Fisher Pruning for Developing Real-Time UAV Trackers

Pengzhi Zhong
Guilin University of Technology
Wanying Wu
Guilin University of Technology
Xiaowei Dai
Sichuan University
Qijun Zhao
Sichuan University
Shuiwang Li (lishuiwang0721@163.com)
Guilin University of Technology

Research Article

Keywords: UAV tracking, Fiser pruning, filter pruning, real-time

Posted Date: May 9th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2885467/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Journal of Real-Time Image Processing on July 27th, 2023. See the published version at https://doi.org/10.1007/s11554-023-01348-x.

Noname manuscript No.
(will be inserted by the editor)

Fisher Pruning for Developing Real-Time UAV Trackers

Pengzhi Zhong[1] · **Wanying Wu**[1] · **Xiaowei Dai**[2] · **Qijun Zhao**[2] · **Shuiwang Li**[1,✉]

Received: date / Accepted: date

Abstract Unmanned aerial vehicle (UAV)-based tracking has shown large potential in various domains such as transportation, logistics, public safety, and more. However, deploying deep learning (DL)-based tracking algorithms on UAVs is challenging because of limitations in computing resources, battery capacity, and maximum load. Discriminative correlation filter (DCF)-based trackers have become a popular choice in the UAV tracking community owing to their ability to provide superior efficiency while consuming fewer resources. However, the limited representation learning ability of DCF-based trackers leads to lower precision in complex scenarios compared to DL-based methods. Filter pruning is a prevalent practice for deploying deep neural networks on edge devices with constrained resources, and it may be an effective way to solve problems encountered when deploying deep learning trackers on UAVs. However, the application of filter pruning to UAV tracking is underexplored, and a straightforward and useful pruning standard is desirable. This paper proposes using Fisher pruning to reduce the SiamFC++ model for UAV tracking, resulting in the F-SiamFC++ tracker. The proposed tracker achieves a remarkable balance between precision and efficiency, as demonstrated through exhaustive experiments on four popular UAV benchmarks: UAVDT, DTB70, UAV123@10fps, and Vistrone2018, showing state-of-the-art performance.

Keywords UAV tracking · Fiser pruning · filter pruning · real-time

✉ Corresponding author
E-mail: lishuiwang0721@163.com

1 College of Information Science and Engineering, Guilin University of Technology, Guilin 541000, China
2 College of Computer Science, Sichuan University, Chengdu 30332, China

1 Introduction

With the widespread use of UAVs, UAV-based tracking technology has become a new hot topic, attracting increasing interest in visual tracking. It has broad potential applications in fields such as navigation, agriculture, transportation, aerial photography, and emergency response [2,3,4,5]. While tracking in general scenes is already challenging, UAV tracking faces even more onerous challenges. On the one hand, the motion of UAVs causes great challenges to the accuracy of tracking algorithms, such as scale changes, motion blur, severe occlusion; on the other hand, constrained computing capabilities, demand of minimal power consumption, and constraint of battery endurance of UAV present tremendous difficulties regarding their efficacy as well [2,6,7]. The present state of technology in UAV tracking places a great emphasis on efficiency, which is why DCF-based trackers are often used instead of DL-based ones [4,5]. Despite great improvements in tracking precision for DCF-based trackers, they still do not achieve the same level of precision as the majority of state-of-the-art DL-based trackers. A DL-based UAV tracking method with superior speed and precision was proposed in [3] recently, which applies a lightweight backbone for consideration of efficiency and fuses features of shallow and deep layers for robust representation learning with a hierarchical feature transformer. Regrettably, despite attaining a striking trade-off between precision and efficiency, and accomplishing top performance in UAV tracking, this tracker is incapable of real-time tracking on a single CPU. But importantly, this indicates that an effective and lightweight DL-based tracker may be a viable alternative to DCF-based trackers, as it can balance precision and speed. Therefore, we are motivated to develop lightweight DL-based trackers for UAV tracking and apply model compression techniques to trade preci-

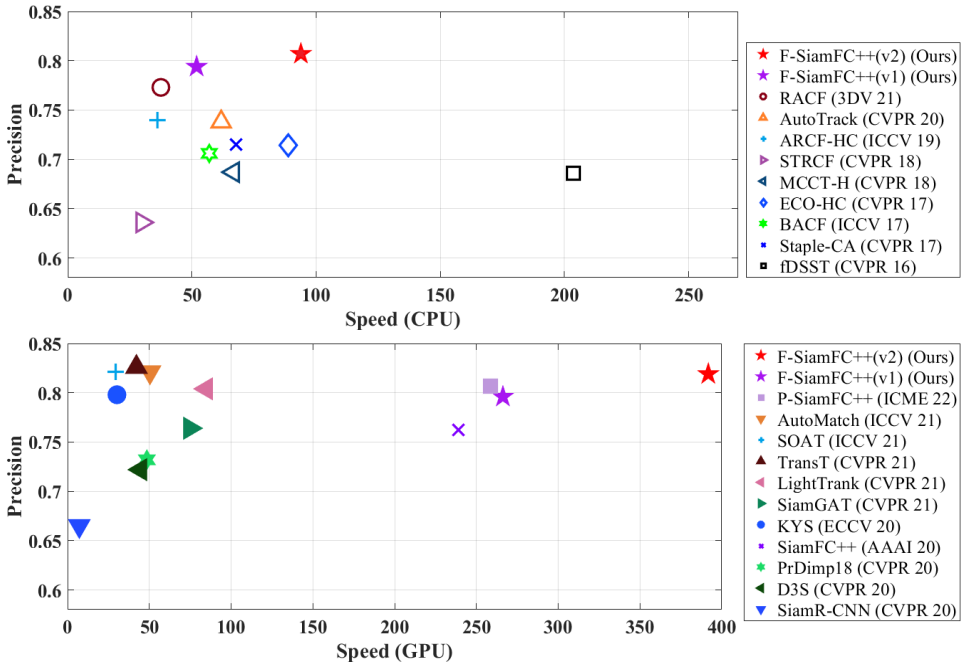


Fig. 1 Our F-SiamFC++ tracker accomplishes the optimal trade-off between precision and efficiency in the UAV benchmarks, compared with both DCF- and DL-based trackers. Taking the UAVDT [1] benchmark as an example. (a) With only a single CPU, we can perform real-time tracking with the highest precision. (b) Using a deep-learning architecture, we can get the greatest speed (efficiency).

sion for speed. Our goal is to develop a real-time DL-based tracker that achieves close precision to the original model.

Model compression aims to reduce the complexity of deep neural network models using methods like parameter pruning and knowledge distillation while maintaining accuracy. The goal is to deploy cutting-edge deep network models on resource-constrained environments, such as UAVs and embedded devices [8]. The methods extensively researched and widely used for achieving model compression involve low-rank approximation, parameter pruning, knowledge distillation, quantization, and etc [9]. It is impractical to anticipate that a universal method for model compression will effectively compress all DL-based trackers to fulfill real-time demands while preserving high precision. The choice of DL-based tracker and compression technique can significantly impact the real-time and tracking precision performance. In this study, we present utilizing Fisher pruning [10] to reduce the SiamFC++ [11] model, in order to achieve real-time drone target tracking. This pruning method does not require additional constraints or retraining of the model to achieve optimized training, making it simple and efficient. The SiamFC++ tracker is an extension of the efficient SiamFC [12] tracker, which includes a regression branch and a center-ness branch, aimed at achieving better precision and speed. Our exploration of this combination was effective, yielding an outstanding trade-off between efficiency and precision in comparison to previous DCF and DL-based trackers, as shown in Fig. 1. We expect that our research could promote the study and application of model compression in the field of UAV tracking. The following is a summary of our contributions:

- We introduce Fisher pruning as a method to narrow the performance disparity between DCF-based and DL-based trackers in UAV tracking, which is an unexplored approach.
- We present the F-SiamFC++ tracker which obtains an excellent balance between precision and efficiency by utilizing Fisher information as the pruning criterion to reduce the deep model SiamFC++.
- We evaluate the proposed approach on four established UAV datasets, consisting of UAVDT, DTB70, UAV123@10fps and Vistrone2018. The experimental results exhibit that our proposed F-SiamFC++ tracker accomplishes state-of-the-art performance.

The remaining content of this article is organized as follows. Section 2 summarizes the related work. Section 3 provides an overview of the proposed approach. Section 4 contains a description of the experiments and the results obtained. And the final section presents the conclusions drawn from this research.

2 Related Works

This paper has improved and extended our earlier work [13] and conducting a thorough analysis of fisher pruning in real-time UAV tracking. In this paper, we additionally explore

block-wise pruning ratios that enable us to obtain a better balance between tracking efficiency and precision. Thanks to this improvement, we have achieved outstanding real-time tracking performance on a single CPU, achieving an average speed of over 90FPS. Note that the previous version is represented as F-SiamFC++(v1) and the enhanced version is represented as F-SiamFC++(v2).

2.1 Visual Tracking Approachs

Tracking techniques have advanced rapidly with the emergence of modern visual trackers. There are two major categories of modern trackers: DCF-based trackers that locate targets by learning the correlation between templates and search areas, and DL-based trackers that automatically learn features using powerful neural networks. DCF-based trackers originated from the minimum output sum of squared error (MOSSE) filter [14], which is an early representative in the field of visual tracking. Since then, DCF-based tracker has continuously improved its correlation learning methods and update mechanisms within the correlation filtering framework, achieving notable progress [6]. DCF-based trackers can accomplish competitive performance while maintaining relatively high efficiency, thanks to their use of handcrafted features and the ability to be calculated in the Fourier domain. This is why they have become popular in UAV tracking. However, handcrafted features are difficult to maintain tracking stability and accuracy under complex and challenging conditions.

Recently, visual tracking has seen significant improvements in precision and robustness, due to the widespread adoption of deep learning techniques. SiamFC [12] was the first to propose formulating the visual tracking task as a generalized similarity learning problem and employing a Siamese network [15] to measure the similarity between target and search images. Siamese-based trackers can be further classified into two main types: anchor-based and anchor-free trackers [16]. In the class of anchor-based methods, SiamRPN [17] introduced a region proposal network (RPN) into Siamese networks and treated tracking as two sub-tasks that were accomplished by a classification and a regression branch. DaSiamRPN [18] incorporated an effective sampling strategy and a distractor-aware module. SiamMask [19] added a new branch to produce a pixel-wise binary mask. Recently, in order to leverage the powerful representation ability of deep features, more and more researchers have devoted to studying deeper architectures for visual tracking, such as SiamRPN++ [20] and SiamDW [21]. However, the use of these methods often results in a significant reduction in efficiency. With regards to anchor-free trackers, SiamFC++ [11] proposed a novel quality assessment branch for classification, which constitutes a simple yet effective framework for visual tracking. After that, SiamCAR [22] harnessed this framework to reengineer the anchor-free structure and integrate multiple layers of features, delivering impressive performance gains. Besides, SiamBAN [23] proposed a new strategy for generating classification labels and regression targets. Apart from Siamese-based trackers, there exist multiple DL-based trackers that extend online discriminative frameworks using deep networks for end-to-end training, e.g., ATOM [24], DiMP [25], KYS [26], and KeepTrack [27]. Unfortunately, the efficiency of these methods is too low for real-time UAV tracking.

In summary, the development of deeper architectures in recent years indeed has substantially enhanced tracking precision, but typically at the expense of efficiency. In contrast, SiamFC++ [11] is a simple yet powerful DL-based object tracking framework with a lightweight backbone and an effective quality assessment branch. Regrettably, while it exhibits remarkable GPU speed, its CPU speed appears insufficient to satisfy rigorous real-time requirements (i.e., with a speed of $\gg$ 30 FPS). Our purpose in this work is to enhance the efficiency of SiamFC++ by employing model compression methods while sustaining its precision to the greatest extent for real-time UAV tracking [11].

2.2 Methods of Filter Pruning

Pruning is a widely applied method for compressing neural networks, whose pipeline conventionally includes three stages: pretraining, pruning, and finetuning. There are four typical topics involved in the pipeline: pruning structure, pruning ratio, pruning criterion, and pruning schedule [9]. Typically, pruning structures are classified into two categories: weight pruning and filter pruning. The first method includes eliminating individual neurons or weights, which is difficult to harness for accelerating on general-purpose hardware [28]. In contrast, the second method involves deleting entire filters or channels, it is simpler due to the regular weight arrangement and can achieve significant acceleration [29]. Pruning ratios determine the percentage of weights that should be removed. Typically, there are two methods for adjusting the pruning ratio. The first is to directly specify one global ratio or numerous layer-wise ratios. The second alternative is to indirectly modify the pruning ratio, such as by using regularization-based pruning procedures, but this involves a significant amount of technical adjustment to achieve a given ratio [30]. The pruning criterion indicates which weights should be removed, and weight magnitude is the most fundamental criterion for weight pruning. Popular criteria used for filter pruning include the Frobenius norm, filter response sparsity, and selecting weights that result in the smallest amount of weight loss [9]. The pruning schedule outlines how network sparsity is gradually increased

from zero to a target value, with two available approaches [9]: (1) in a single step, namely, one-shot, followed by fine-tuning, or (2) progressively, pruning and training interlaced. While the progressive approach may yield better results with more training time available, the one-shot method is more efficient and can reduce the burden of developing sophisticated training techniques.

Overall, despite extensive research, there is still much potential for further exploration in filter pruning methods. The Fisher pruning presented in [10] recently and developed into Group Fisher pruning in [31], has proved to be an efficient and effective filter pruning approach. It is scheduled in the one-shot way and adopts Fisher information as the pruning criterion. It obviates the necessity to impose additional constraints or retraining, thereby simplifying the pruning procedure substantially. We have employed this approach in our work to achieve our objective of model compression. The difference lies in the fact that we have used global pruning ratios and block-wise pruning ratios rather than layer-wise ratios in order to simplify the pruning process. Determining layer-wise pruning ratios in the previous approach is a tedious and time-consuming task, but our approach using global and block-wise pruning ratios significantly simplifies the pruning process.

3 Proposed Method

Our F-SiamFC++ is built up by pruning SiamFC++ [11] using the Fisher pruning technique introduced in [10]. Unlike previous methods, we employed both global and block-wise pruning ratios to identify the optimal pruning ratios. The details are described as follows.

3.1 F-SiamFC++ Overview

Our proposed F-SiamFC++ tracker is composed of two branches, which is trained offline and is utilized for online prediction. The first branch holds the template and the second branch is responsible for searching. Refer to Fig. 2 for depiction. The input for the preceding is the tracking target patch Z, whereas the input for the following is the search patch X. The two branches utilize a common backbone for feature extraction, which is represented by the mapping $\phi(\cdot)$. Before being used for subsequent classification and regression tasks, the features from both branches are interacted through cross-correlation. The definition of the coupled features is given as follows:

$$f_i(Z,X) = \psi_i(\phi(Z)) \star \psi_i(\phi(X)), i \in \{cls, reg\} \tag{1}$$

where $\psi_i(\cdot)$ denotes the task-specific layer (abbreviated as 'cls' for classification and 'reg' for regression) and $\star$ denotes the cross-correlation operation. The outputs of ψ_{cls} and ψ_{reg} are equal in size, and to evaluate the accuracy of classification and ultimately reweight the classification scores, a center-ness branch is parallel to the classification branch. The following summarizes the entire training loss:

$$\begin{aligned} L(\{p_z\},\{q_z\},\{t_z\}) = \frac{1}{N_{pos}} \sum_z (L_{cls}(p_z, p_z^*) + \\ \lambda_1 I_{\{p_z^*>0\}} L_{qual}(q_z, q_z^*) + \lambda_2 I_{\{p_z^*>0\}} L_{reg}(t_z, t_z^*)), \end{aligned} \tag{2}$$

where z denotes a location on a feature map represented by its coordinates, the variables p_z, q_z, and t_z represent the predicted values, while p_z^* , q_z^* , and t_z^* represent the corresponding target label, $I_{\{\cdot\}}$ represents the indicator function, L_{cls}, L_{qual} , and L_{reg} denote the focal loss, the binary cross entropy loss and the IoU loss for classification, quality assessment, and regression, respectively. Refer to [11] for more details. Constants λ_1 and λ_2 are used to balance the losses, and $N_{pos} = \sum_z I_{\{p_z^*>0\}}$. Be aware that p_z^* is given 1 if z is thought of as a positive sample and 0 if it is thought of as a negative sample. Our F-SiamFC++ follows the same pipeline as SiamFC++, but differs in terms of the pruned filters obtained through filter pruning, and this difference will be explained with thorough explanations later on.

3.2 Fisher Pruning

Fisher information offers an approach to measure the quantity of information that an observable random variable contains about an unknown parameter of a distribution that serves as a model for the variable [32]. Formally, it represents the score variance or the expected value of the information that has been observed. Fisher Pruning is a filter pruning technique that builds its pruning criterion using Fisher information. Its purpose is to use the Fisher information [10] measure to discard feature maps that have little impact on the model's overall performance. Note that $Q_\theta(\mathbf{z}|\mathbf{I})$ represent the model, where I, $\mathbf{z}$, and θ correspond to the input, output, and parameters to be trained, respectively. Assuming the objective of training is the minimization of the subsequent loss function, without loss of generality [10]:

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{P}(\mathbf{I})}[-logQ_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})], \tag{3}$$

where $\mathbf{P}(\mathbf{I})$ is a specific data distribution with respect to which the expectation is taken. Fisher pruning uses a 2nd order approximation to approximate the corresponding change in L for a slight change $\mathbf{d}$ in the parameters:

$$L(\boldsymbol{\theta} + \mathbf{d}) \approx L(\boldsymbol{\theta}) + \mathbf{g}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{H}\mathbf{d}, \tag{4}$$

where the gradient and the Hessian matrix, respectively, are denoted by $\mathbf{g} = \bigtriangledown L(\boldsymbol{\theta})$ and $\mathbf{H} = \bigtriangledown^2 L(\boldsymbol{\theta})$. The change in

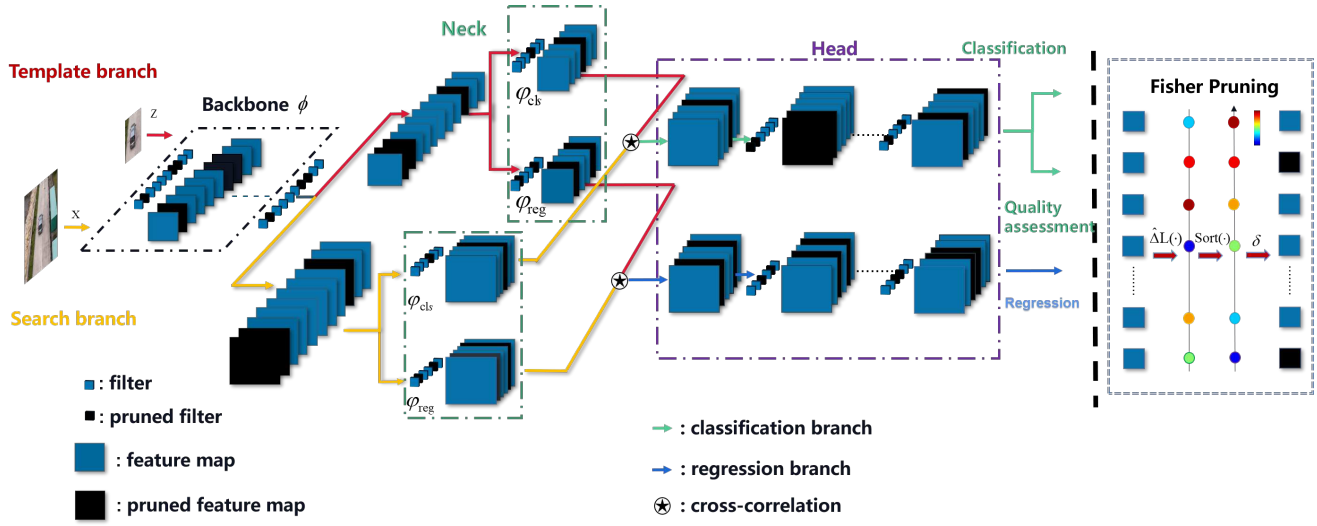


Fig. 2 An illustration of the network structure of the proposed F-SiamFC++ method, which is similar to that of SiamFC++ except for differences in the pruned feature maps and filters. It should be noted that the subsequent architectures linked to the head are not considered here since they do not involve pruning.

loss that can be expressed as follows result from dropping the kth parameter, θ_k:

$$\Delta L(\theta_k) = L(\boldsymbol{\theta} - \theta_k \mathbf{e}_k) - L(\boldsymbol{\theta}) \approx -g^T \theta_k + \frac{1}{2} H_{kk} \theta_k^2, \quad (5)$$

where $\mathbf{e}_k$ denotes the one-hot vector whose ith element has a value of 1. We would have $\bigtriangledown L(\boldsymbol{\theta}) \approx 0$ if the model had converged during training, which simplifies equation (5) to

$$L(\boldsymbol{\theta} - \theta_k \mathbf{e}_k) - L(\boldsymbol{\theta}) \approx \frac{1}{2} H_{kk} \theta_k^2. \quad (6)$$

For the diagonal entry of the Hessian matrix, it follows

$$\begin{aligned} H_{kk} &= \frac{\partial^2}{\partial \theta_k^2} \mathbb{E}_{\mathbf{P(I)}}[-log Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})] \\ &= \mathbb{E}_{\mathbf{P(I)}} \left[(\frac{\partial}{\partial \theta_k} log Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I}))^2 \right] - \mathbb{E}_{\mathbf{P(I)}} \left[\frac{\frac{\partial^2}{\partial \theta_k^2} Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})}{Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})} \right]. \end{aligned} \quad (7)$$

If $Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})$ had been trained to be close to the ideal distribution $P(\mathbf{z}|\mathbf{I})$, then

$$\begin{aligned} \mathbb{E}_{\mathbf{P(I)}} \left[\frac{\frac{\partial^2}{\partial \theta_k^2} Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})}{Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})} \right] &= \int P(\mathbf{I}) \frac{P(\mathbf{z}|\mathbf{I})}{Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})} \frac{\partial^2}{\partial \theta_k^2} Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I}) d(\mathbf{z}|\mathbf{I}) \\ &\approx \frac{\partial^2}{\partial \theta_k^2} \int P(\mathbf{I}) Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I}) d(\mathbf{z}|\mathbf{I}) = 0. \end{aligned} \quad (8)$$

Moreover, the Hessian matrix simply reduces to the Fisher information matrix if $Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})$ and $P(\mathbf{z}|\mathbf{I})$ are equivalent and $Q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{I})$ is twice differentiable with respect to θ [10]. In accordance with Fisher pruning, the change in loss resulting from pruning an individual parameter θ_k can be estimated with the help of N samples, as follows:

$$\hat{\Delta} L(\theta_k) = \frac{1}{2N} \theta_k^2 \sum_{n=1}^{N} (\frac{\partial L_n}{\partial \theta_k})^2, \quad (9)$$

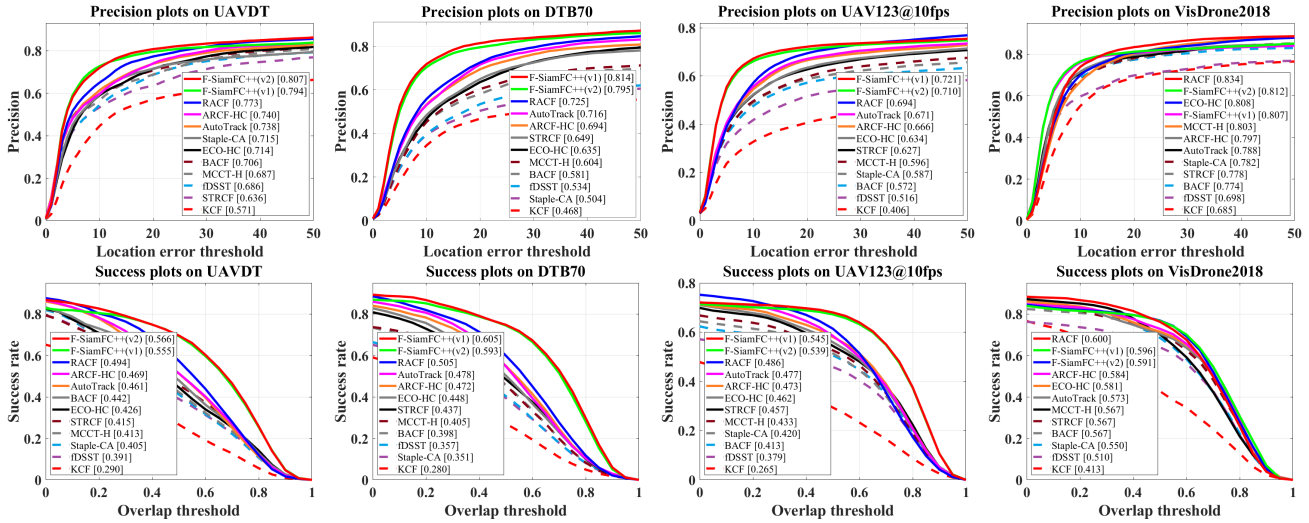
where L_n stands for the n-th sample's loss. As our goal is to speedups by pruning entire feature channels as opposed to individual parameters, we indicate the kth filter by $\boldsymbol{\theta}_k$ and denote its parameter at position (i, j) by θ_{kij}, then the resulting change of the loss due to the kth filter is removed is

$$\hat{\Delta} L(\boldsymbol{\theta}_k) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{ij} (\theta_{kij} \frac{\partial L_n}{\partial \theta_{kij}})^2, \quad (10)$$

which in Fisher pruning should be minimized.

3.3 Fisher Pruning Schedule

Let a group of 3-D filters represents the i-th ($i \in [1, K]$) convolutional layer C^i of the SiamFC++. Let n_i be defined as the number of filters in C^i, k_i be defined as the kernel size, and the j-th filter be $w_j^i \in \mathbb{R}^{n_{i-1} \times k_i \times k_i}$. And then the set of 3-D filters is $W_{C^i} = \{w_1^i, w_2^i, ..., w_m^i\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i}$. The procedure of Fisher pruning can be described as follows in detail. First, a Fisher information set $\{F^i\}_{i=1}^K = \{\{f_1^i, f_2^i, ..., f_{n_i}^i\}\}_{i=1}^K$ is created by first calculating the Fisher information of any filter for each layer. Second, each F^i is sorted from highest to lowest, resulting in $\bar{F}^i = \{f_{s_1^i}^i, f_{s_2^i}^i, ..., f_{s_{n_i}^i}^i\}$, where s_j^i is the index of the j-th highest value in F^i. Third, we develop a pruned SiamFC++



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

Fig. 3 The overall performance of hand-crafted trackers on UAVDT [1], DTB70 [33], UAV123@10fps [34], and VisDrone2018 [35] datasets are presented from left to right, evaluated by precision and success rate in one-pass evaluation (OPE). The ranking of trackers is determined based on the precision at 20 pixels and the area under curve (AUC), which are represented in the precision and success plots, respectively.

model, denoted by F-SiamFC++, by experimentally determining the number of pruned filters of each layer n_p^i in accordance with a global pruning ratio or block-wise ratios. After pruning, F^i becomes $\hat{F}^i = \{f^i_{s^i_1}, f^i_{s^i_2}, ..., f^i_{s^i_{\hat{n}_i}}\}$, where $\hat{n}_i = n_i - n_p^i$, n_p^i specifies the number of filters to remove in C^i. In the final step, trained SiamFC++ model's original weights are used to initialize the filters kept. The resulting compressed model, F-SiamFC++, is then fine-tuned to adjust its parameters for better performance.

4 Experiments

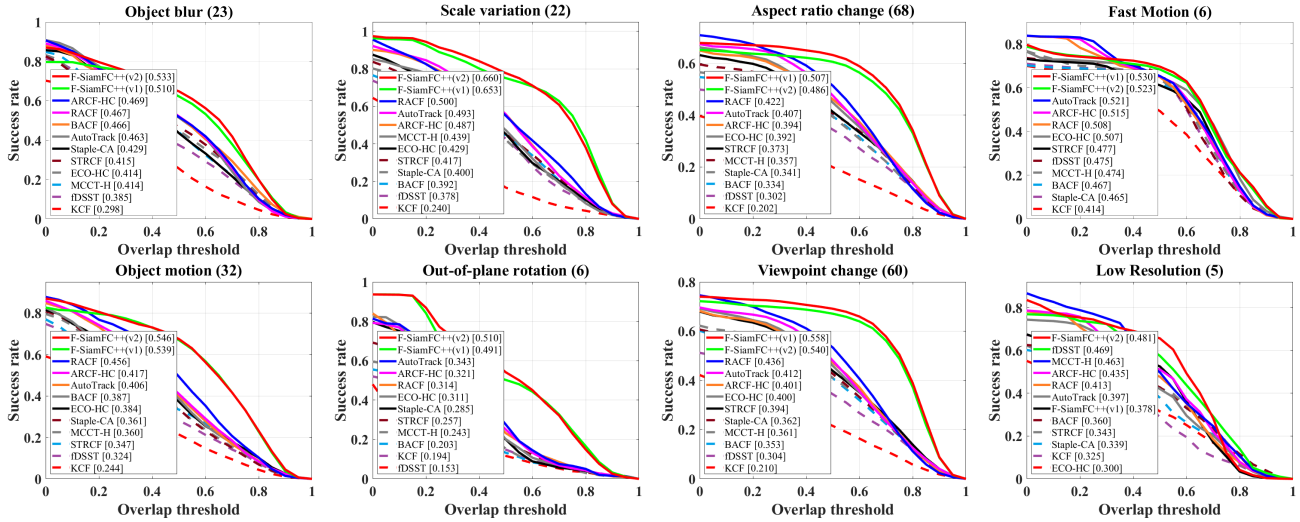
To validate the excellent performance of our proposed F-SiamFC++ tracker, we conducted a comprehensive evaluation on four challenging UAV tracking benchmarks, i.e., UAVDT [1], DTB70 [33], UAV123@10fps [34] and Vistrone2018 [35]. UAVDT is mainly designed for tracking vehicles under various weather conditions, flying altitudes, and camera perspectives. DTB70 consists of 70 sequences captured by drones, containing cluttered scenes and objects of various sizes. It is designed to evaluate the robustness of tracking algorithms in complex scenarios. UAV123@10fps is constructed by sampling the UAV123 [34] benchmark from original 30FPS to 10FPS, and Its purpose is to investigate the effect of camera capture rate on tracking performance. The Vistrone2018 dataset is derived from a single-object tracking challenge that was conducted in conjunction with the European Conference on Computer Vision (ECCV2018) and focused on evaluating drone tracking algorithms.

4.1 Experimental environment

We conducted all evaluation experiments on a PC equipped with an Intel Core i9-10850K processor (3.6GHz), an NVIDIA TitanX GPU, and 16GB RAM. Note that there are two pruning ratio settings for our F-SiamFC++, which correspond to two different versions of the implementation denoted by F-SiamFC++(v1) and F-SiamFC++(v2), respectively. F-SiamFC++(v1) employs a global pruning ratio and is the realization in our previous work [13]. F-SiamFC++(v2) is implemented here and uses block-wise pruning ratios for enhancement. Specifically, F-SiamFC++(v1) utilizes a global pruning ratio of 0.2. In F-SiamFC++(v2), three blocks (backbone, neck, and head) need to be pruned, with pruning ratios of 0.5, 0.6, and 0.4, respectively. Other parameters for training and inference follow SiamFC++ [11]. It is worth noting that the real-time performance discussed in this paper is relatively defined and only applicable to platforms with equal or greater computing resources than those we used. This means that if lower computing resources are used, the performance may be lower than our experimental results.

4.2 Comparison with DCF-based trackers

We compared our proposed tracker to ten state-of-the-art trackers based on hand-crafted features: KCF [36], fDSST [37], BACF [39], Staple-CA [38], ECO-HC [40], STRCF [42], MCCT-H [41], AutoTrack [4], ARCF-HC [5], RACF [7]. Fig. 3 illustrates the precision and success rate on four benchmark tests, namely UAVDT, DTB70,



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

Fig. 4 Comparison based on attributes such as object blur, object motion, scale variation, out-of-plane rotation, aspect ratio change, viewpoint change, fast motion, and low resolution.

Table 1 The average precision and speed (FPS) of F-SiamFC++ and hand-crafted based trackers were compared on the UAVDT [1], DTB70 [33], UAV123@10fps [34], and VisDrone2018 [35] datasets. All FPS values reported were evaluated on a single CPU. Note that F-SiamFC++ is the top real-time tracker (with a speed >30FPS) on a single CPU.

	KCF[36] TPAMI 15	fDSST[37] CVPR 16	Staple-CA[38] CVPR 17	BACF[39] ICCV 17	ECO-HC[40] CVPR 17	MCCT-H[41] CVPR 18	STRCF[42] CVPR 18	ARCF-HC[5] ICCV 19	AutoTrack[4] CVPR 20	RACF[7] PR 22	**F-SiamFC++(v1) Ours**	**F-SiamFC++(v2) Ours**
Precision	53.3	60.4	64.2	65.3	68.8	66.8	67.1	71.9	72.3	**75.7**	**78.4**	**78.1**
FPS (CPU)	**655.6**	**203.6**	67.7	57.0	88.9	66.7	29.9	36.0	61.8	37.5	51.9	**93.9**

UAV123@10fps, and VisDrone2018, arranged from left to right. Fig. 4 shows the evaluation results of partial attributes on the corresponding benchmark tests. In addition, Table. 1 presents the average performance calculated in terms of frames per second (FPS) and precision (PRC) on a single CPU.

Overall performance evaluation: The overall performance of F-SiamFC++ compared to other trackers on the four benchmarks is illustrated in Fig. 3. It can be observed that F-SiamFC++(v1) and F-SiamFC++(v2) exhibit better performance than all other trackers on all four benchmarks, except for the VisDrone2018. In particular, in UAVDT, DTB70, and UAV123@10fps benchmarks, F-SiamFC++(v1) demonstrates a notable superiority over the second-best tracker RACF by a significant margin in terms of (PRC, AUC), with improvements of (2.1%, 6.1%), (8.9%, 10.0%), and (2.7%, 5.9%), respectively. Meanwhile, F-SiamFC++(v2) is superior to RACF with gains of (3.4%, 7.2%), (6.9%, 8.8%), and (1.6%, 5.3%) on the same three benchmarks, respectively. On VisDrone2018, although F-SiamFC++(v1) and F-SiamFC++(v2) exhibit inferiority to the first-ranked tracer RACF in terms of (PRC, AUC), with respective gaps of (2.7%, 0.4%) and (2.2%, 0.9%), they surpass all other trackers. Note that averagely the gaps on VisDron2018 are much smaller than those on the other three benchmarks. Although the more efficient version F-SiamFC++(v2) is inferior to F-SiamFC++(v1) in terms of (PRC, AUC) on DTB70 and UAV123@10fps, with respective margins of (2.0%, 1.2%) and (1.0%, 0.6%), F-SiamFC++(v2) shows better performance on UAVDT and VisDrone2018, resulting in an average precision difference between this two version of only 0.3% on the four benchmarks. In terms of speed, we assess the average FPS on a single CPU for the competing trackers across the four benchmarks.Table 1 displays the average FPS and average precision rates (PRCs) of the competing trackers on a single CPU. As can be seen, F-SiamFC++(v1) and F-SiamFC++(v2) outperform all the other competing trackers in precision, with average PRCs of 78.4% and 78.1%, respectively. They are also the best real-time trackers (with a speed of >30FPS) on a single CPU, achieving speeds of 51.9 FPS and 93.9 FPS, respectively. Note that although F-SiamFC++(v1) has a slight advantage in average precision over F-SiamFC++(v2), F-SiamFC++(v2) achieves a higher average pruning ratio and faster speed, close to 1.81 times that of F-SiamFC++(v1). In summary, F-SiamFC++(v2) achieves a better balance between efficiency and precision compared to F-SiamFC++(v1).

Attribute-based evaluation: In the four benchmarks, our F-SiamFC++(v1) and F-SiamFC++(v2) are superior

Table 2 This table presents a comparison between F-SiamFC++ and other deep-based trackers on UAVDT [1] in terms of precision and speed (FPS). All FPS values reported are evaluated on a single GPU, where the first, second, and third place are indicated by Red, blue and green colors, respectively.

	SiamR-CNN[43] CVPR 20	D3S[44] CVPR 20	PrDimp18[45] CVPR 20	KYS[26] ECCV 20	SiamGAT[46] CVPR 21	LightTrank[47] CVPR 21	TransT[48] CVPR 21	HiFT[3] ICCV 21	SOAT[49] ICCV21	AutoMatch[50] ICCV21	P-SiamFC++[51] ICME 2022	**F-SiamFC++(v1) Ours**	**F-SiamFC++(v2) Ours**
Precision	66.5	72.2	73.2	79.8	76.4	80.4	**82.6**	65.2	**82.1**	**82.1**	80.7	79.6	**81.9**
FPS (GPU)	7.2	44.6	48.5	30.2	74.8	84.8	42.1	135.3	29.4	50.4	**258.8**	**266.2**	**391.8**

to other DCF-based trackers in the majority of the defined attributes. Fig. 4 illustrates examples of success plots. As can be seen, in the situations of object blur and object motion on UAVDT, scale variation and out-of-plane rotation on DTB70, aspect ratio change and viewpoint change on UAV123@10fps, and fast motion and low resolution on VisDrone2018, significant improvements have been demonstrated by F-SiamFC++(v1) and F-SiamFC++(v2) over other trackers, which can be attributed to the effectiveness of feature representation achieved through deep learning. For example, F-SiamFC++(v1) and F-SiamFC++(v2) significantly surpass the second-ranked tracker RACF on the scale variation subset of DTB70 by a gap of 15.3% and 15.9%, on the object motion subset of UAVDT by a gap of 8.3% and 9.0%, respectively. This justifies the effectiveness of developing lightweight deeper trackers for UAV tracking. Meanwhile we can observe that F-SiamFC++(v2) performs better than F-SiamFC++(v1) in the cases of scale variation and out-of-plane rotation on DTB70, object blur/motion on UAVDT, and low resolution on VisDrone2018, for instance, although the former contains more network parameter. This supports that block-wise pruning ratios can provide a better pruned network architecture than a global pruning ratio does so that we are able to achieve a more efficient compressed model when seeking a certain level of tracking precision. And thanks to the relatively small number of pruning ratios, block-wise ratios are easier to determine than layer-wise ones, justifying the reasonability of the choice of block-wise pruning ratios to enhance our previous version of a global pruning ratio.

Qualitative evaluation: Fig. 6 shows eight qualitative tracking results of our F-SiamFC++ and four top DCF-based trackers, i.e., ECO-HC [40], ARCF-HC [5], AutoTrack [4], and RACF [7]. We selected a total of eight video sequences from the four benchmarks (two from each benchmark) including S1607, S0304, BMX5, Surfing06, person16, boat3, uav0000180_00050_s, and uav000020_00675_s for demonstration. As can be observed, the four DCF-based trackers cannot remain robustness in challenging scenarios with significant deformation, pose change, or partial occlusion, whereas our F-SiamFC++ exhibits better performance and generates visually more satisfactory results by virtue of its deep representation learning. Specifically, all the trackers except F-SiamFC++(v1) and F-SiamFC++(v2), fail to track the person in the sequence person16; only

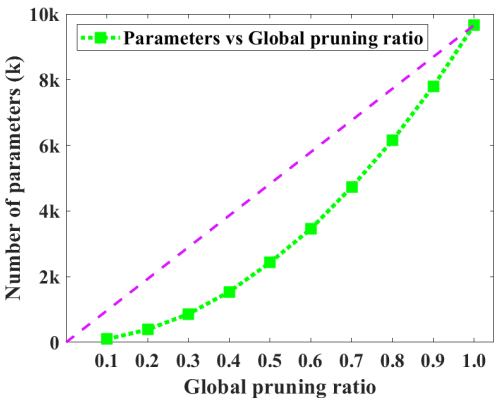


Fig. 5 Illustration of the non-linear correlation between the number of parameters of F-SiamFC++ and the global pruning ratio. Note that the purple dashed line depicts a linear correlation for comparison.

F-SiamFC++(v1), F-SiamFC++(v2), RACF and ECO-HC succeed in tracking the target in BMX5 but our F-SiamFC++(v1) and F-SiamFC++(v2) are more accurate; only F-SiamFC++(v1), F-SiamFC++(v2) and Auto Track succeed in tracking the surfer in Surfing06 but also our F-SiamFC++(v1) and F-SiamFC++(v2) are more accurate; all trackers are able to track the targets in S1607, S0304, boat3, uav0000180_00050_s, and uav0000207_00675_s successfully. However, our F-SiamFC++(v1) and F-SiamFC++(v2) still perform better in these sequences than the DCF-based trackers. These results suggest that developing lightweight DL-based trackers for UAV tracking might be a more effective means for enhancing tracking precision.

4.3 Comparison with DL-based trackers

We also compared our proposed tracker to eleven state-of-the-art DL-based trackers in the field, including SiamR-CNN [43], D3S [44], PrDiMP18 [45], KYS [26], LightTrack [47], SiamGAT [46], TransT [48], HiFT [3], SOAT [49], AutoMatch [50], and P-siamFC++ [51], on all four benchmarks. Table 2 displays the FPS and precision values obtained on the UAVDT benchmark. As can be seen, although our F-SiamFC++ (v1) and F-SiamFC++ (v2) have lower precision compared to TransT, SOAT, and AutoMatch, the differences are less than 3.0%, and the gap between F-SiamFC++(v2) and the first-ranked tracker TransT is less than 1.0%, i.e., 0.7%, specifically. Remarkably,

Fig. 6 Qualitative evaluation on 8 sequences from, respectively, UAVDT, DTB70, UAV123@10fps, and VisDrone2018 (i.e. S1607, S0304, BMX5, Surfing06, person16, boat3, uav0000180_00050_s, uav0000207_00675_s). We have used different colors to mark the tracking results of different methods.

Table 3 Illustration of how the average precision (PRC) and model size of F-SiamFC++ change as the block-wise pruning ratios (ρ_B, ρ_N, ρ_H) are adjusted for the backbone, neck, and head, respectively. We constructed a range of combinations for each ratio using the values of 0.2 to 0.5 for the backbone, 0.3 to 0.6 for the neck, and 0.4 to 0.7 for the head, with a step size of 0.1, on all four benchmarks.

(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)
(0.2, 0.3, 0.4)	76.7	5.04	(0.3, 0.3, 0.4)	**77.9**	4.33	(0.4, 0.3, 0.4)	76.8	3.67	(0.5, 0.3, 0.4)	76.8	3.11
(0.2, 0.3, 0.5)	77.0	4.71	(0.3, 0.3, 0.5)	77.1	3.99	(0.4, 0.3, 0.5)	76.9	3.34	(0.5, 0.3, 0.5)	76.0	2.78
(0.2, 0.3, 0.6)	76.7	4.41	(0.3, 0.3, 0.6)	77.6	3.69	(0.4, 0.3, 0.6)	77.2	3.04	(0.5, 0.3, 0.6)	74.2	2.48
(0.2, 0.3, 0.7)	76.5	4.16	(0.3, 0.3, 0.7)	76.5	3.43	(0.4, 0.3, 0.7)	76.5	2.79	(0.5, 0.3, 0.7)	75.9	2.23
(0.2, 0.4, 0.4)	77.1	4.78	(0.3, 0.4, 0.4)	77.4	4.09	(0.4, 0.4, 0.4)	77.3	3.46	(0.5, 0.4, 0.4)	75.0	2.92
(0.2, 0.4, 0.5)	76.9	4.46	(0.3, 0.4, 0.5)	77.0	3.77	(0.4, 0.4, 0.5)	76.4	3.14	(0.5, 0.4, 0.5)	76.3	2.60
(0.2, 0.4, 0.6)	77.0	4.17	(0.3, 0.4, 0.6)	77.5	3.48	(0.4, 0.4, 0.6)	77.1	2.85	(0.5, 0.4, 0.6)	**78.1**	2.31
(0.2, 0.4, 0.7)	75.9	3.93	(0.3, 0.4, 0.7)	**78.8**	3.24	(0.4, 0.4, 0.7)	76.5	2.61	(0.5, 0.4, 0.7)	74.8	2.07
(0.2, 0.5, 0.4)	76.2	4.53	(0.3, 0.5, 0.4)	77.4	3.86	(0.4, 0.5, 0.4)	76.0	3.25	(0.5, 0.5, 0.4)	76.9	2.73
(0.2, 0.5, 0.5)	76.1	4.22	(0.3, 0.5, 0.5)	76.6	3.55	(0.4, 0.5, 0.5)	75.7	2.94	(0.5, 0.5, 0.5)	77.6	2.42
(0.2, 0.5, 0.6)	77.1	3.94	(0.3, 0.5, 0.6)	77.3	3.27	(0.4, 0.5, 0.6)	75.5	2.67	(0.5, 0.5, 0.6)	77.1	2.15
(0.2, 0.5, 0.7)	77.3	3.72	(0.3, 0.5, 0.7)	75.8	3.05	(0.4, 0.5, 0.7)	76.4	2.44	(0.5, 0.5, 0.7)	77.1	**1.92**
(0.2, 0.6, 0.4)	76.0	4.27	(0.3, 0.6, 0.4)	76.6	3.62	(0.4, 0.6, 0.4)	76.9	3.04	(0.5, 0.6, 0.4)	76.2	2.54
(0.2, 0.6, 0.5)	76.4	3.97	(0.3, 0.6, 0.5)	77.2	3.32	(0.4, 0.6, 0.5)	77.4	2.74	(0.5, 0.6, 0.5)	76.6	2.24
(0.2, 0.6, 0.6)	76.6	3.70	(0.3, 0.6, 0.6)	76.8	3.06	(0.4, 0.6, 0.6)	77.0	2.47	(0.5, 0.6, 0.6)	75.8	**1.98**
(0.2, 0.6, 0.7)	77.1	3.49	(0.3, 0.6, 0.7)	77.2	2.84	(0.4, 0.6, 0.7)	76.1	2.26	(0.5, 0.6, 0.7)	74.0	**1.76**

SiamFC++(v1) is 5, 8, and 4 times faster than TransT, SOAT, and AutoMatch, respectively. And the enhanced version SiamFC++(v2) is more efficient and its GPU speed is close to 1.5 times of SiamFC++(v1), with an increase of 2.3% precision on UAVDT. These results indicate that our F-SiamFC++, and in particular F-SiamFC++(v2), offers a superior trade-off between precision and efficiency (i.e., speed), providing an inspiring solution alternative to DCF-based trackers for real-time UAV tracking.

4.4 Ablation study

Impact of layer-wise and global pruning ratios: We trained F-SiamFC++ with both layer-wise and global pruning ratios to investigate their impact on the model's precision. In layer-wise pruning, we prune a specific layer of SiamFC++ with a pruning ratio setting from 0.1 to 0.8, while keeping other layers fixed. In global pruning, we prune each convolutional layer in the backbone, neck, and head with a global ratio ranging from 0.1 to 0.8. The precisions of F-SiamFC++ on DTB70 benchmark with different settings of pruning ratios are shown in Table 4. Note that the larger pruning ratios are, the greater the number of filters that will be pruned. It is also worthy of note that it is not linear that correlation between the number of model parameters and even the global pruning ratio. Because the actual pruning ratio of each convolutional layer depends as well on the target pruning ratio of its previous layer. This non-linear correlation can be easily seen from Fig. 5, which plots how F-SiamFC++'s number of parameters varies with the global pruning ratio. As we can see, in the layer-wise manner the best precision is reached when the pruning ratio is smaller than 0.7, but the distribution of the best pruning ratios for each layer does not provide any guidance on how to find better layer-wise pruning ratios. In other words, finding good layer-wise pruning ratios is laborious and time-consuming in view of the huge possibility of combinations. In the global manner, the highest precision is achieved with a pruning ratio of 0.2. However, if the global pruning ratio exceeds 0.7, there is a significant drop in precision. Last but not least, this result implies that filter pruning is not only beneficial for simplifying the model and improving efficiency but also for increasing precision, as it may enhance the model's generalization ability if appropriate pruning ratios are chosen. However, applying a global pruning ratio would overlook the variations among different layers, which makes it difficult to achieve optimal pruning effects for each layer simultaneously. Meanwhile, determining optimal layer-wise pruning ratios is too cumbersome and time-consuming, which is why we explore block-wise pruning ratios to overcome this problem in this work.

Table 4 The accuracy (PRC) of F-SiamFC++ on DTB70 varies when the pruning ratio goes from 0.1 to 0.8 in step of 0.1. 'L1 (Backbone)' means that only the first convolutional layer is pruned in the backbone, and 'Backbone + Neck + Head' means all the convolutional layers are pruned with the same pruning ratio in all the backbone, neck, and head.

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 (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

Impact of block-wise pruning ratios: The process of finding the best layer-wise pruning ratios through exhaust-

Table 5 Comparison of model size (parameters), multiply-accumulates (MACs), precision (PRC), and tracking speed between our F-SiamFC++ and the baseline method SiamFC++ on four UAV benchmarks. Note that only the precision on CPU is shown here since the difference between the precision on CPU and on GPU is very small.

Methods	Parameters	MACs	UAVDT			DTB70			UAV123@10fps			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	297.98G	76.2	36.9	243.8	80.5	36.1	232.6	**72.8**	36.2	226.5	72.5	37.0	211.0	75.5	36.5	228.4
F-SiamFC++ (v1)	7.73M	193.58G	79.4	52.4	266.2	**81.4**	50.5	254.3	72.1	53.0	259.0	80.7	51.5	251.5	**78.4**	51.9	257.8
F-SiamFC++(v2)	2.31M	80.53G	**80.7**	**93.6**	**391.8**	79.5	**92.1**	**389.4**	71.0	**94.1**	**391.3**	**81.2**	**95.7**	**385.5**	78.1	**93.9**	**389.5**

ing search methods is a time-consuming endeavor. Even if the same pruning ratios are applied to corresponding layers of the head and neck branches, there are still 10 layer-wise pruning ratios that require determination. Therefore, we utilize block-wise pruning ratios to simplify the process, dividing the model into three blocks: backbone, neck, and head, and defining a separate pruning ratio for each block denoted as ρ_{B}, ρ_{N}, and ρ_{H}, respectively. The combinations consist of the following ranges: 0.2 to 0.5 for the backbone, 0.3 to 0.6 for the neck, and 0.4 to 0.7 for the head, all with step size of 0.1. As a result, the total number of combinations is reduced from 8^{10} to 4^3, a manageable amount given the time and computation resources we have. Table 3 shows the average precision (PRC) on the four benchmarks and model size of F-SiamFC++ with different block-wise pruning ratios. From the table, it is evident that the model achieves the highest average precision of 78.8 with a model size of around 3.24M when (ρ_{B}, ρ_{N}, ρ_{H}) is set to (0.3, 0.4, 0.7). With (ρ_B, ρ_N, ρ_H) set to (0.5, 0.6, 0.7), F-SiamFC++ has the smallest size of around 1.76M, but it only achieves an average precision of 74.0%, significantly lower than the highest average precision, i.e., with a gap of 4.8%. The second highest precision of 78.1 is achieved by setting it to (0.5, 0.4, 0.6), with a model size of around 2.31M, which makes the setting of pruning ratios for our tracker F-SiamFC++(v2) considering its better trade-off between precision and efficiency. We can also observe that no simple correlation between model size and average precision. For example, the largest model has 5.04M parameters when (ρ_B, ρ_N, ρ_H) is set to (0.2, 0.3, 0.4), but its average precision, i.e., 76.7%, is lower than that of F-SiamFC++(v2), with a gap of 1.4%. These results also reinforce the idea that Fisher pruning can enhance both efficiency and accuracy if pruning ratios are properly determined, as previously suggested.

Effect of fisher pruning: We compared the proposed F-SiamFC++(v1) and F-SiamFC++(v2) with the baseline tracker SiamFC++ on all four UAV benchmarks. Our aim was to investigate the impact of applying Fisher filter pruning on the model size, multiply-accumulates (MACs), precision, and tracking speed of the baseline SiamFC++. Their comparison in terms of model size, MACs, precision (PRC), and speed (on both CPU and GPU) are shown in Table 5. We can observe that, the model size of F-SiamFC++(v1) and F-SiamFC++(v2) are reduced to 80.0% (≈7.73/9.66) and 23.9% (≈2.31/9.66) of the original, respectively. Significant decreases of MACs can also be observed, from SiamFC++'s 297.98G to F-SiamFC++(v1)'s 193.58G and F-SiamFC++(v2)'s 80.53G. Both CPU and GPU speeds are improved. Due to the parallel computing units on our GPU significantly exceed the size of these models, the average GPU speeds grow by just 12.8% and 70.5% on F-SiamFC++(v1) and F-SiamFC++(v2), respectively. However, their average CPU speeds increase, respectively, by 42.2% to 51.9 FPS and by 157.2% to 93.9 FPS from SiamFC++'s 36.5 FPS. Although F-SiamFC++(v1) performs slightly worse than the baseline SiamFC++ on UAV123@10fps with a gap of 0.7%, it achieves significant precision improvements on UAVDT and VisDrone2018 benchmarks, with gains of 4.2% and 11.3%, respectively. Regarding F-SiamFC++(v2), it improves the precision of F-SiamFC++(v1) on UAVDT and VisDron2018, meanwhile significantly enhancing its speed and reducing its model size. Specifically, F-SiamFC++(v2) improves the precision of F-SiamFC++(v1) by 1.6%, 0.6% on UAVDT and VisDrone2018, respectively, reduces its model size by nearly 70%, i.e., from 7.73M to 2.31M, and raises the CPU and GPU speed by 42.0 FPS and 131.7 FPS, respectively, despite a slight decrease of 1.0% and 1.9% on UAV123@10fps and DTB70, respectively. Overall, F-SiamFC++(v2) achieves a better trade-off between precision and efficiency compared to F-SiamFC++(v1), and both variants enhance the efficiency and precision of SiamFC++ across all benchmarks. These results justify that the proposed method is effective for real-time UAV tracking and may encourage more work on developing lightweight DL-based trackers with filter pruning for real-time UAV tracking.

5 Conclusions

In this paper, we are the first to employ Fisher pruning to reduce the gap between DCF- and DL-based trackers in UAV tracking. The proposed F-SiamFC++(v1) and F-SiamFC++(v2) strike an impressive balance between pre-

cision and efficiency while showcasing state-of-the-art performance on four widely used UAV benchmarks: UAVDT, DTB70, UAV123@10fps, and Vistrone2018. Remarkably, the proposed approach not only enhances efficiency but also surprisingly enhances tracking precision. Specifically, compared with the baseline tracker SiamFC++ which runs at 36.5 FPS on a single CPU with 75.5% precision, F-SiamFC++(v1) can run at more than 50 FPS on a single CPU with 78.4% precision, while F-SiamFC++(v2) has a CPU speed above 90 FPS with an almost equal precision of 78.1%, effectively combating the negative effects (i.e., a precision drop) with filter pruning. Our research is expected to generate more interest in model compression and DL-based trackers among the UAV tracking community..

In this research, fisher pruning was the sole method used for compressing the baseline SiamFC++. Future studies may look into alternative filter pruning methods, superior pruning criteria, and different baseline trackers. Another avenue of research would be to explore utilizing discriminative representation learning to enhance the discriminative power of the compressed model in view of the significant decrease of model parameters.

Acknowledgment

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

References

1. 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.
2. C. Fu, B. Li, F. Ding, F. Lin, G. Lu, Correlation filters for unmanned aerial vehicle-based aerial tracking: A review and experimental evaluation, 2021, pp. 2–387.
3. 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 (ICCV), 2021, pp. 15457–15466.
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. 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.
6. 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, in: 2021 International Conference on 3D Vision (3DV), 2021, pp. 1238–1248.
7. S. Li, Y. Liu, Q. Zhao, Z. Feng, Learning residue-aware correlation filters and refining scale for real-time uav tracking, Pattern Recognition 127 (2022) 108614.
8. T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artificial Intelligence Review 53 (7) (2020) 5113–5155.
9. H. Wang, C. Qin, Y. Zhang, Y. Fu, Emerging paradigms of neural network pruning, arXiv preprint arXiv:2103.06460 (2021).
10. L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster gaze prediction with dense networks and fisher pruning, arXiv preprint arXiv:1801.05787 (2018).
11. 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 (AAAI-20), Vol. 34, 2020, pp. 12549–12556.
12. 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 (ECCV), Springer, 2016, pp. 850–865.
13. W. Wu, P. Zhong, S. Li, Fisher pruning for real-time uav tracking, in: 2022 International Joint Conference on Neural Networks (IJCNN), 2022, pp. 1–7.
14. D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010) 2544–2550.
15. D. Chicco, Siamese neural networks: An overview, Artificial Neural Networks (2021) 73–94.
16. 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, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 9756–9765.
17. B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 8971–8980.
18. Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: European Conference on Computer Vision (ECCV), 2018, p. 101–117.
19. Q. Wang, L. Zhang, L. Bertinetto, W. Hu, P. H. Torr, Fast online object tracking and segmentation: A unifying approach, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1328–1338.
20. 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 (CVPR), 2019, pp. 4282–4291.
21. 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 (CVPR), 2019, pp. 4591–4600.
22. D. Guo, J. Wang, Y. Cui, Z. Wang, S. Chen, Siamcar: Siamese fully convolutional classification and regression for visual tracking, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 6268–6276.
23. 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 (CVPR), 2020, pp. 6668–6677.
24. 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 (CVPR), 2019, pp. 4660–4669.

25. 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 (ICCV), 2019, pp. 6182–6191.
26. G. Bhat, M. Danelljan, L. Van Gool, R. Timofte, Know your surroundings: Exploiting scene information for object tracking, in: European Conference on Computer Vision (ECCV), Springer, 2020, pp. 205–221.
27. 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.
28. D. Blalock, J. J. G. Ortiz, J. Frankle, J. Guttag, What is the state of neural network pruning?, arXiv preprint arXiv:2003.03033 (2020).
29. 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 (CVPR), 2020, pp. 1529–1538.
30. H. Wang, C. Qin, Y. Zhang, Y. Fu, Neural pruning via growing regularization, arXiv preprint arXiv:2012.09243 (2020).
31. L. Liu, S. Zhang, Z. Kuang, A. Zhou, J.-H. Xue, X. Wang, Y. Chen, W. Yang, Q. Liao, W. Zhang, Group fisher pruning for practical network compression, in: International Conference on Machine Learning (ICML), PMLR, 2021, pp. 7021–7032.
32. P. Zegers, Fisher information properties, Entropy 17 (7) (2015) 4918–4939.
33. 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 (AAAI-17), 2017, pp. 4140–4146.
34. M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for uav tracking, Far East Journal of Mathematical Sciences 2 (2) (2016) 445–461.
35. 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.
36. J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI) 37 (3) (2015) 583–596.
37. 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.
38. 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.
39. H. K. Galoogahi, A. Fagg, S. Lucey, Learning background-aware correlation filters for visual tracking, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1144–1152.
40. 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.
41. 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 (CVPR), 2018, pp. 4844–4853.
42. 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 (CVPR), 2018, pp. 4904–4913.
43. 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.
44. 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.
45. 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.
46. 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 (CVPR), 2021, pp. 9543–9552.
47. 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 (CVPR), 2021, pp. 15180–15189.
48. 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 (CVPR), 2021, pp. 8126–8135.
49. 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 (ICCV), 2021, pp. 9866–9875.
50. 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 (ICCV), 2021, pp. 13339–13348.
51. X. Wang, D. Zeng, Q. Zhao, S. Li, Rank-based filter pruning for real-time uav tracking, in: 2022 IEEE International Conference on Multimedia and Expo (ICME), 2022, pp. 01–06.