the average rank of the feature map of any filter in each layer to obtain the rank set $\{R^i\}_{i=1}^K = \{\{r_1^i, r_2^i, ..., r_{n_i}^i\}\}_{i=1}^K$. Second, each R^i is sorted in decreasing order, ending up with $\bar{R}^i = \{r_{s_1^i}^i, r_{s_2^i}^i, ..., r_{s_{n_i}^i}^i\}$, where s_j^i is the index of th j-th top value in R^i. Third, we empirically determine the number of pruned filters of each layer n_p^i in order to prune the SiamFC++ model, and then conduct filter pruning to obtain P-SiamFC++. After pruning, R^i turns to $\hat{R}^i = \{r_{s_1^i}^i, r_{s_2^i}^i, ..., r_{s_{\hat{n}_i}^i}^i\}$ in which $\hat{n}_i = n_i - n_p^i$. Finally, the filters retained are initialized with the original weights in the trained SiamFC++ model, and then the compressed model P-SiamFC++ is finetuned.

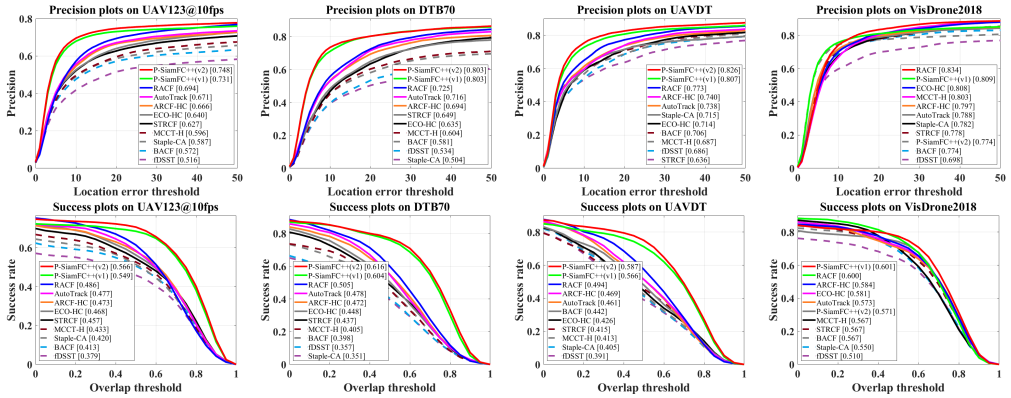
4. Experiments

In this section, the proposed tracker is exhaustively evaluated on four public challenging UAV benchmarks, i.e., UAV123@10fps [52], DTB70 [53], UAVDT [54] and Vistrone2018 [55]. UAV123@10fps is designed to investigate the impact of camera capture speed on tracking performance, which was constructed by down sampling the UAV123 benchmark [52] to 10 FPS from 30FPS. DTB70 consists of 70 UAV sequences primarily addresses the problem of severe UAV motion, but includes as well various cluttered scenes and objects with different sizes. UAVDT is mainly for vehicle tracking with various weather conditions, flying altitudes and camera views. Vistrone2018 (VisDrone2018-test-dev) is from the single object tracking challenge held in conjunction with the European conference on computer vision (ECCV2018), which focuses on evaluating tracking algorithms on drones.

4.1. Experimental setup

The evaluation experiments are conducted using MATLAB R2019a on a PC with an i9-10850K processor (3.GHz), 16GB RAM and a NVIDIA Titan X GPU. Notice that there are two settings of pruning ratios for our P-SiamFC++, corresponding to two versions of realizations which are dubbed P-SiamFC++(v1) and P-SiamFC++(v2), respectively. P-SiamFC++(v1) uses layer-wise pruning ratios and is just the implementation of our previous work [19]. P-SiamFC++(v2) is implemented here and uses block-wise pruning ratios for enhancement. Specifically, for P-SiamFC++(v1), the pruning ratios for the five convolutional layers in backbone (AlexNet) are (0.792, 0.875, 0.878, 0.870, 1.0), and for the three convolutional layers in head, for classification and regression, respectively, are

Electronic copy available at: https://ssrn.com/abstract=4159540



(a) UAV123@10fps (b) DTB70 (c) UAVDT (d) VisDrone2018

Figure 3: Overall performance of hand-crafted based trackers on (a) UAV123@10fps [52] (b) DTB70 [53](c) UAVDT [54] and (d) VisDrone2018 (VisDrone2018-test-dev) [55]. Precision and success rate for one-pass evaluation (OPE) [65] are used for evaluation. The precision at 20 pixels and area under curve (AUC) are used for ranking and marked in the precision plots and success plots respectively.

(0.898, 0.539, 0.875) and (0.887, 0.566, 0.875). The neck is not pruned. For P-SiamFC++(v2), the block-wise pruning ratios for the backbone, the neck, and the head are 0.7, 0.5, and 0.3, respectively. Other parameters for training and inferencing follow that of SiamFC++ [32]. Note that the real-time performance is defined relatively and is generalized only to platforms with computational resources equal to or more than the ones we used.

4.2. Comparison with DCF-based trackers

Fourteen state-of-the-art trackers based on hand-crafted features for comparison are: RACF [10], AutoTrack [4], ARCF-HC [13], STRCF [56], MCCT-H [57], KCC [58], ECO-HC [24], BACF [59], Staple-CA [60], CSRDCF [61], fDSST [62], KCF [23], DSST [63] and SAMF [64]. The precision and success plots on the four benchmarks are shown in Fig. 3. Some attribute-based evaluation is shown in Fig. 4. In addition, the average performance in terms of precision (PRC) and frames per second (FPS) on a single CPU is displayed in the Table 1.

Overall performance evaluation: The overall performance of P-SiamFC++ with the competing trackers on the four benchmarks is shown in Fig. 3. As can be seen, P-SiamFC++(v1) and P-SiamFC++(v2) outperform all other trackers on all four benchmarks. Specifically, on UAV123@10fps, DTB70 and UAVDT, P-SiamFC++(v1) significantly outperforms the second tracker RACF in (PRC,

Electronic copy available at: https://ssrn.com/abstract=4159540

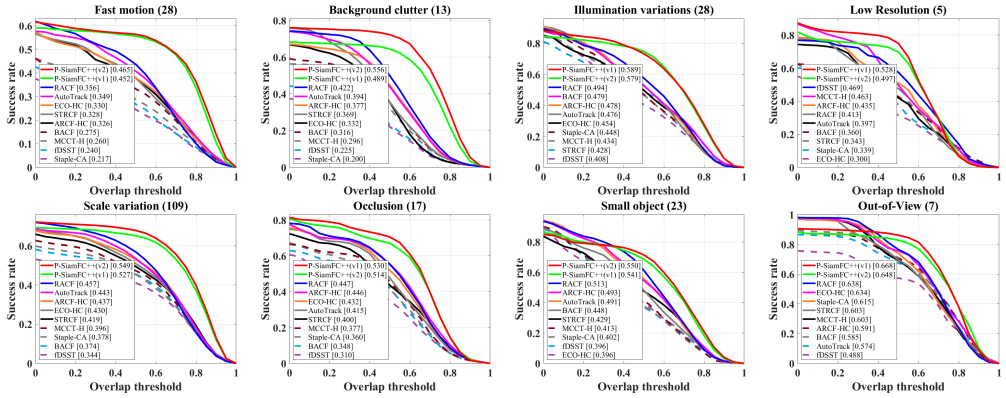
Table 1: Average precision (PRC) and speed (FPS) comparision between P-SiamFC++ and hand-crafted based trackers on UAV123@10fps, DTB70, UAVDT and VisDrone2018. All the reported FPSs are evaluated on a single CPU. Red, blue and green respectively indicate the first, second and third place. Note that P-SiamFC++ is the best real-time tracker (with a speed >30FPS) on CPU.

	KCF	fDSST	Staple-CA	BACF	ECO-HC	MCCT-H	STRCF	ARCF-HC	AutoTrack	RACF	**P-SiamFC++(v1)**	**P-SiamFC++(v2)**
PRC	53.3	60.4	64.2	65.3	68.8	66.8	67.1	71.9	**72.3**	**75.7**	**78.8**	**78.8**
FPS (CPU)	**622.5**	**193.4**	64.3	54.2	**84.5**	63.4	28.4	34.2	58.7	35.7	46.1	76.4

AUC) with gains of (3.7%, 6.3%), (7.8%, 9.9%) and (3.4%, 7.2%), respectively, while P-SiamFC++(v2) improves the (PRC, AUC) of P-SiamFC++(v1) by (1.7%, 1.7%), (0.0%, 1.2%), and (1.9%, 2.1%) on these three benchmarks, respectively. On VisDrone2018, P-SiamFC++(v2) is inferior to P-SiamFC++(v1) with gaps of (3.5%, 3.0%) in (PRC, AUC), and P-SiamFC++(v1) is inferior to the first tracker RACF in PRC with a gap of 2.5% but is the first place in AUC, although by a narrow margin of 0.1% to the second place. In terms of speed, we evaluate the average FPS on a single CPU of the competing trackers on the four benchmarks, the average FPSs along with the average PRCs are shown in Table 1. As can be seen, P-SiamFC++(v1) and P-SiamFC++(v2) achieve the same average PRC, i.e. 78.8%, and outperform all the competing trackers, and they are also the best real-time tracker (with a speed of >30FPS) on CPU, with a speed of 46.6 FPS and 76.4 FPS, respectively. Note that although P-SiamFC++(v1) significantly surpasses P-SiamFC++(v2) on VisDrone2018, P-SiamFC++(v2) has a higher pruning ratio overall achieving, therefore, a significantly faster speed, with the same average PRC as that of P-SiamFC++(v1). Given these, P-SiamFC++(v2) strikes a better balance between efficiency and precision than P-SiamFC++(v1).

Attribute-based evaluation: Our P-SiamFC++(v1) and P-SiamFC++(v2) outperform other competing DCF-based trackers in most attributes defined respectively in the four benchmarks. Examples of success plots are shown in Fig. 4. As can be seen, in the situations of fast motion and scale variation on UAV123@10fps, background clutter and occlusion on DTB70, illumination variations and small object on UAVDT, low resolution and out-of-view on VisDrone2018, P-SiamFC++(v1) and P-SiamFC++(v2) demonstrate significant improvements over other trackers because the effectiveness of feature representation with deep learning, justifying the effectiveness of developing lightweight deeper trackers for UAV tracking. For example, P-SiamFC++(v1) and P-SiamFC++(v2) significantly surpass the second place RACF on fast motion subset of UAV123@10fps by a gap of 9.6% and 10.9%, on background clutter subset of DTB70 by a gap of 6.7% and 13.4%, respectively. We can also observe that in most these attribute subsets P-

Electronic copy available at: https://ssrn.com/abstract=4159540



(a) UAV123@10fps (b) DTB70 (c) UAVDT (d) VisDrone2018

Figure 4: Attribute-based comparison on fast motion, scale variation, background clutter, occlusion, illumination variations, small object, low resolution, and out-of-view.

SiamFC++(v2) outperforms P-SiamFC++(v1) except on the subsets of the VisDrone2018. This suggests that many filters that may be helpful for tracking on the VisDrone2018 have been pruned because of their smaller rank information, but pruning them increases the AUC on the rest benchmarks. Therefore, there is room for improvement of the rank-based filter pruning criterion considering it is kind of data dependent as a criterion. We will explore better pruning criterion in our future work.

Qualitative evaluation: Some qualitative tracking results of our P-SiamFC++ and four top CPU-based trackers are shown in Fig. 5. From the four benchmarks, eight video sequences (i.e., boat3, person10, BMX3, SUP2, S1201, S0304, uav0000164_00000_s, and uav0000353_00001_s, two from each benchmark) are selected for demonstration. It can be seen that the four CPU-based trackers fail to maintain robustness in these challenging examples that objects are experiencing large deformation or pose change or partial occlusion, but our P-SiamFC++ performs much better and is visually more satisfying by virtue of the deep representation learning. Specifically, all trackers except the proposed P-SiamFC++(v1) and P-SiamFC++(v2) fail in tracking the person in the sequence persion10 and the car in the sequence uav0000164_00000_s; only ARCF-HC, P-SiamFC++(v1) and P-SiamFC++(v2) succeed in tracking the target in BMX3 but our P-SiamFC++(v1) and P-SiamFC++(v2) are more accurate; only ECO-HC, P-SiamFC++(v1) and P-SiamFC++(v2) succeed in tracking the car in S1201 but also our P-SiamFC++(v1) and P-SiamFC++(v2) are more accurate; the target in boat3, SUP2, S0304, and

Electronic copy available at: https://ssrn.com/abstract=4159540

Table 2: Precision (PRC) and speed (FPS) comparison between P-SiamFC++ and deep-based trackers on UAVDT [54]. All the reported FPSs are evaluated on a single GPU. Red, blue and green indicate the first, second and third place.

	SiamR-CNN	D3S	PrDimp18	KYS	SiamGAT	LightTrank	TransT	HiFT	SOAT	AutoMatch	**P-SiamFC++** (v1)	**P-SiamFC++** (v2)
PRC	66.5	72.2	73.2	79.8	76.4	80.4	**82.6**	65.2	**82.1**	**82.1**	**80.7**	**82.6**
FPS (GPU)	7.2	44.2	48.1	30.0	74.2	84.1	41.8	**134.1**	29.2	50.0	**258.8**	**298.5**

uav0000353_00001_s are successfully tracked by all trackers but, likewise, our P-SiamFC++(v1) and P-SiamFC++(v2) are more accurate. This suggests that developing more efficient deep trackers for UAV tracking might be more effective in terms of improving tracking precision.

4.3. Comparison with deep-based trackers

The proposed P-SiamFC++ is also compared with thirteen state-of-the-art deep trackers on the UAVDT dataset, including AutoMatch [66], SOAT [67], HiFT [5], TransT [68], LightTrack [69], SiamGAT [70], KYS [37], D3S [71], SiamR-CNN [72], PrDiMP18 [73], Atom [35], Dimp50 [36] and SiamDW [74]. The FPS along with the precision (PRC) of the trackers on UAVDT are shown in Table 2. As can be seen, both P-SiamFC++(v2) and TransT achieve the best precision of 82.6% PRC, but P-SiamFC++(v2) is above 6 times faster than TransT, i.e., 298.5 FPS vs 41.8 FPS, and 9 times and near 5 times, respectively, faster than SOAT and AutoMatch that tie for second place in precision, i.e., 82.1% PRC. Although P-SiamFC++(v1) is inferior in precision to TransT, SOAT and AutoMatch, the gaps are less than 2% and its speed is 5, 7, and 4 times faster than TransT, SOAT and AutoMatch, respectively. This justifies that our P-SiamFC++(v1) and especially P-SiamFC++(v2) can achieve a better balance between precision and efficiency (i.e., speed).

4.4. Ablation study

Impact of layer-wise pruning ratios: To see how the pruning ratios affect the precision of P-SiamFC++, we trained P-SiamFC++ with different pruning ratios, including layer-wise ratios and global ratios. In the layer-wise manner, a certain layer of SiamFC++ is pruned with a pruning ration ranging from 0.1 to 0.8 while other layers stay untouched. The same layer of the two branches in the head and neck is pruned with the same ratio for simplicity. In the global manner, each convolutional layer in backbone, neck and head is pruned with the a global ratio that ranges also from 0.1 to 0.8. The precisions (PRCs) of P-SiamFC++ with different pruning ratios are shown in Table 3. As can be seen, in the layer-wise manner, the

Electronic copy available at: https://ssrn.com/abstract=4159540

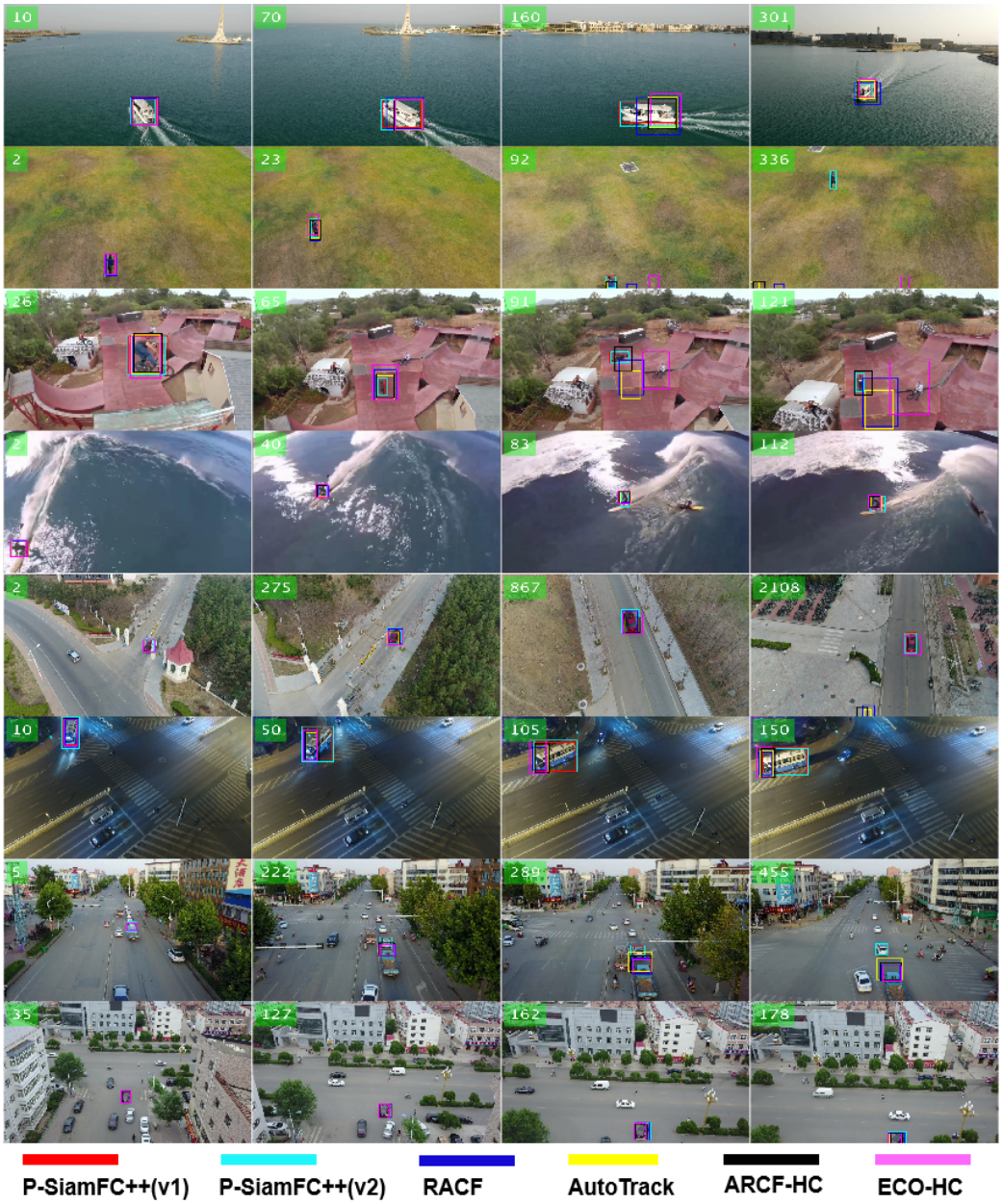


Figure 5: Qualitative evaluation on 8 video sequences from, respectively, UAV123@10fps, DTB70 and UAVDT (i.e. boat3, person10, BMX3, SUP2, S1201, S0304, uav0000164_00000_s, and uav0000353_00001_s.). The results of RACF, AutoTrack, ARCF-HC, ECO-HC, and our P-SiamFC++ are shown with different colors.

Electronic copy available at: https://ssrn.com/abstract=4159540

Table 3: Illustration of how the precision (PRC) of P-SiamFC++ on DTB70 varies with the pruning ratio that ranges from 0.1 to 0.8 in step of 0.1. 'L1 (Backbone)' represents pruning the first convolutional layer in the backbone only, and 'Backbone + Neck + Head' means all the convolutional layers in backbone, neck and head are pruned with the same pruning ratio.

Model	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
L1 (Backbone)	79.2	79.0	**79.8**	79.1	78.6	74.7	76.5	76.3
L2 (Backbone)	77.3	79.0	78.4	**81.1**	80.4	78.5	78.7	74.0
L3 (Backbone)	77.6	78.6	78.6	76.0	**79.7**	78.2	78.1	79.1
L4 (Backbone)	79.0	**81.7***	80.0	78.6	81.0	79.3	78.6	78.5
L5 (Backbone)	78.5	**80.1**	79.9	78.3	78.8	77.5	78.2	77.9
L1 (Neck)	77.2	76.7	77.4	77.9	77.0	79.7	**81.6**	79.8
L2 (Neck)	77.0	80.4	79.2	**81.5**	78.2	78.0	79.1	78.8
L1 (Head)	77.5	77.9	77.5	77.9	76.9	**78.9**	77.7	77.3
L2 (Head)	79.4	78.9	78.7	79.8	78.1	77.1	**80.6**	76.7
L3 (Head)	79.3	**81.7***	77.3	81.7	79.0	74.8	76.6	76.3
Backbone + Neck + Head	79.6	80.0	**81.0**	79.5	77.6	78.6	77.9	76.4

best precision is most often achieved when the pruning ratio is 0.2 while the best in the global manner happens when the pruning ratio is 0.3, which suggests that filter pruning is not only good for simplifying the model and raising efficiency but also benefit the precision, because it can improve the generalization ability of the model. However, using a global pruning ratio would neglect the differences between different layers, hardly leading to an optimal pruning for each layer simultaneously, whereas, optimal layer-wise pruning ratios is too cumbersome and time-consuming to determine. This motivates the block-wise pruning ratios we explore here.

Impact of block-wise pruning ratios: Exhaustively searching an optimal setting of layer-wise pruning ratios is rather time-consuming. Even though the same layer of the two branches in the head and neck was pruned with the same ratio, there are 10 layer-wise pruning ratios to be settled. And there are 8^{10} possible combinations of layer-wise pruning ratios even if each layer tries 8 pruning ratios only. To make things easier, we use block-wise pruning ratios instead, defining three blocks: backbone, neck, and head, ending up to 3 block-wise pruning ratios denoted by ρ_B, ρ_N, and ρ_H, respectively. The combinations are built up with the ranges from 0.5 to 0.8, from 0.4 to 0.7, and from 0.3 to 0.6 for the backbone, neck, and head, respectively, all with step size 0.1, resulting in 4^3 combinations in total. The average precision (PRC) on the four benchmarks and model size of P-SiamFC++ with different block-wise pruning ratios are shown in Table 4. As can be seen, when (ρ_B, ρ_N, ρ_H)=(0.7, 0.5, 0.3), it achieves the highest average precision of 78.8.% with a model size of about 3.05M, which makes the setting of pruning ratios for our tracker P-SiamFC++(v2); when (ρ_B, ρ_N, ρ_H)=(0.5, 0.4,

Electronic copy available at: https://ssrn.com/abstract=4159540

Table 4: Illustration of how the average precision of P-SiamFC++ on the four benchmarks and model size (parameters) vary with the three block-wise pruning ratios (ρ_B, ρ_N, ρ_H) that specify the pruning ratios for the backbone, the neck, and the head, respectively. The combinations of the three block-wise pruning ratios are built up with the ranges for backbone, neck, and head from 0.5 to 0.8, 0.4 to 0.7, and 0.3 to 0.6, respectively, with step size 0.1.

(ρ_B, ρ_N, ρ_H)	PRC	Parameters (M)	(ρ_B, ρ_N, ρ_H)	PRC	Parameters (M)	(ρ_B, ρ_N, ρ_H)	PRC	Parameters (M)	(ρ_B, ρ_N, ρ_H)	PRC	Parameters (M)
(0.5, 0.4, 0.3)	76.4	**1.76**	(0.6, 0.4, 0.3)	76.3	2.26	(0.7, 0.4, 0.3)	76.4	2.84	(0.8, 0.4, 0.3)	76.3	3.49
(0.5, 0.4, 0.4)	76.6	**1.98**	(0.6, 0.4, 0.4)	75.7	2.47	(0.7, 0.4, 0.4)	75.6	3.06	(0.8, 0.4, 0.4)	76.6	3.70
(0.5, 0.4, 0.5)	74.6	2.24	(0.6, 0.4, 0.5)	77.0	2.74	(0.7, 0.4, 0.5)	75.6	3.32	(0.8, 0.4, 0.5)	76.3	3.97
(0.5, 0.4, 0.6)	75.4	2.54	(0.6, 0.4, 0.6)	76.6	3.04	(0.7, 0.4, 0.6)	75.9	3.62	(0.8, 0.4, 0.6)	76.3	4.27
(0.5, 0.5, 0.3)	75.5	**1.92**	(0.6, 0.5, 0.3)	77.0	2.44	(0.7, 0.5, 0.3)	**78.8**	3.05	(0.8, 0.5, 0.3)	76.7	3.72
(0.5, 0.5, 0.4)	76.3	2.15	(0.6, 0.5, 0.4)	76.3	2.67	(0.7, 0.5, 0.4)	76.8	3.27	(0.8, 0.5, 0.4)	75.5	3.94
(0.5, 0.5, 0.5)	74.8	2.42	(0.6, 0.5, 0.5)	76.4	2.94	(0.7, 0.5, 0.5)	76.6	3.55	(0.8, 0.5, 0.5)	76.0	4.22
(0.5, 0.5, 0.6)	76.0	2.73	(0.6, 0.5, 0.6)	76.2	3.25	(0.7, 0.5, 0.6)	77.3	3.86	(0.8, 0.5, 0.6)	**78.2**	4.53
(0.5, 0.6, 0.3)	74.9	2.07	(0.6, 0.6, 0.3)	77.0	2.61	(0.7, 0.6, 0.3)	75.8	3.24	(0.8, 0.6, 0.3)	75.8	3.93
(0.5, 0.6, 0.4)	75.5	2.31	(0.6, 0.6, 0.4)	77.3	2.85	(0.7, 0.6, 0.4)	76.0	3.48	(0.8, 0.6, 0.4)	**77.8**	4.17
(0.5, 0.6, 0.5)	74.8	2.60	(0.6, 0.6, 0.5)	77.1	3.14	(0.7, 0.6, 0.5)	77.3	3.77	(0.8, 0.6, 0.5)	76.2	4.46
(0.5, 0.6, 0.6)	76.4	2.92	(0.6, 0.6, 0.6)	76.4	3.46	(0.7, 0.6, 0.6)	77.1	4.09	(0.8, 0.6, 0.6)	77.2	4.78
(0.5, 0.7, 0.3)	76.9	2.23	(0.6, 0.7, 0.3)	76.2	2.79	(0.7, 0.7, 0.3)	77.5	3.43	(0.8, 0.7, 0.3)	76.6	4.16
(0.5, 0.7, 0.4)	75.9	2.48	(0.6, 0.7, 0.4)	76.3	3.04	(0.7, 0.7, 0.4)	76.6	3.69	(0.8, 0.7, 0.4)	76.6	4.41
(0.5, 0.7, 0.5)	76.3	2.78	(0.6, 0.7, 0.5)	**78.2**	3.34	(0.7, 0.7, 0.5)	**77.8**	3.99	(0.8, 0.7, 0.5)	77.2	4.71
(0.5, 0.7, 0.6)	76.0	3.11	(0.6, 0.7, 0.6)	77.1	3.67	(0.7, 0.7, 0.6)	77.0	4.33	(0.8, 0.7, 0.6)	77.5	5.04

0.3), P-SiamFC++ has the smallest model size of about 1.76M with an average precision of 76.4%, a decrease of 2.4% and 1.29M in precision and model size, respectively, compared with P-SiamFC++(v2). We can observe that it is not always true that the large the model size the higher the average precision. For example, the largest model has 5.04M parameters happening when (ρ_B, ρ_N, ρ_H)=(0.8, 0.7, 0.6), but its average precision, i.e., 77.5%, is lower than that of P-SiamFC++(v2), with a gap of 1.3%. And the second place in precision is when (ρ_B, ρ_N, ρ_H)=(0.6, 0.7, 0.5) or (ρ_B, ρ_N, ρ_H)=(0.8, 0.5, 0.6), both of which correspond to a larger model size, i.e., 3.34M and 4.53M, than P-SiamFC++(v2). These results imply that the relation between precision and model size is much more complicated than a linear correlation, but they also suggest as before that filter pruning may improve both efficiency and accuracy if the pruning ratios are appropriately designed.

Effect of rank-based filter pruning: To see how the model size, precision and tracking speed will change when the rank-based filter pruning is applied to the baseline tracker SiamFC++, we compare the proposed P-SiamFC++(v1) and P-SiamFC++(v2) with the baseline SiamFC++ on all the four UAV benchmarks. Their comparison in terms of model size, precision (PRC) and speed (on both CPU and GPU) are shown in Table 5. As can be seen, the model size of P-SiamFC++ is reduced to 77.6% (≈7.49/9.66) and 31.6% (≈3.05/9.66) of the original for P-SiamFC++(v1) and P-SiamFC++(v2), respectively. Both the CPU and GPU speed are improved. Since the parallel computing units on our GPU far exceeds the size

Electronic copy available at: https://ssrn.com/abstract=4159540

Table 5: Comparison of model size (parameters) precision and tracking speed of our P-SiamFC++ and the baseline SiamFC++ on four UAV benchmarks. PRC is short for precision. Note that only the precisions on CPU are shown here since the difference of precision on CPU and GPU is very small.

Methods	Parameters	UAV123@10fps			DTB70			UAVDT			VisDrone2018			Avg.		
		PRC	FPS		PRC	FPS		PRC	FPS		PRC	FPS		PRC	FPS	
			CPU	GPU		CPU	GPU		CPU	GPU		CPU	GPU		CPU	GPU
SiamFC++	9.66M	72.8	29.0	212.1	**80.5**	29.8	219.3	76.2	30.0	238.9	72.5	28.9	205.8	75.5	29.4	219.1
P-SiamFC++ (v1)	7.49M	73.1	45.1	236.4	80.3	45.6	238.2	80.7	48.8	258.8	**80.9**	45.0	230.5	**78.8**	46.1	241.0
P-SiamFC++ (v2)	**3.05M**	**74.8**	**67.1**	**300.8**	80.3	**64.7**	**298.0**	**82.6**	**66.3**	**298.5**	77.4	**71.4**	**294.9**	**78.8**	**76.4**	**298.1**

of both models, the average GPU speed has increased by only 11% and 36% on P-SiamFC++(v1) and P-SiamFC++(v2), respectively. But the average CPU speed has been raised from 29.4 FPS to 46.1 FPS and to 76.4FPS, with an increase of 56.8% and 160%, respectively. Remarkably, although P-SiamFC++(v1) is slightly inferior to the baseline SiamFC++ on DTB70, the precision improvement on UAVDT and VisDrone2018 is very significant, specifically, with gains of 4.5% and 8.4%, respectively. As to P-SiamFC++(v2), it has improved or maintained the precision of P-SiamFC++(v1) on all benchmarks except VisDrone2018, and has significantly improved its speed and model size. Specifically, P-SiamFC++(v2) improves the precision of P-SiamFC++(v1) by 1.7%, 1.9% on UAV123@10fps and UAVDT, respectively, decreases its model size to less than a half, i.e., from 7.49M to 3.05M, and raises the CPU and GPU speed by 30.3FPS and 57.1FPS, respectively, despite the same precision on DTB70 and a drop of 3.5% on VisDrone2018. Taken together, P-SiamFC++(v2) strikes a better balance between precision and efficiency than P-SiamFC++(v1) does, and they both improve the efficiency and precision of SiamFC++ on the whole. These results justify that the proposed method is effective for real-time UAV tracking.

5. Conclusions

In this work, we are the first to use rank-based filter pruning to narrow the gap between DCF- and DL- based trackers in UAV tracking. The proposed P-SiamFC++(v1) and P-SiamFC++(v2) achieve a remarkable balance between efficiency and precision, and demonstrates state-of-the-art performance on four public UAV benchmarks, namely, UAV123@10fps, DTB70, UAVDT and Vistrone2018. The proposed method can not only improve efficiency but, surprisingly, also improve tracking precision. In particular, the proposed P-SiamFC++(v2) reduces the

Electronic copy available at: https://ssrn.com/abstract=4159540

model size of the baseline SiamFC++ to less than one-third of the original, i.e., from 9.66M to 3.05M, and raises its average precision from 75.5% to 78.8%, having an increase of 3.3%. We believe our work will draw more attention to model compression in UAV tracking.

In this work, we only employ a ranked-based filter pruning to compress the baseline SiamFC++. Future works includes investigating other filter pruning methods, better pruning criterion, and other baseline trackers. Another research direction is to explore effective filter pruning with one global pruning ratio in view of that the determination of layer-wise or block-wise pruning ratios in this is so laborious and time-consuming.

6. acknowledgement

Thanks to the supports by Guangxi Key Laboratory of Embedded Technology and Intelligent System, Research Institute of Trustworthy Autonomous Systems, the National Natural Science Foundation of China (No. 62176170, 62066042, 61971005), the Science and Technology Department of Tibet (No. XZ202102YD0018C), the Sichuan Province Key Research and Development Project (No. 2020YJ0282), and the Guangxi Science and Technology Base and Talent Special Project (No. 2021AC9330).

References

[1] C. Fu, A. Carrio, M. A. Olivares-Mendez, R. Suarez-Fernandez, P. Campoy, Robust real-time vision-based aircraft tracking from unmanned aerial vehicles, in: IEEE International Conference on Robotics Automation, 2014.

[2] S. Lin, M. A. Garratt, A. J. Lambert, Monocular vision-based real-time target recognition and tracking for autonomously landing an uav in a cluttered shipboard environment, Autonomous Robots 41 (4) (2017) 1–21.

[3] Y. Yin, X. Wang, D. Xu, F. Liu, Robust visual detectionlearningtracking framework for autonomous aerial refueling of uavs, IEEE Transactions on Instrumentation and Measurement (2016).

[4] Y. Li, C. Fu, F. Ding, Z. Huang, G. Lu, Autotrack: Towards high-performance visual tracking for uav with automatic spatio-temporal regularization, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11923–11932.

[5] Z. Cao, C. Fu, J. Ye, B. Li, Y. Li, Hift: Hierarchical feature transformer for aerial tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15457–15466.

Electronic copy available at: https://ssrn.com/abstract=4159540

[6] C. Fu, J. Xu, F. Lin, F. Guo, Z. Zhang, Object saliency-aware dual regularized correlation filter for real-time aerial tracking, IEEE Transactions on Geoscience and Remote Sensing (2020) 12.

[7] Y. Li, C. Fu, Z. Huang, Y. Zhang, J. Pan, Keyfilter-aware real-time uav object tracking, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 193–199.

[8] F. Lin, C. Fu, Y. He, F. Guo, Q. Tang, Bicf: Learning bidirectional incongruity-aware correlation filter for efficient uav object tracking, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 2365–2371.

[9] Y. He, C. Fu, F. Lin, Y. Li, P. Lu, Towards robust visual tracking for unmanned aerial vehicle with tri-attentional correlation filters., arXiv preprint arXiv:2008.00528 (2020).

[10] S. Li, Y. Liu, Q. Zhao, Z. Feng, Learning residue-aware correlation filters and refining scale estimates with the grabcut for real-time uav tracking, arXiv preprint arXiv:2104.03114 (2021).

[11] S. Li, Q. Zhao, Z. Feng, L. Lu, Equivalence of correlation filter and convolution filter in visual tracking, ArXiv abs/2105.00158 (2021).

[12] S. Li, Q. Jiang, Q. Zhao, L. Lu, Z. Feng, Asymmetric discriminative correlation filters for visual tracking, Frontiers Inf. Technol. Electron. Eng. 21 (2020) 1467–1484.

[13] Z. Huang, C. Fu, Y. Li, F. Lin, P. Lu, Learning aberrance repressed correlation filters for real-time uav tracking, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 2891–2900.

[14] F. Li, C. Fu, F. Lin, Y. Li, P. Lu, Training-set distillation for real-time uav object tracking, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 9715–9721.

[15] Y. Li, C. Fu, F. Ding, Z. Huang, J. Pan, Augmented memory for correlation filters in real-time uav tracking., arXiv preprint arXiv:1909.10989 (2019).

[16] T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artificial Intelligence Review 53 (7) (2020) 5113–5155.

[17] H. Wang, C. Qin, Y. Zhang, Y. Fu, Emerging paradigms of neural network pruning, arXiv preprint arXiv:2103.06460 (2021).

[18] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, L. Shao, Hrank: Filter pruning using high-rank feature map, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1529–1538.

[19] X. Wang, D. Zeng, Q. Zhao, S. Li, Rank-based filter pruning for real-time uav tracking, 2022.

Electronic copy available at: https://ssrn.com/abstract=4159540

[20] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2010, pp. 2544–2550.

[21] S. Li, Y. Liu, Q. Zhao, Z. Feng, Learning residue-aware correlation filters and refining scale for real-time uav tracking, Pattern Recognit. 127 (2022) 108614.

[22] Z. Huang, C. Fu, Y. Li, F. Lin, P. Lu, Learning aberrance repressed correlation filters for real-time uav tracking, IEEE (2019).

[23] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis & Machine Intelligence 37 (3) (2015) 583–596.

[24] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg, Eco: Efficient convolution operators for tracking, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6931–6939.

[25] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in: European conference on computer vision, Springer, 2016, pp. 850–865.

[26] S. Zhang, C. Chi, Y. Yao, Z. Lei, S. Z. Li, Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 9756–9765.

[27] B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8971–8980.

[28] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 101–117.

[29] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, P. H. Torr, Fast online object tracking and segmentation: A unifying approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1328–1338.

[30] Z. Zhang, H. Peng, Deeper and wider siamese networks for real-time visual tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4591–4600.

[31] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, J. Yan, Siamrpn++: Evolution of siamese visual tracking with very deep networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4282–4291.

Electronic copy available at: https://ssrn.com/abstract=4159540

[32] Y. Xu, Z. Wang, Z. Li, Y. Yuan, G. Yu, Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 12549–12556.

[33] D. Guo, J. Wang, Y. Cui, Z. Wang, S. Chen, Siamcar: Siamese fully convolutional classification and regression for visual tracking, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 6269–6277.

[34] Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 6668–6677.

[35] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg, Atom: Accurate tracking by overlap maximization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4660–4669.

[36] G. Bhat, M. Danelljan, L. V. Gool, R. Timofte, Learning discriminative model prediction for tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6182–6191.

[37] G. Bhat, M. Danelljan, L. Van Gool, R. Timofte, Know your surroundings: Exploiting scene information for object tracking, in: European Conference on Computer Vision, Springer, 2020, pp. 205–221.

[38] C. Mayer, M. Danelljan, D. P. Paudel, L. Van Gool, Learning target candidate association to keep track of what not to track, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 13444–13454.

[39] D. Blalock, J. J. G. Ortiz, J. Frankle, J. Guttag, What is the state of neural network pruning?, arXiv preprint arXiv:2003.03033 (2020).

[40] H. Wang, C. Qin, Y. Zhang, Y. Fu, Neural pruning via growing regularization, arXiv preprint arXiv:2012.09243 (2020).

[41] H. Wang, Q. Zhang, Y. Wang, L. Yu, H. Hu, Structured pruning for efficient convnets via incremental regularization, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[42] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural networks, arXiv preprint arXiv:1506.02626 (2015).

[43] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710 (2016).

[44] H. Hu, R. Peng, Y.-W. Tai, C.-K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, arXiv preprint arXiv:1607.03250 (2016).

Electronic copy available at: https://ssrn.com/abstract=4159540

[45] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, Q. Tian, Variational convolutional neural network pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2780–2789.

[46] C. Wang, G. Zhang, R. Grosse, Picking winning tickets before training by preserving gradient flow, arXiv preprint arXiv:2002.07376 (2020).

[47] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440 (2016).

[48] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, arXiv preprint arXiv:1803.03635 (2018).

[49] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012) 1097–1105.

[50] J. Yu, Y. Jiang, et al., Unitbox: An advanced object detection network, Proceedings of the 24th ACM international conference on Multimedia (2016).

[51] Z. Zhang, M. R. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: NeurIPS, 2018.

[52] M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for uav tracking, Far East Journal of Mathematical Sciences 2 (2) (2016) 445–461.

[53] S. Li, D. Y. Yeung, Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models., in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 4140–4146.

[54] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, Q. Tian, The unmanned aerial vehicle benchmark: Object detection and tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 375–391.

[55] L. Wen, P. Zhu, D. Du, et al., Visdrone-sot2018: The vision meets drone single-object tracking challenge results, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018, pp. 469–495.

[56] F. Li, C. Tian, W. Zuo, L. Zhang, M.-H. Yang, Learning spatial-temporal regularized correlation filters for visual tracking, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4904–4913.

[57] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, H. Li, Multi-cue correlation filters for robust visual tracking, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4844–4853.

[58] C. Wang, L. Zhang, L. Xie, J. Yuan, Kernel cross-correlator, in: The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2018.

Electronic copy available at: https://ssrn.com/abstract=4159540

[59] H. K. Galoogahi, A. Fagg, S. Lucey, Learning background-aware correlation filters for visual tracking, IEEE Computer Society (2017).

[60] M. Mueller, N. Smith, B. Ghanem, Context-aware correlation filter tracking, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1387–1395.

[61] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, M. Kristan, Discriminative correlation filter with channel and spatial reliability, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4847–4856.

[62] M. Danelljan, G. Hager, F. S. Khan, M. Felsberg, Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1430–1438.

[63] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: British Machine Vision Conference, 2014.

[64] Y. Li, J. Zhu, A scale adaptive kernel correlation filter tracker with feature integration, in: European Conference on Computer Vision, 2014, pp. 254–265.

[65] Y. Wu, J. Lim, M. H. Yang, Online object tracking: A benchmark, in: Computer Vision & Pattern Recognition, 2013.

[66] Z. Zhang, Y. Liu, X. Wang, B. Li, W. Hu, Learn to match: Automatic matching network design for visual tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13339–13348.

[67] Z. Zhou, W. Pei, X. Li, H. Wang, F. Zheng, Z. He, Saliency-associated object tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9866–9875.

[68] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, H. Lu, Transformer tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8126–8135.

[69] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, H. Lu, Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15180–15189.

[70] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, C. Shen, Graph attention tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 9543–9552.

[71] A. Lukezic, J. Matas, M. Kristan, D3s – a discriminative single shot segmentation tracker, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7133–7142.

Electronic copy available at: https://ssrn.com/abstract=4159540

[72] P. Voigtlaender, J. Luiten, P. H. Torr, B. Leibe, Siam r-cnn: Visual tracking by re-detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 6578–6588.

[73] M. Danelljan, L. V. Gool, R. Timofte, Probabilistic regression for visual tracking, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7183–7192.

[74] Z. Zhang, H. Peng, Deeper and wider siamese networks for real-time visual tracking, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Electronic copy available at: https://ssrn.com/abstract=4159540