Learning Disentangled Representation in Pruning for Real-Time UAV Tracking

Siyu Ma MSYDDP@YEAH.NET
Guilin University of Technology, Guilin, China

Yuting Liu ALENALIU17@GMAIL.COM
JD Logistic, Beijing, China

Dan Zeng ZENGD@SUSTECH.EDU.CN
Southern University of Science and Technology, Shenzhen, China

Yaxin Liao LIAOYAXIN230827@163.COM
Xiaoyu Xu XIAOYU_XU729@163.COM
Shuiwang Li(✉) LISHUIWANG0721@163.COM
Guilin University of Technology, Guilin, China

Editors: Emtiyaz Khan and Mehmet Gonen

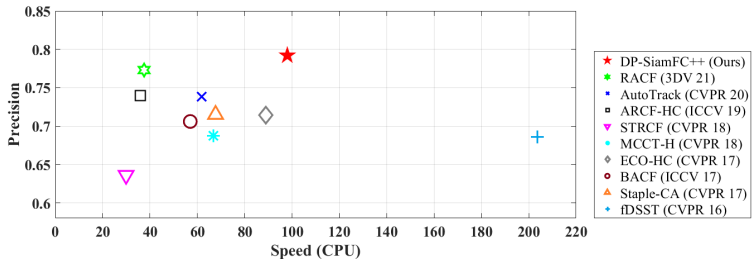
Abstract

Efficiency is a critical issue in UAV tracking because of the limitations of computing resources, battery capacity, and maximum load of unmanned aerial vehicle (UAV). However, deep learning (DL)-based trackers hardly achieve real-time tracking on a single CPU despite their high tracking precision. To the contrary, discriminative correlation filters (DCF)-based trackers have high efficiency but their precision is barely satisfactory. Despite the precision is inferior, DCF-based trackers instead of DL-based ones are widely applied in UAV tracking to trade precision for efficiency. This paper aims to improve the efficiency of the DL-based tracker SiamFC++, in particular, for UAV tracking using the model compression technique, i.e., rank-based filter pruning, which has not been well explored before. Meanwhile, to combat the potential loss of precision caused by pruning we exploit disentangled representation learning to disentangle the output feature of the backbone into two parts: the identity-related features and the identity-unrelated features. Only the identity-related features are used for subsequent classification and regression tasks to improve the effectiveness of the feature representation. With the proposed disentangled representation in pruning, we achieved higher precisions when compressing the original model SiamFC++ with a global pruning ratio of 0.5. Extensive experiments on four public UAV benchmarks, i.e., UAV123@10fps, UAVDT, DTB70, and Vistrone2018, show that the proposed tracker DP-SiamFC++ strikes a remarkable balance between efficiency and precision, and achieves state-of-the-art performance in UAV tracking.

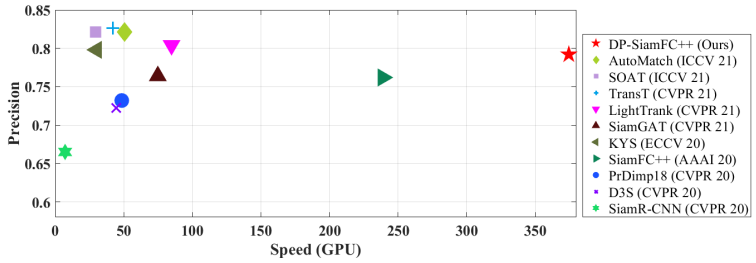
Keywords: UAV tracking; Filter pruning; Disentangled representation learning.

1. Introduction

Unmanned aerial vehicle (UAV) tracking plays an important part in various applications, such as target following, counter-terrorism, border security, product deliveries, agriculture, etc. Li et al. (2020b); Cao et al. (2021), which has attracted increasing attention recently. However, UAV tacking confronts more formidable challenges compared with visual tracking



(a) compared with popular DCF-based trackers



(b) compared with popular DL-based trackers

Figure 1: Our tracker DP-SiamFC++ can (a) perform real-time tracking with the highest precision on a single CPU, (b) achieve the best speed (efficiency) using a deep-learning architecture.

in general scenes. On the one hand, motion blur, extreme viewing angle, and scale changes are ubiquitous in UAV tracking, which pose major challenges to the *precision* of the UAV tracking algorithms; On the other hand, limited computing resources, battery capacity limitations, low power consumption requirements, and UAV's maximum load likewise pose a big challenge to its *efficiency* Li et al. (2021, 2022); Wang et al. (2022); Wu et al. (2022). As the efficiency is a fundamental issue in UAV tracking at the current technical level, discriminative correlation filters (DCF)-based trackers instead of deep learning (DL)-based ones are frequently adopted to trade precision for efficiency. Despite the tracking precision of DCF-based trackers having been improved greatly, they still cannot match up to most state-of-the-art DL-based trackers. Notwithstanding, DL-based trackers can hardly run at real-time speed on a single CPU so far, hindering their deployment on UAVs. Very recently, an efficient and effective deep tracker for UAV tracking was proposed in Cao et al. (2021), which uses a lightweight backbone attending to efficiency and utilizes a hierarchical feature transformer to fuse features of shallow and deep layers for robust representation learning. Unfortunately, this tracker has not yet reached real-time tracking on a single CPU, while striking a fair balance between efficiency and precision and demonstrating state-of-the-art UAV tracking capabilities. However, it shows that an efficient, lightweight DL-based tracker may be a viable alternative for DCF-based trackers in terms of efficiency and accuracy, which motivated our work. Hopefully, we will achieve a real-time DL-based tracker running on a single CPU without sacrificing precision of the uncompressed model.

Model compression is the most popular method utilized to deploy cutting-edge deep networks on resource-constrained and low-power edge devices without significantly affecting model accuracy. Prevalent methods include pruning, quantization, low-rank approximation,

knowledge distillation Wang et al. (2021). It is unrealistic to expect a general model compression method to compress any DL-based tracker to meet the real-time requirement and obtain a satisfactory precision at the mean time. Therefore, the selection of DL tracker and compression method is extremely important to achieve real-time yet high precision tracking performance. SiamFC++ Xu et al. (2020), based on the efficient tracker SiamFC Bertinetto et al. (2016), achieves cutting-edge precision and speed by including a regression branch and a center-ness branch. Free from the need to introduce additional constraints and retain the model, the rank-based filter pruning method Lin et al. (2020) is very straightforward and has a fairly high training efficiency. The catch, however, is that the determination of layer-wise pruning ratios is laborious and time-consuming. Using a **global pruning ratio** is an easy way out of this, which, however, may impair the precision to a considerable degree. Considering recent advances in many tasks, e.g., face anti-spoofing Zhang et al. (2020), aspect extraction Tai et al. (2021), 3D point cloud processing Xu et al. (2021), voice style transfer Yuan et al. (2021), and fine grained categorization Dang et al. (2021), are attributed to the disentangled representation learning, we exploit it to combat the loss of precision when pruning the SiamFC++ Xu et al. (2020) with a global pruning ratio. Disentangled representation learning is able to tease apart the identity-related and identity-unrelated information in the feature representation output by the backbone, providing a more effective feature representation for downstream tasks considered here. Therefore, only the identity-related feature is used for subsequent classification and regression tasks in our model. We name the proposed method DP-SiamFC++ since our tracker is obtained by learning disentangled representation in pruning the SiamFC++. Compared with the existing CPU- and DL-based trackers, DP-SiamFC++ has achieved a significant balance between efficiency and precision, and specially DP-SiamFC++ boosts the performance of the uncompressed model, i.e., SiamFC++, in terms of both efficiency and precision, as illustrated in Fig. 1. More specific, DP-SiamFC++ can run at 98.0 FPS on a single CPU and at 374.5 FPS on a single GPU with the highest and near the highest precision in CPU-based and DL-based trackers, respectively. Our contributions can be summarized as follows:

- Utilizing such model compression as rank-based filter pruning for UAV tracking has not been well explored before, our work provides a fresh perspective to boost efficiency and precision in UAV tracking.
- We propose the DP-SiamFC++ tracker that learns disentangled representation in pruning. The proposed method can compress the SiamFC++ to about 50% of its original model size and achieve 98.0 FPS on a single CPU, meanwhile maintaining and even significantly boosting precision simultaneously.
- We evaluate the proposed method DP-SiamFC++ on four public UAV benchmarks. Experimental results show that the proposed DP-SiamFC++ tracker achieves state-of-the-art performance.

2. Related Works

2.1. Visual Tracking

Modern trackers can be divided into DCF-based trackers and DL-based ones. MOSSE Bolme et al. (2010) is the first DCF tracker to use the minimum output sum of squared error filter for tracking, and it comes with other versions Li et al. (2021, 2020a). In order to achieve competitive performance with high efficiencies, DCF-based trackers usually adopt handcrafted features and are calculated in the Fourier domain. As efficiency is a fundamental issue in UAV tracking, DCF-based trackers, therefore, was popular in the UAV tracking community. However, DCF-based trackers struggle to retain robustness under adverse conditions because of the limited representation ability of handcrafted features.

Deep learning for visual tracking has been proved to be very successful in recent years, and has significantly improved the tracking performance. SiamFC Bertinetto et al. (2016) is the very early tracker where visual tracking is treated as a broad similarity-learning issue and the Siamese network is used to quantify the similarity between target and search images. Since then, many Siamese-like DL-based trackers have been developed. Recently, SiamRPN++ Li et al. (2019), SiamBAN Chen et al. (2020) and DualTFR Xie et al. (2021) use deeper architectures for the sake of further improving tracking precision. Unfortunately, their tracking efficiency has been sacrificed to a large extent. In contrast, SiamFC++'s lightweight backbone and its effective quality assessment branch for classification make it a simple yet powerful framework. However, its CPU speed seems farfetched to meet strict real-time requirements (i.e., with a speed of $\gg$ 30 FPS), in spite of its excellent GPU speed. In this paper, we aim to improve the efficiency of SiamFC++ while maintaining its precision as much as possible for real-time UAV tracking Xu et al. (2020) using model compression and disentangled representation learning.

2.2. Filter Pruning

Pruning is divided into weight pruning and filter pruning, which is a commonly used neural network compression technology. Weight pruning usually removes neurons or weights, but is difficult to achieve acceleration on general-purpose hardware Blalock et al. (2020). Filter pruning is much easier to achieve considerable speed-up since it removes entire filters or channels Lin et al. (2020). The pruning ratio is used to indicate how many weights to remove, and there are usually two ways to decide the pruning ratio(s). One is a predefined global ratio or multiple layer-wise ratios. The other is to adjust the pruning rate indirectly, such as the pruning method based on regularization. However, the second way requires a lot of engineering tweaks in order to achieve specific ratios Wang et al. (2020). The pruning criterion is used to determine which weights to prune. For filter pruning, the commonly used criteria take in Frobenius norm or sparsity of the filter response and the scaling factor of the Batch Normalization layer Wang et al. (2021). Finally, in order to clarify the change rule of network sparsity from zero to target number, namely pruning scheduling, we also provide two typical options Wang et al. (2021): (1) a single step (one-shot), then fine adjustment (2) alternate progressive pruning and training. The incremental approach may be better than the one-time approach because there is more training time, but the latter is more efficient because it eliminates the need to design complex training strategies.

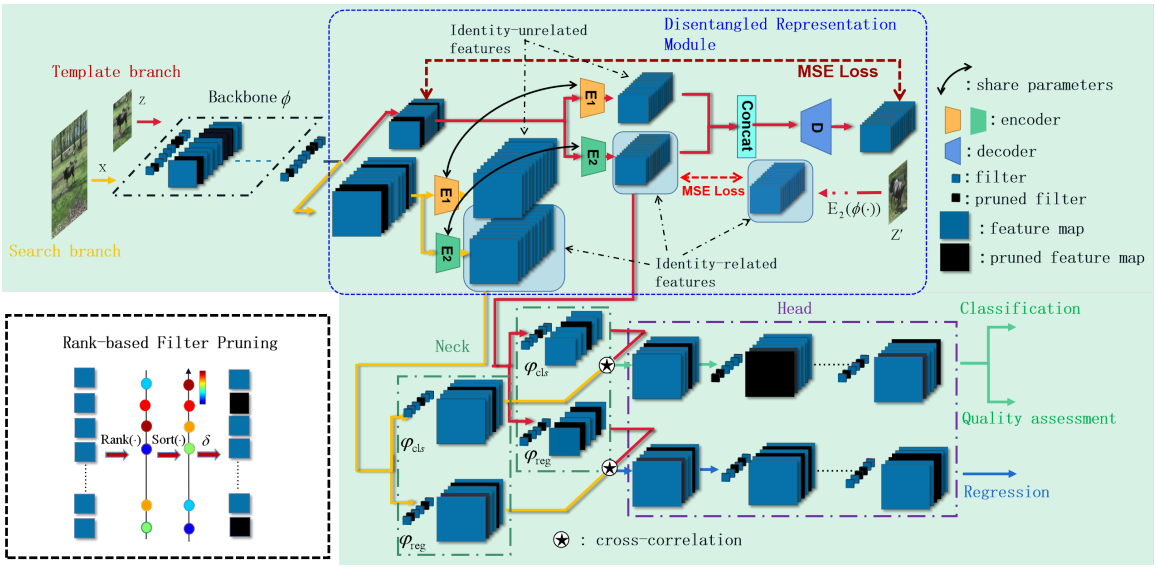


Figure 2: An illustration of the proposed DP-SiamFC++ method. Bottom left shows how rank-based filter pruning is applied to one convolutional layer.

So far, filter pruning remains an unresolved issue. An effective and efficient filter pruning approach was recently proposed in Lin et al. (2020). It takes the rank of the feature mapping of each layer as the pruning criterion and directly schedules the pruning process in the one-shot way without introducing additional constraints or retraining, which greatly simplifies the pruning process. However, it is laborious and time-consuming to determine the layering pruning rate in this method. To solve this problem, we recommend using a global pruning rate in this work. Furthermore, to combat the precision drop, we utilize disentangled representation to tease apart the identity-related and identity-unrelated information in the feature representation output by the backbone, seeking to improve the effectiveness of feature representation for our tracking task. Hopefully, this may compensate the precision drop caused by model compression.

2.3. Disentangled Representation Learning

Disentangled representation is a kind of distributed feature representation that improves model performance by identifying and disentangling latent explanatory factors in the observed data. In a disentangled representation, information about an individual factor value can be readily accessed and is robust to changes in the input that do not affect this factor Dang et al. (2021). In recent years, learning to solve a down-stream task from a disentangled representation has shown great success in many tasks. For instance, Zhang et al. Zhang et al. (2020) used disentangled representation learning to disentangles an image into a liveness feature and a content feature, and the former was further used for classification. Yuan et al. Yuan et al. (2021) proposed a zero-shot voice transfer method via disentangled representation learning, in which each input voice is disentangled into two separated low-dimensional features in the embedding space, encoding speaker-related style and voice

content, respectively. Considering that each seed word's representation should have different latent semantics and be distinct when it represents a different aspect, Tai et al. Tai et al. (2021) used an aspect-disentangled representation to model the distinct latent semantics of each seed word. Xu et al. Xu et al. (2021) used disentangled representation learning to dynamically disentangle point clouds into the contour and flat part of 3D objects by which they were able to capture and refine the holistic and complementary 3D geometric semantics from the two disentangled components. Dand et al. Dang et al. (2021) used disentangled representation learning to disentangle the fine-grained visual feature into two parts: identity-related features and identity-unrelated features, in which only the identity-related features were used for the final classification.

The above examples well justify that disentangled representation learning is helpful to disentangle task-related features from a latent representation to get more effective feature representation, thus improving models' effectiveness in solving specific tasks. In this work, we make the first attempt to utilize disentangled representation learning for UAV tracking.

3. Proposed Method

3.1. DP-SiamFC++ Overview

The proposed DP-SiamFC++ consists of a backbone, a disentangled representation module, a neck and a head network. As illustrated in Fig. 2, the template branch and the search branch share the same backbone network $\phi(\cdot)$ and take the target patch Z and the search patch X as input. The output features of the backbone are fed into the disentangled representation module, where they are disentangled by two separate encoders, i.e., E_1 and E_2, into two parts: identity-related features and identity-unrelated features. In training phase, the disentangled features will be concatenated and then put into a decoder to get reconstructed features, which combined with the original one form a mean squared error (MSE) loss along with the similarity loss enforcing the same target to has close identity-related feature representation at different frames and losses in the classification and regression heads to supervise the training. Only the identity-related features are used for subsequent classification and regression tasks in order to improve the effectiveness of the feature representation. Specifically, the identity-related features from the template branch and the search branch are adjusted in the neck and then coupled with cross-correlation, which are finally fed into the classification and regression heads. The coupled features are formulated by:

$$f_l(Z, X) = E_2(\psi_l(\phi(Z))) \star E_2(\psi_l(\phi(X))), \psi_l \in \{\psi_{cls}, \psi_{reg}\}, \tag{1}$$

where $\star$ denotes the cross-correlation operation, E_2 represents the encoder for identity-related feature embedding, and $\psi_{cls}(\cdot)$ and $\psi_{reg}(\cdot)$ denote the task-specific layer for classification and regression, respectively, whose output are of the same size. The classification branch, with output being $O^{cls}_{h\times w\times 2}$, is used to predict the category for each location, while the regression branch, with output being $O^{reg}_{h\times w\times 4}$ (w and h denote the width and height, respectively), is to calculate the target bounding box at this point. $O^{cls}_{h\times w\times 2}(i, j, :)$ 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 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 DP-SiamFC++ is similar to that of SiamFC++. The main difference lies in the pruned feature maps determined by filter pruning and the disentangled representation module proposed to keep and even boost tracking performance, which will be detailed in the following subsection.

3.2. Disentangled Representation in Filter Pruning

Let's denote the i-th ($1 \leqslant i \leqslant K$) convolutional layer C^i of SiamFC++ by $W_{C^i} = \{w_1^i, w_2^i, ..., w_m^i\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i}$, where n_i indicates the number of filters in C^i, k_i denotes the kernel size, and the j-th filter is $w_j^i \in \mathbb{R}^{n_{i-1} \times k_i \times k_i}$. We denote the output feature maps of the filters by $O_{C^i} = \{o_1^i, o_2^i, ..., o_m^i\} \in \mathbb{R}^{n_i \times g \times h_i \times w_i}$, where $o_j^i \in \mathbb{R}^{g \times h_i \times w_i}$ is related to w_j^i, the number of input images by g, the height and width of the feature maps by h_i and w_i, respectively. The rank-based filter pruning, as illustrated at the bottom left of Fig. 2, is formulated as the following objective function:

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

where I denotes an input image that follows the distribution $P(I)$, 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_j^i is pruned, $\delta_{i,j} = 1$ if it is, otherwise $\delta_{i,j} = 0$. $\mathfrak{R}(\cdot)$, as a measure of information richness, computes the rank of a feature map. Empirical evidence shows that the expectation of ranks connected to a single filter is insensitive to the input images Lin et al. (2020), on the ground of which Eq. (2) is approximated by

$$\min_{\delta_{i,j}} \sum_{i=1}^{K} \sum_{j=1}^{n_i} \delta_{i,j} \sum_{t=1}^{g} \mathfrak{R}(o_j^i(I_t)), \quad s.t \sum_{j=1}^{n_i} \delta_{i,j} = n_i^p, \tag{3}$$

where t indexes the input images. Eq. (3) is easily minimized by pruning n_i^p filters with the least average rank of feature maps.

The performance of the classification and the regression tasks in the tracker strongly depends on the extracted feature by the backbone. However, identity-related and identity-unrelated information is entangled in the feature representation. Hence, the latter could be a discriminate cue for classification and regression during training, degrading the generalization ability of the model. Therefore, we propose to learn to disentangle the identity-related features from the identity-unrelated ones in an unsupervised manner. The disentangled representation module is integrated to achieve a disentangled representation for classification and regression after the filter pruning is completed. The module consists of an encoder-decoder network architecture with carefully designed loss functions, as shown in Fig. 2. The encoder is comprised of two sub-encoders, namely, E_1 and E_2, splitting the backbone feature $\mathbf{f}$ into two parts, namely, identity-unrelated $\mathbf{f}_u$ and identity-related $\mathbf{f}_r$ features:

$$\mathbf{f}_u = E_1(\mathbf{f}), \quad \mathbf{f}_r = E_2(\mathbf{f}). \tag{4}$$

These two features are expected to fully describe $\mathbf{f}$ since they can be decoded back to the original feature through the decoder D:

$$\tilde{\mathbf{f}} = D(\mathbf{f}_r, \mathbf{f}_u). \tag{5}$$

Reconstruction Loss. The reconstructed $\tilde{\mathbf{f}}$ should be close to the original backbone feature $\mathbf{f}$. Hence, the reconstruction loss, defined by the MSE loss, is to punish the differences between $\tilde{\mathbf{f}}$ and $\mathbf{f}$, which is defined as follow,

$$L_{rcons} = \left\| \tilde{\mathbf{f}} - \mathbf{f} \right\|_2^2 = \| D(\mathbf{f}_r, \mathbf{f}_u) - \mathbf{f} \|_2^2 . \tag{6}$$

Identity Similarity Loss. Suppose the template Z and Z' have the same identity, i.e., representing the target at different frames. Then the two corresponding identity-related features $E_2(\phi(Z))$ and $E_2(\phi(Z'))$ should be as close as possible for the purpose of classification. So we use the MSE loss to define the identity similarity loss to punish the differences between the two identity-related features, which is formulated by

$$L_{Idsim} = \| E_2(\phi(Z)) - E_2(\phi(Z')) \|_2^2 . \tag{7}$$

Classification, Regression and Centerness Loss. We now define the loss functions for learning the classification and regression tasks. 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) correspond to the spatial location (i, j) on $O^{reg}_{h\times w\times 4}$, 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

$$\begin{aligned} \hat{t}^0_{(i,j)} &= \hat{l} = x - x_0, \quad \hat{t}^1_{(i,j)} = \hat{t} = y - y_0, \\ \hat{t}^2_{(i,j)} &= \hat{r} = x_1 - x, \quad \hat{t}^3_{(i,j)} = \hat{b} = y_1 - y. \end{aligned} \tag{8}$$

The loss for regression is defined as follows

$$L_{reg} = \frac{1}{\sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)})} \sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)}) L_{IoU}(O^{reg}_{h\times w\times 4}(i, j, :), \hat{t}_{(i,j)}), \tag{9}$$

where L_{IoU} indicates the IoU loss as in Yu et al. (2016), $\mathbb{I}(\cdot)$ is the indicator function defined by:

$$\mathbb{I}(\hat{t}_{(i,j)}) = \begin{cases} 1 & if \quad \hat{t}^k_{(i,j)} > 0, k = 0, 2, 2, 3 \\ 0 & otherwise. \end{cases} \tag{10}$$

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

$$c(i, j) = \mathbb{I}(\hat{t}_{(i,j)}) * \sqrt{\frac{min(\hat{l}, \hat{r})}{max(\hat{l}, \hat{r})} \times \frac{min(\hat{t}, \hat{b})}{max(\hat{t}, \hat{b})}}. \tag{11}$$

Then the centerness loss is defined by

$$\begin{aligned} L_{cen} = \frac{-1}{\sum_{i,j} \mathbb{I}(\hat{t}_{(i,j)})} \sum_{\mathbb{I}(\hat{t}_{(i,j)})==1} & c(i, j) * log(O^{cen}_{h\times w\times 1}(i, j)) \\ & + (1 - c(i, j)) * log(1 - O^{cen}_{h\times w\times 1}(i, j)). \end{aligned} \tag{12}$$

The overall loss for training PD-SiamFC++ is:

$$L = L_{cls} + \lambda_1 L_{reg} + \lambda_2 L_{cen} + \lambda_3 L_{rcons} + \lambda_4 L_{Idsim}, \tag{13}$$

where L_{cls} is the cross-entropy loss for classification, λ_1, λ_2, λ_3, and λ_4 are predefined constants to balance the losses.

3.3. Pruning Schedule

The pipeline of pruning is as follows: First, calculate the average rank of the feature map of any filter in each layer to obtain the rank sets $\{R^i\}_{i=1}^K = \{\{r_1^i, r_2^i, ..., r_{n_i}^i\}\}_{i=1}^K$. Second, each set 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 the j-th top value in R^i. Third, conduct filter pruning according to a predefined global pruning ratio ρ, after which R^i turns to $\hat{R}^i = \{r_{s_1^i}^i, r_{s_2^i}^i, ..., r_{s_{\hat{n}_i}^i}^i\}$, $\hat{n}_i = n_i - n_i^p$ and integrate the disentangled representation module to obtain the DP-SiamFC++ model. Finally, the filters maintained are initialized with the original weights, and the parameters in the disentangled representation module are randomly set, before DP-SiamFC++ is finetuned.

4. Experiments

4.1. Experiment Settings

Four challenging UAV benchmarks, i.e., UAV123@10fps Mueller et al. (2016), DTB70 Li and Yeung (2017), UAVDT Du et al. (2018) and Vistrone2018 Wen et al. (2018) are used to conduct our experiment. UAV123@10fps is created by reducing the original 30FPS UAV123 benchmark to 10FPS, and it is designed to investigate the effect of camera capture speed on tracking performance. DTB70 is made up of 70 UAV sequences that largely address the issue of extreme UAV motion. UAVDT is primarily used for vehicle tracking in a variety of weather situations, altitudes, and camera perspectives. Vistrone2018 is from a challenge held in conjunction with the European conference on computer vision (ECCV2018) that focuses on evaluating tracking algorithms on drones.

Our encoder-decoder network is a typical CNN in which an encoder consists of a stride-2 convolutional layer followed by a Batch Normalization and a Leaky ReLU layer and the decoder is built from a transposed convolution, a Batch Normalization and a Leaky ReLu layer. All evaluation experiments are carried out on a PC equipped with i9-10850K processor (3.6GHz), 16GB RAM, and an NVIDIA TitanX GPU. The global pruning ratio of our DP-SiamFC++ is set to 0.5. The λ_1, λ_2, λ_3, and λ_4 (Eqn. 13) are set to 1.0, 1.0, 0.05, and 0.01 in all experiments. Other parameters follow SiamFC++ Xu et al. (2020). Code is available on: https://github.com/PD-SiamFCpp/-PD-SiamFCpp.

4.2. Comparison with DCF-based Trackers

Ten state-of-the-art trackers based on handcrafted features for comparison are: SAMF Li and Zhu (2014), fDSST Danelljan et al. (2016), Staple-CA Mueller et al. (2017), BACF Galoogahi et al. (2017), ECO-HC Danelljan et al. (2017), MCCT-H Wang et al. (2018), STRCF Li et al. (2018), ARCF-HC Huang et al. (2019), AutoTrack Li et al. (2020b), and RACF Li et al. (2021).

Quantitative evaluation: Fig. 3 shows the overall performance of DP-SiamFC++ with the competing trackers on the four benchmarks. As can be seen, DP-SiamFC++ outperformed all other trackers by a wide margin on three benchmarks, i.e., UAV123@10fps, DTB70 and UAVDT. Specifically, DP-SiamFC++ dramatically outperforms the second tracker RACF in terms of precision[1] and area under curve (AUC), with gains of (4.3%,

1. The precision metric in our experiment means distance precision at 20 pixels unless otherwise stated.

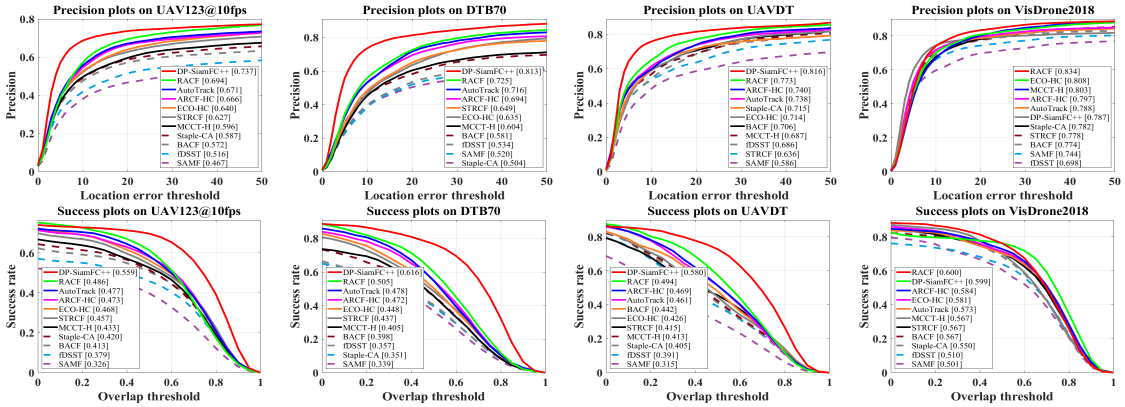


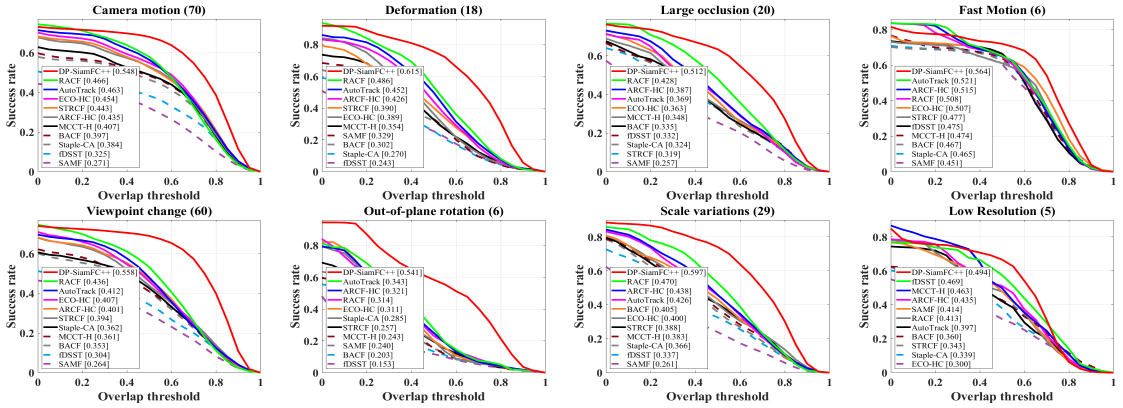
Figure 3: Overall performance of ten hand-crafted based trackers and our DP-SiamFC++ on, from left to right, UAV123@10fps, DTB70, UAVDT and VisDrone2018. Precision and success rate for one-pass evaluation (OPE) are used for evaluation. The precision at 20 pixels and area under curve (AUC) are used for ranking and marked in the precision plots and success plots respectively.

Table 1: Average precision and speed (FPS) comparison between DP-SiamFC++ and hand-crafted based trackers on the four benchmarks. All the reported FPSs are evaluated on a single CPU. Red, blue and green respectively indicate the first, second and third place.

	KCF TPAMI 15	fDSST CVPR 16	Staple-CA CVPR 17	BACF ICCV 17	ECO-HC CVPR 17	MCCT-H CVPR 18	STRCF CVPR 18	ARCF-HC ICCV 19	AutoTrack CVPR 20	RACF 3DV 21	**DP-SiamFC++ Ours**
Precision	53.3	60.4	64.2	65.3	68.8	66.8	67.1	71.9	**72.3**	**75.7**	**78.5**
FPS (CPU)	**655.6**	**203.6**	67.7	57.0	88.9	66.7	29.9	36.0	61.8	37.5	**98.0**

7.3%), (8.8%, 11.1%) and (4.3%, 8.6%), respectively. For VisDrone2018, although DP-SiamFC++ gets the second place on AUC, its precision is inferior to the first tracker RACF by a significant gap of 4.7%. But this large gap may be attributed to the inferior precision of the original model SiamFC++ as shown in Table 3, which is only 72.5%, much lower than 78.7% of DP-SiamFC++ by a gap of 6.2%. We use the average FPS on CPU throughout the four benchmarks as a metric for speed. Table 1 illustrates the average precision and FPS produced by different trackers. It can be seen that DP-SiamFC++ outperforms all the competing trackers in precision, and is also the most effective real-time tracker (speed of >30FPS) on CPU. To be specific, DP-SiamFC++ achieves 78.5% in precision at a very fast CPU speed of 98.0 FPS in average.

Attribute-based evaluation: The proposed DP-SiamFC++ outperforms other competing DCF-based trackers in most attributes defined respectively in the four benchmarks except on VisDrone2018. Examples of success plots are shown in Fig. 4. As can be seen, in the situations of camera motion and viewpoint change on UAV123@10fps, deformation and out-of-plane rotation on DTB70, large occlusion and scale variation on UAVDT, fast motion and low resolution on VisDrone2018, DP-SiamFC++ demonstrates a significant improvement over other trackers because the effectiveness of feature representation by



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

Figure 4: Attribute-based comparison on camera motion, viewpoint change, deformation, out-of-plane rotation, large occlusion, scale variations, fast motion, and low resolution.

learning. For example, DP-SiamFC++ significantly surpasses the second place RACF on the viewpoint change subset of UAV123@10fps by a gap of 12.2%, and surpasses the second place AutoTrack on the out-of-plane rotation subset of DTB70 by a gap of 19.8%. Due to space limitation, all attribute-based evaluation results are displayed in the supplementary materials.

Qualitative evaluation: In Fig. 5, we show some qualitative tracking results of our method in comparison with four top CPU-based trackers. It can be seen in these challenging situations that the four CPU-based trackers finally fail to accurately track the objects when objects are experiencing large pose changes (i.e., truck1 and uav0000294_0000_s) in-plane rotation (i.e., S0102), and severe occlusion (i.e., Horse2). However, our DP-SiamFC++ performs much better and is visually more satisfying in these cases owing to the powerful deep representation learning. It suggests that developing more efficient DL-based trackers through model compression, for instance, maybe more effective in increasing the tracking precision for UAV tracking.

4.3. Comparison with DL-based Trackers

The proposed DP-SiamFC++ is also compared with ten state-of-the-art DL-based trackers, namely, PrDiMP18 Danelljan et al. (2020), SiamR-CNN Voigtlaender et al. (2020), D3S Lukezic et al. (2020), KYS Bhat et al. (2020), SiamGAT Guo et al. (2021), LightTrack Yan et al. (2021), TransT Chen et al. (2021), HiFT Cao et al. (2021), SOAT Zhou et al. (2021), and AutoMatch Zhang et al. (2021).

The FPSs and the precisions of the competing trackers on UAVDT are shown in Table 2. As is shown, although DP-SiamFC++ is not as accurate as TransT, SOAT, and AutoMatch, the gap is less than or equal to 1.0%, i.e., 1.0%, 0.5%, and 0.5%, respectively. Meanwhile the GPU speed of DP-SiamFC++ is greater than 8 times that of the first tracker TransT and is greater than 12 times that of second tracker SOAT. This justifies that our DP-SiamFC++ achieves a remarkable balance between precision and efficiency.

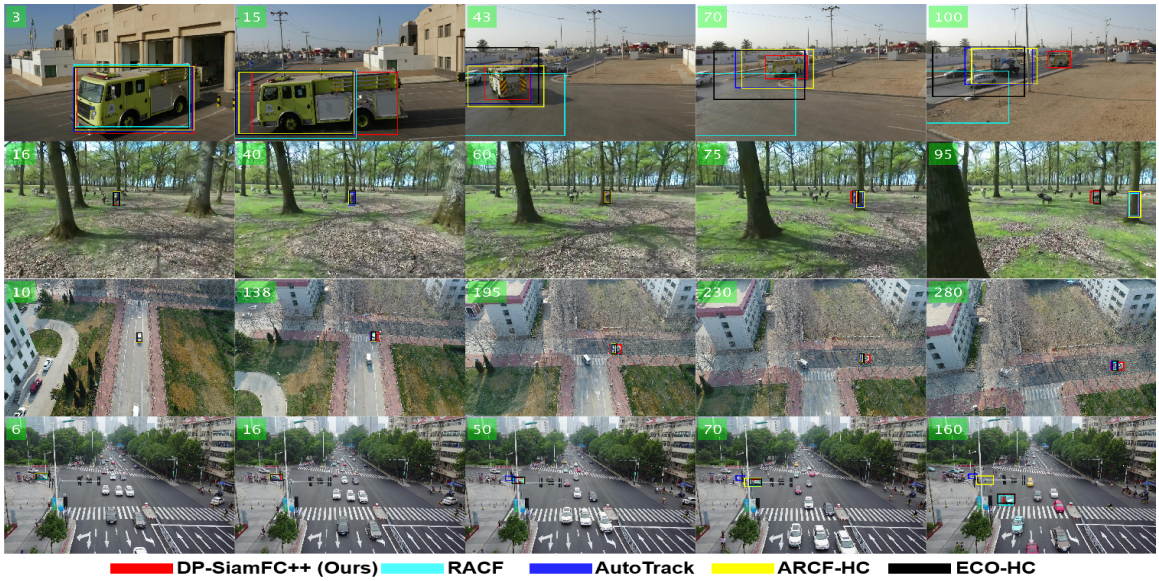


Figure 5: Qualitative evaluation on 4 sequences from UAV123@10fps, DTB70, UAVDT and VisDrone2018 (i.e. truck1, Horse2, S0102 and uav0000294_00000_s), respectively. The results of different methods have been shown with different colors.

Table 2: Precision and speed (FPS) comparison between DP-SiamFC++ and deep-based trackers on UAVDT Du et al. (2018). All of the FPSs were tested on a single GPU. ρ stands for the global pruning ratio.

	SiamR-CNN CVPR 20	D3S CVPR 20	PrDimp18 CVPR 20	KYS ECCV 20	SiamGAT CVPR 21	LightTrank CVPR 21	TransT CVPR 21	HiFT ICCV 21	SOAT ICCV 21	AutoMatch ICCV 21	**DP-SiamFC++ Ours**
Precision	66.5	72.2	73.2	79.8	76.4	80.4	**82.6**	65.2	**82.1**	**82.1**	**81.6**
FPS (GPU)	7.2	44.6	48.5	30.2	74.8	**84.8**	42.1	**135.3**	29.4	50.4	**374.5**

4.4. Ablation Study

Evaluation of disentangled representation in pruning: We compare the proposed DP-SiamFC++ with the baseline SiamFC++ on all four UAV benchmarks to examine how the model size, accuracy, and tracking speed change when the suggested filter pruning is applied to the baseline SiamFC++. Table 3 shows results. We can observe that the model size of DP-SiamFC++ is reduced to about 50.0% (4.83/9.66) of the original, the CPU and GPU speed are thus significantly raised. Since the parallel computing units on our GPU far exceed the size of both models, the GPU speed has increased, approximately, by only 62.9% on average, from 228.4 FPS to 372.1 FPS. But the average CPU speed is raised from 36.5 FPS to 98.0 FPS, a noticeable increase of about 168.5%. Remarkably, although DP-SiamFC++ is slightly inferior to the baseline SiamFC++ on UAV123@10fps with a small gap of 0.5%, it outperforms the baseline on all the other three benchmarks, and the improvements on UAVDT and VisDrone2018 are very significant, with the gains being 5.4% and 6.2% in precision, respectively. These results justify the effectiveness of the proposed method of learning disentangled representation in pruning, which not only is able

Table 3: Comparison of model size (Parameters), precision (DP) and tracking speed (FPS) on CPU and GPU between proposed DP-SiamFC++ and the baseline method.

Methods	Parameters	UAV123@10fps			DTB70			UAVDT			VisDrone2018			Avg.		
		DP	FPS		DP	FPS		DP	FPS		DP	FPS		DP	FPS	
			CPU	GPU		CPU	GPU		CPU	GPU		CPU	GPU		CPU	GPU
SiamFC++	9.66M	**72.8**	36.2	226.5	80.5	36.1	232.6	76.2	36.9	243.8	72.5	37.0	211.0	75.5	36.5	228.4
DP-SiamFC++ (Ours)	**4.83M**	72.3	**98.8**	**373.7**	**81.2**	**95.7**	**372.8**	**81.6**	**98.2**	**374.5**	**78.7**	**99.4**	**367.3**	**78.5**	**98.0**	**372.1**

Table 4: Illustration of how the precision of DP-SiamFC++ with or without disentangled representation varies with the global pruning ratio, ranging from 0.1 to 0.8 in step of 0.1. The precisions that have been improved by the disentangled representation are marked in bold.

ρ	UAV123@10fps		DTB70		UAVDT		VisDrone2018	
	w/o	w/	w/o	w/	w/o	w/	w/o	w/
0.1	70.8	**72.5**	**79.6**	79.4	**81.4**	77.4	**79.6**	75.4
0.2	71.6	**73.4**	80.0	**80.4**	**76.9**	76.5	80.2	**80.6**
0.3	71.3	**73.8**	**81.0**	77.9	**83.9**	80.1	**75.6**	75.2
0.4	71.9	**72.1**	79.5	**79.6**	78.8	**78.8**	**79.3**	77.2
0.5	71.1	**72.2**	77.6	**81.3**	77.5	**81.6**	72.7	**78.7**
0.6	**68.7**	68.0	**78.6**	76.2	76.6	**82.2**	75.2	**80.0**
0.7	**69.1**	67.9	**77.9**	76.5	77.8	**79.4**	76.7	**78.8**
0.8	65.2	**65.7**	**76.4**	73.8	74.6	**78.0**	71.8	**73.4**

to improve efficiency because of the reduced model size but also is capable of remaining and even increasing the precision due to the disentangled representation module introduced.

Impact of the global pruning ratio: We trained DP-SiamFC++ with different global pruning ratios to understand the impact of the global pruning ratio on the final precision of DP-SiamFC++. For comparison, we also trained and evaluated DP-SiamFC++ with different global pruning ratios but without the proposed disentangled representation module being incorporated. Specifically, each convolutional layer in the backbone, neck and head is pruned with the same global ratio, which ranges from 0.1 to 0.8. Note that the larger the ratio is, the more filters will be pruned. The precisions of DP-SiamFC++ with or without the proposed disentangled representation module with respect to the global pruning ratio are shown in Table 4. As can be observed, the highest precisions are obtained at pruning ratios smaller than 0.6, which is consistent with our expectations for filter pruning techniques that the larger the pruning ratio the larger the decrease of precision may be. However, most precisions are improved with the proposed disentangled representation module incorporated. For example, six out of eight precisions are improved on UAV123@10fps, five out of eight are raised on both UAVDT and VisDrone2018. This enables us to maintain very favorable precisions with higher pruning ratios, especially when the global ratio is the default setting 0.5 at which the increases of precision are 1.1%, 3.7%, 4.1%, and 6.0%, respectively, on UAV123@10fps, DTB70, UAVDT, and VisDrone2018. It demonstrates the effectiveness of our method of learning disentangled representation in pursuing for real-time UAV tracking.

5. Conclusion

In this work, we propose to learn disentangled representation in pruning for real-time UAV tracking for the first time and achieve state-of-the-art performance on four public UAV tracking benchmarks. Experimental results show that the proposed method is very effective to maintain and even improve precision when applying the rank-based filter pruning to improve UAV tracking efficiency. Remarkably, the proposed DP-SiamFC++ not only significantly improves efficiency over the baseline SiamFC++, specifically DP-SiamFC++ can run at more than 98 FPS on a single CPU, but also significantly increases precision on UAVDT and VisDrone2018, well combating the adverse effects of using filter pruning.

Acknowledgments

Thanks to the supports by Guangxi Key Laboratory of Embedded Technology and Intelligent System and Guangxi Science and Technology Base and Talent Special Project (No. Guike AD22035127)

References

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

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

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

David S Bolme, J Ross Beveridge, and at al. Visual object tracking using adaptive correlation filters. In *2010 IEEE CVPR*, pages 2544–2550. IEEE, 2010.

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

Xin Chen, Bin Yan, and et al. Transformer tracking. In *CVPR*, pages 8126–8135, 2021.

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

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

Martin Danelljan, Goutam Bhat, and at al. Eco: Efficient convolution operators for tracking. In *CVPR*, pages 6931–6939, 2017.

Martin Danelljan, LucVanGool, and at al. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020.

Wenjie Dang, Shuiwang Li, and et al. Learning disentangled representation for fine-grained visual categorization. In *ICIG*, 2021.

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

Hamed Kiani Galoogahi, Ashton Fagg, and at al. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017.

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

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

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

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

Shuiwang Li, Qianbo Jiang, and et al. Asymmetric discriminative correlation filters for visual tracking. *Frontiers Inf. Technol. Electron. Eng.*, 21:1467–1484, 2020a.

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

Shuiwang Li, Yuting Liu, and et al. Learning residue-aware correlation filters and refining scale for real-time uav tracking. *Pattern Recognit.*, 127:108614, 2022.

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

Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European Conference on Computer Vision*, pages 254–265, 2014.

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

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

Alan Lukezic, Jiri Matas, and at al. D3s – a discriminative single shot segmentation tracker. In *CVPR,2020*, pages 7133–7142, 2020.

Matthias Mueller, Neil Smith, and at al. A benchmark and simulator for uav tracking. *FJMS*, 2(2):445–461, 2016.

Matthias Mueller, Neil Smith, and at al. Context-aware correlation filter tracking. In *CVPR*, pages 1387–1395, 2017.

Chang-You Tai, Ming-Yao Liand, and et al. Hyperbolic disentangled representation for fine-grained aspect extraction. *ArXiv*, abs/2112.09215, 2021.

Paul Voigtlaender, Jonathon Luiten, and et al. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, pages 6578–6588, 2020.

Huan Wang, Can Qin, and et al. Neural pruning via growing regularization. *arXiv:2012.09243*, 2020.

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

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

Xucheng Wang, Dan Zeng, Qijun Zhao, and Shuiwang Li. Rank-based filter pruning for real-time uav tracking. In *ICME*, pages 01–06, 2022.

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

Wanying Wu, Pengzhi Zhong, and Shuiwang Li. Fisher pruning for real-time uav tracking. In *IJCNN*, pages 1–7, 2022.

Fei Xie, Chunyu Wang, and et al. Learning tracking representations via dual-branch fully transformer networks. In *ICCVW*, pages 2688–2697, 2021.

Mutian Xu, Junhao Zhang, and et al. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI*, 2021.

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

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

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

Siyang Yuan, Pengyu Cheng, and et al. Improving zero-shot voice style transfer via disentangled representation learning. *ArXiv*, abs/2103.09420, 2021.

Ke-Yue Zhang, Taiping Yao, and et al. Face anti-spoofing via disentangled representation learning. *ArXiv*, abs/2008.08250, 2020.

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

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