electronics

Article

SiamPRA: An Effective Network for UAV Visual Tracking

Jiafeng Li [1,2,*], Kang Zhang [1], Zheng Gao [1], Liheng Yang [1] and Li Zhuo [1,2]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; zhangawawa@emails.bjut.edu.cn (K.Z.); gz@emails.bjut.edu.cn (Z.G.); ylh@emails.bjut.edu.cn (L.Y.); zhuoli@bjut.edu.cn (L.Z.)

[2] Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing 100124, China

* Correspondence: lijiafeng@bjut.edu.cn

Abstract: The visual navigation system is an important module in intelligent unmanned aerial vehicle (UAV) systems as it helps to guide them autonomously by tracking visual targets. In recent years, tracking algorithms based on Siamese networks have demonstrated outstanding performance. However, their application to UAV systems has been challenging due to the limited resources available in such systems.This paper proposes a simple and efficient tracking network called the Siamese Pruned ResNet Attention (SiamPRA) network and applied to embedded platforms that can be deployed on UAVs. SiamPRA is base on the SiamFC network and incorporates ResNet-24 as its backbone. It also utilizes the spatial-channel attention mechanism, thereby achieving higher accuracy while reducing the number of computations. Further, sparse training and pruning are used to reduce the size of the model while maintaining high precision. Experimental results on the challenging benchmarks VOT2018, UAV123 and OTB100 show that SiamPRA has a higher accuracy and lower complexity than other tracking networks.

Keywords: object tracking; attention mechanism; network compression; embedded vision system

Citation: Li, J.; Zhang, K.; Gao, Z.; Yang, L.; Zhuo, L. SiamPRA: An Effective Network for UAV Visual Tracking. *Electronics* **2023**, *12*, 2374. https://doi.org/10.3390/electronics12112374

Academic Editors: Yoonsik Choe and Taehyeon Kim

Received: 26 April 2023
Revised: 19 May 2023
Accepted: 22 May 2023
Published: 24 May 2023

1. Introduction

Visual object tracking is attracting significant attention in the field of computer vision [1–3], involving a broad extent of applications based on UAV such as industrial inspection, power line inspection [4], and intelligent transportation. For example, in high-voltage transmission line inspection, the use of UAV not only reduces the threat of electrocution of personnel, but also reduces maintenance costs and improves efficiency. The navigation task refers to the process of the control system guiding the UAV to fly to the designated target location. In this process, the visual navigation system uses the video sequence collected by an airborne optical sensor to continuously lock the target position according to the set target. Because the system is not easily affected by external interference signals and has a high degree of autonomy, visual navigation plays an important role as an effective navigation method in achieving long-endurance and high-precision navigation of UAVs in the case of GPS failure. However, the high flight speed of the UAV itself, complex backgrounds, and large changes in the tracking target scale have increased the difficulty of autonomous visual navigation, resulting in the need for more accurate real-time target tracking algorithms.

Early visual object tracking algorithms used traditional manual feature analysis techniques such as particle filter tracking [5] and correlation filter (CF). Owing to the development and expansion of different industrial application scenarios, the generalization ability of traditional algorithms can no longer meet the requirements of these systems. They have difficulty dealing with scenarios involving complex backgrounds, partial occlusions, and small targets.

With the development of deep learning technology, tracking methods based on the Siamese architecture [6–11] have gradually become the standard for visual tracking. Related methods have proven effective in deep feature extraction for targets. However, most deep learning-based tracking algorithms are at the theoretical stage. Furthermore, optimizing the network complexity of the overall system to reduce the amount of calculations is not considered. This paper proposes an efficient target tracking network that employs a lightweight convolutional neural network combined with an attention mechanism to extract the deep semantic features of the target and background. This study reduces the complexity of the algorithm through optimization strategies, making it suitable for practical application in various embedded computing platform.

The location of the target has significant spatial characteristics and the target's various characteristics in different channels of the feature map have corresponding relationships. The attention mechanism can select global features and improve the recognition accuracy by suppressing information that is weakly related to the target. Yang et al. [12] proposed a novel visual tracking method that utilizes spatial selective attention to discriminatively learn from historical data; it dynamically identifies discriminative attentional regions. Woo et al. [13] proposed a convolutional block attention module (CBAM) that infers the attention map from the channel and space dimensions in turn to enable the model to focus on the target. Fu et al. [14] successfully applied the attention mechanism to image segmentation, greatly improving the objective effect of the segmentation. Wu et al. [15] proposed a cross-spatial attention mechanism module (CCLA), which makes the algorithm focus on global and local context information, and improves the accuracy of object state estimation. In the current study, we developed a visual tracking feature extraction network based on spatial and channel attention to focus attention on the location and feature correlation of the tracking target in the video sequence, thereby obtaining more robust and accurate tracking of target depth features.

As stated above, deep networks can extract the deep semantic features of the target in visual object tracking with better generalization that traditional approaches. However, most deep learning algorithms are computation-intensive and are not designed for embedded vision systems, such as those used in UAV visual navigation system. In recent years, lightweight networks have proved to be effective solutions to the above problems. Howard et al. [16] proposed a lightweight network that uses depth-wise separable convolution to fully reduce the amount of computation involved in ordinary convolution. Further, Liu et al. [17] directly pruned the network through several early pruning methods and used retraining to achieve performance equivalent to the original network. Inspired by these previous studies, this study optimizes the efficiency of the system by designing a pruning algorithm. The network parameters and calculations are reduced by removing unimportant calculations and retaining the most effective network components.

Based on the above analysis, this paper proposes a novel network architecture and system optimization method. The main contributions of this study are as follows:

(1) A lightweight visual tracking network, SiamRA (Siamese ResNet Attention), is proposed for embedded vision systems. A spatial-channel attention module is introduced to strengthen the attention to the spatial area of the target in the image and the correlation of similar features between different channels, achieving excellent tracking results with low-cost computing.

(2) A pruning algorithm is designed for SiamRA. Sparse training and pruning minimize the network's calculations, creating a high-performance and lightweight target tracking network, SiamPRA (Siamese Pruned ResNet Attention).

The remainder of this paper is organized as follows. In Section 2, previous studies related to the SiamPRA tracking algorithm are discussed. In Sections 3 and 4, the proposed architecture and optimization algorithm is presented in detail. Section 5 presents the results of ablation experiments and their performance analysis. Finally, the conclusions of this study are presented in Section 6.

2. Related Work

Visual tracking algorithms can be broadly divided into two categories based on the feature extraction methods employed: traditional visual tracking algorithms and deep learning-based visual tracking algorithms.

Traditional visual tracking algorithms primarily use manual features. For example, Vojir et al. [18] proposed an algorithm based on the mean-shift theory using the weighted color histogram generated by the image for tracking. Henriques et al. [19] successfully integrated CF into target tracking tasks in 2014, maintaining high accuracy at 172 FPS. Subsequently, the research on target tracking algorithms began to focus on the improvement of CF algorithms. For example, Lukezic et al. [20] proposed a CSR-DCF method that introduced the concept of channel reliability and spatial reliability into DCF tracking and also proposed a learning algorithm to enable DCF tracking to be efficiently and seamlessly integrated in the process of filter update and tracking; Bertinetto [21] fused the HOG feature and COLOR feature with the CF algorithm, achieving simple and efficient tracking at 80 FPS. Although the processing speed of traditional algorithms is relatively high, the tracking accuracy is not satisfactory because of the small scale of the tracking target and the variable angle of view. This is because these methods have limited ability to extract the features of the target and background.

In recent years, deep learning technology has been successfully applied to visual tracking through deep semantic feature matching to obtain the location of the target. Some early algorithms combined correlation filtering with deep convolution features to achieve higher accuracy. These convolutional neural network models had strong convolution features and good generalization capabilities, but their speed was low; for example, ECO [22]. The SiamFC algorithm [6] effectively applied the Siamese network to the visual tracking task for the first time, quickly replacing the CF algorithm to become the main research direction of current visual tracking algorithms. Li et al. [7] proposed SiamRPN, adding a region proposal network based on SiamFC and strengthening the accuracy of target detection through regression of the target frame and score. It can reach 160 FPS but is memory and storage space-consuming. Zhu et al. [8] added negative samples of different targets of the same kind to the training dataset to enhance the discrimination ability of the model. Zhang et al. [23] used no-padding-residual units to design a deeper and wider Siamese tracker, avoiding the padding operation affecting the translation invariance of the network. Nousi et al. [24] improved the robustness of the tracker by extracting the target corresponding histogram to deal with the change of target appearance during the tracking process. Li et al. [9] used image pair translation to avoid positional deviation, solving translation invariance and strengthening the learning ability of the network through feature fusion. Xu et al. [10] applied the anchor-free concept in the detection algorithm to replace the anchor-base in SiamRPN, obtaining a more accurate bounding-box to enhance the accuracy of the tracking. Tan et al. [25] learned feature weights through two non-local blocks, refined target features to suppress noise interference, and correlated multiple response maps to improve model robustness. Tang et al. [26] rank the importance of training samples, give more weight to important samples, then rank the proposals generated by the lightweight network prediction RPN, and finally combine the classification to decide the tracking results. Yao et al. [27] studied template candidate matching for Siamese trackers from the perspective of image alignment, which enhanced the scene adaptability of the algorithm. Yang et al. [28] introduced a modified corner pooling layer to convert the target bounding box estimation into diagonal prediction, and introduced a layered feature fusion strategy to enable the corner pooling module to predict multiple corners of the tracked target in the deep network. Zhou et al. [29] designed a localization branch to directly locate the object center to help the regression branch to generate accurate results in a variety of challenging scenarios and improve the robustness of the algorithm. Zhao et al. [30] proposed a target template update method based on deep reinforcement learning, which enables the tracker to learn the update strategy of the template pool, so that the algorithm continuously adapts to the appearance changes of the target. Chan et al. [31] implicitly encode the shape and scale information of the target into an anchor-free method, and then utilize the combined module of the channel attention mechanism and the non-local attention mechanism to improve the perception of key features of the template and avoid the interference of background clutter, thereby improving the Robustness of the model. The visual tracking algorithm based on deep learning improves the feature extraction ability by deepening the network and improves the tracking accuracy to a large extent. However, the deeper network architecture requires significant amounts of storage space and computing power, which is incongruent with the real-time processing needs of embedded computing platform.

3. Proposed SiamRA Scheme

In this section, the visual tracking algorithm SiamRA is introduced in detail. Figure 1 shows the overall architecture, which consists of scale adjustment pre-processing, the deep feature extraction backbone ResNet-24, the attention extraction module, and a target positioning module.

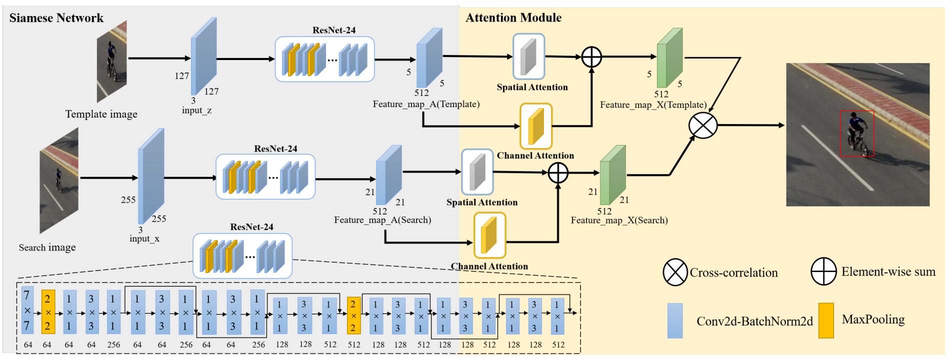


Figure 1. Overall Network Architecture.

3.1. Scale Adjustment Pre-Processing

The coordinate position of the tracking target in the video sequence collected by the visual navigation system usually moves in a relatively small range between two adjacent image frames without any drastic changes. Appropriate cropping of the input image greatly reduces the computation for the target location search. Therefore, our proposed method searches within a range that is twice the length and width of the bounding-box of the pre-set target. In this study, the target image is resized to 127 $\times$ 127 for extracting deep features, and the background image is resized to 255 $\times$ 255 for searching. These images are inputted into two pre-trained shared parameter ResNet-24 twin network branches, namely, $input\_z$ and $input\_x$ in Figure 1, to extract deep visual features. Because the padding operation in ResNet24 destroys the translational invariance of the network, leading to poor tracking results, we removed the effect of padding by cropping the feature maps after each padding operation. In the tracking process, the tracking result is not used as a new template image, and the initial image provided by the first frame continues to be used; this is because the image in the first frame is usually selected as the image when the target is completely acquired, which is more accurate. Although the continuous updating of the template image can help to determine the characteristics of the target in the current state, it is often affected by factors such as occlusion, blur, and background similarity during the tracking process. In this case, updating the tracking results to the template image would affect the tracking accuracy. Meanwhile, because the template branching remains unchanged, its output also does not change. This output is stored in the memory after the first frame is calculated, and it is not updated in subsequent tracking, which also reduces the number of calculations.

3.2. Feature Extraction

The ResNet-24 network architecture has a unique residual block module that deepens the network while avoiding the vanishing gradient problem. At the same time, its MACs and Params are only 9.5% and 6.2% of ResNet-50, making it more compatible with the constrained resources of embedded computing platforms. However, compared with ResNet-50, its relatively latent network depth inevitably results in a lack of feature extraction capabilities.

To address this problem, this study proposes an attention module in the system to further mine deep features. As shown in Figure 1, the features extracted by ResNet-24 are input into a spatial-channel attention module that aggregates features from different locations and channels of the target by giving higher weights to the region of interest. As shown in Figure 2, this attention module helps the system to suppress feature redundancy in the spatial and channel dimensions and better categorize the features of the target and background.

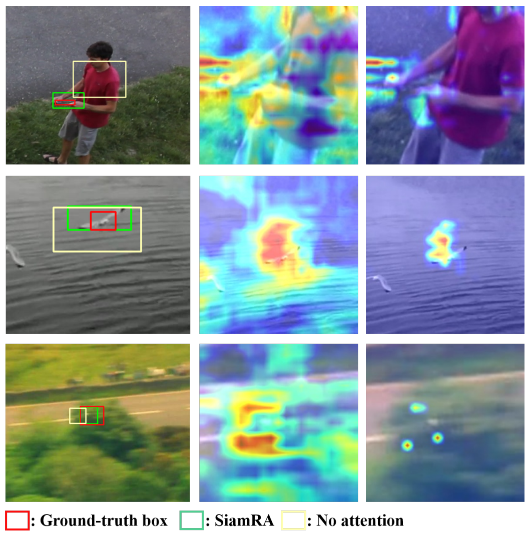


Figure 2. Attention mechanism visualization. Left to right: tracking results, no attention, and spatial-channel attention. Top to bottom: frisbee, bird, and obscured motorcycle.

The architecture of the attention module used in this study is shown in Figure 3. It is composed of spatial and channel attention modules without shared parameters. After *Feature_map_A* in Figure 1 is reshaped by this spatial-channel attention module, the final *Feature_map_X* is obtained for the target positioning module. The network structure of the spatial attention module is shown in Figure 4. The extraction process of spatial attention features can be divided into the following three steps.

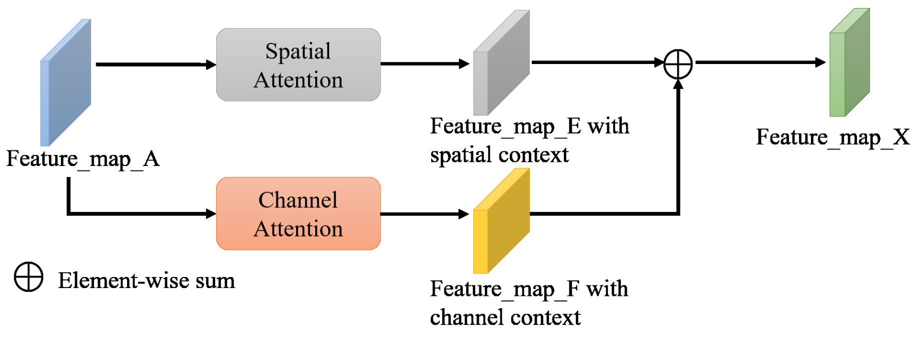


Figure 3. Structure of the Attention Module.

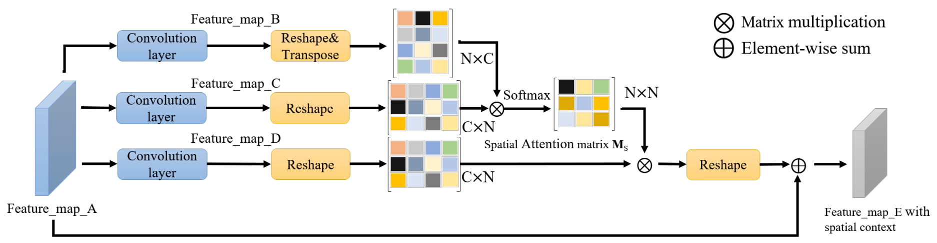


Figure 4. Overall Architecture of the Spatial Attention Module in Figure 3.

Step 1: After obtaining $Feature\_map\_A \in R^{C\times H\times W}$, input it into a convolutional layer to generate two new feature maps, $Feature\_map\_B$ and $Feature\_map\_C$, where $\{B, C\} \in R^{C\times H\times W}$. Then, reshape the two features into a size of $R^{C\times N}$, where $N = H \times W$ is the number of pixels, and further transpose $Feature\_map\_B$. Finally, perform matrix multiplication in the N domain between the transposition of B and C, obtaining the original spatial attention matrix $M_s \in R^{N\times N}$. Equation (1) uses Softmax to calculate the weight s_{ji} by row as the updated attention matrix M_s [14].

$$s_{ji} = \frac{\exp\left(B_i * C_j\right)}{\sum_{i=1}^{N} \exp\left(B_i * C_j\right)}, \tag{1}$$

where B_i is the i_{th} C-dimensional row vector in $Feature\_map\_B$, C_j is the j_{th} C-dimensional column vector in $Feature\_map\_C$, s_{ji} is a scalar that denotes the influence of the i_{th} position on the j_{th} position and plays the role of spatial attention. The more similar the feature representations of two locations are, the stronger is the correlation between them.

Step 2: Input $Feature\_map\_A$ into the convolution layer, generate a new $Feature\_map\_D$, $D \in R^{C\times H\times W}$, and reshape it into $R^{C\times N}$. Perform matrix multiplication between the transposition of $Feature\_map\_D$ and the spatial attention matrix M_s, reshaping the result into $R^{C\times H\times W}$. The result denotes the local feature of the spatial attention.

Step 3: Multiply the spatial attention local feature by a scale parameter α, and further merge it with $Feature\_map\_A$ to obtain the final spatial attention $Feature\_map\_E \in R^{C\times H\times W}$ as follows:

$$E_j = \alpha \sum_{i=1}^{N} \left(s_{ji} D_i\right) + A_j, \tag{2}$$

where α is a scale parameter initialized to zero and gaining more weight through gradual learning. It can be seen from Equation (2) that the final spatial attention $Feature\_map\_E$ is the weighted sum of all the spatial attention local features and the original features. Compared with the original features, it aggregates the contexts of the target in space.

The architecture of the channel attention module is shown in Figure 5. Its process resembles the spatial attention feature extraction process. The convolution part is removed to ensure that the relationship between channels does not change. By reshaping $Feature\_map\_A$ into $R^{C\times N}$ and multiplying it with its transpose, the channel attention matrix $M_C \in R^{C\times C}$ is calculated by the Softmax layer. This matrix is multiplied by the $Feature\_map\_A$ matrix, and the result is reshaped to $R^{C\times H\times W}$, which represents the local feature of the channel attention. Finally, the same weighting method as the spatial attention is used to combine the local features of the channel attention and $Feature\_map\_A$, obtaining the final channel attention $Feature\_map\_F \in R^{C\times H\times W}$.

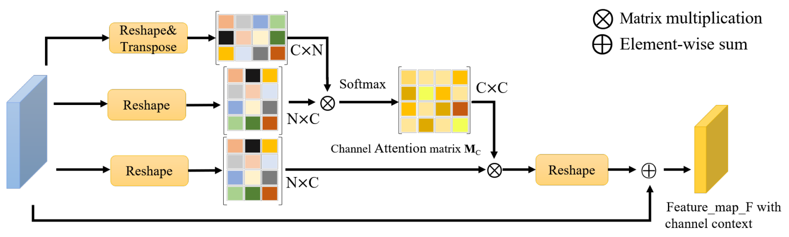


Figure 5. Overall Architecture of the Channel Attention Module in Figure 3.

3.3. Target Positioning

The target positioning module uses *Feature_map_X* (Template) as the convolution kernel and *Feature_map_X* (Search) as the feature map for convolution to achieve cross-correlation. Then, the coordinates are determined according to the confidence of the generated feature map and upsampled to the original image size using bicubic interpolation. The overall tracking network scheme is summarized in Algorithm 1.

Algorithm 1 SiamRA Tracking Network Scheme

```
1. Input: Input of this scheme
   • Train: Train on Image sequences $I$ (GOT10K, TrackingNet, and COCO) and ground-truth $gt$, Pre-train ResNet-24 model $R$;
   • Implement: Initial target coordinates $(x_0, y_0, w_0, h_0)$, Test Image sequences $I'$;
2. Output: Output of this scheme
   • Train: SiamRA tracking network model $T$;
   • Implement: Tracking result $(x_i, y_i, w_i, h_i)$;
3. Initialization: Initialize the proposed CNN with given hyperparameters
   • Epoch = 50, $\lambda$ = 0.00001;
   • Optimizer = SGD, learning-rate = 0.01~0.00001, Momentum = 0.9, Weight decay = 0.0001, Scheduler = logspace;
   • Load Pre-trained ResNet-24 model $R$ and initialize the attention module randomly;
4. Training:
   1: Obtain example images $Z$ and search images $X$ by cropping and resizing sequences $I$;
   2: for $i \rightarrow 1$ to Epoch do
   3:     Randomly shift example images $Z$ from 0 to 64;
   4:     Input example images $Z$ to input_z in Figure 1;
   5:     Randomly shift search images $X$ from 0 to 8;
   6:     Input search images $X$ to input_x in Figure 1;
   7:     Train model $T$ with logistic loss $l$ in learning-rate.
   8: end
   9: Obtain the SiamRA tracking network model $T$;
5. Implementation:
   • Input: Initial target coordinates $(x_0, y_0, w_0, h_0)$, Test Image sequences $I'$;
   • Obtain example image $Z$ by cropping and resizing initial frame from sequences $I'$;
   • Get search images $X$ by cropping and resizing sequences $I'$;
   • $(x_i, y_i, w_i, h_i) = T(Z, X)$;
   • Output: Tracking result $(x_i, y_i, w_i, h_i)$.
```

4. System Optimization

Although the SiamRA network uses the attention mechanism to reduce the complexity of the algorithm, redundant calculations are still unavoidable. In UAV systems with limited resources, any excess calculations will inevitably affect the efficiency of the visual navigation system. The removal of network redundancy by a pruning algorithm has provided excellent results in vision tasks such as target detection and classification [32]. However, the previous method can only reduce the branches of the intermediate convolution kernel in SiamRA without maximizing the system efficiency because the residual block structure of ResNet needs to ensure the consistency of the front and rear channels. To enhance the efficiency of the visual tracking system without reducing the network's deep feature extraction capability, this paper proposes SiamPRA, in which SiamRA is optimized as in Algorithm 1. In the SiamPRA scheme, the BatchNormal (BN) layer coefficients are sparsified via sparsification training. Then, the structural pruning algorithm, full pruning (FP), optimizes the Siamese network presented in Figure 1 to retain the most important and effective convolution kernel. The number of channels reserved in the last layer of the ResNet feature map is taken as the number of convolution kernels reserved in the attention module. Finally, the visual targets are tracked according to the fine-tuned pruned network. The overall SiamPRA scheme is summarized in Algorithm 2.

Algorithm 2 Search for Pruned Structure

```
1. Input:
   • SiamRA Networks, pruning ratio $\varepsilon$;
2. Output:
   • SiamPRA model $T^*$ ;
3. Procedure:
   1: Train the SiamRA networks with the improved loss function Equation (3). Obtain the
        sparse BN layer $\gamma$;
   2: for $j$→to $J$ do
   3:   Sort BN layer weight $\gamma$ from largest to smallest to obtain $\Lambda = \{\gamma_1, \gamma_2, \ldots\ldots, \gamma_{0.9^{j-1}n}\}$;
   4:   Prune 10% to obtain new $\Lambda^* = \{\gamma_1, \gamma_2 \ldots\ldots, \gamma_{0.9^{j}n}\}$;
   5:   Obtain pruned model $T^*$ with $\Lambda^*$ and fine-tuning;
   6:   If pruning ratio $\varepsilon <= 1 - \mathrm{num}(\Lambda^*)/\,\mathrm{num}(\Lambda)$;
   7:     break;
   8:   end if;
   9: end for;
   10: Obtain the SiamPRA model $T^*$.
```

Figure 6 shows the network-structured pruning process based on the BN layer coefficients. First, sparsification training of the network based on spatial channel attention feature extraction is performed to obtain the trained network parameters. Then, the BN layer coefficients in the network parameters are sorted and the smaller part of the BN layer and its corresponding convolution kernel are deleted according to the pruning rate ε. Finally, fine tuning and pruning continue until the network performance decreases significantly. The network pruning process can be divided into the following three steps:

Step 1: Add the $L1$ norm of the constraint BN layer coefficient γ to the loss function in the training process to promote the sparseness of γ. (This is done because the $L1$ norm can make the constraint item approach zero while solving for the minimum value to sparse the parameters.) The loss equation [32] is

$$Loss = \arg\min_{\omega} \sum_{i} \log\left(1 + e^{-gt_i * T(X_i, Z_i)}\right) + \lambda \sum_{\gamma} |\gamma|. \tag{3}$$

The first term denotes the logistic loss function in Algorithm 1 and is used to ensure that the difference between the estimated value of the convolutional network reference function and the ground truth is sufficiently small. The second item denotes the $L1$ regular item that has the effect of sparse parameters and λ is the regularization constraint coefficient. In this paper, sparsification training is achieved by constraining γ; thus, most of γ is zero or approximately zero to achieve γ sparsity. This ensures that all the convolution kernels with minimal effects are pruned in subsequent pruning. There is low loss of accuracy after pruning, thus accelerating the pruning process.

This study adopts the calculation formula for pruning judgment based on the BN layer as follows:

$$z_{m+1} = \gamma_m \hat{z}_m + \beta_m; \hat{z}_m = \frac{z_m - \mu_m}{\sqrt{\sigma_m^2 + \epsilon_m}}, \tag{4}$$

where the BN layer has two trainable parameters: γ and β. The next layer input z_{m+1} is a line equation with the standard deviation $\hat{z}_m$ of the convolution calculation result of the current layer m as the independent variable, γ_m as the coefficient, and β_m as the bias.

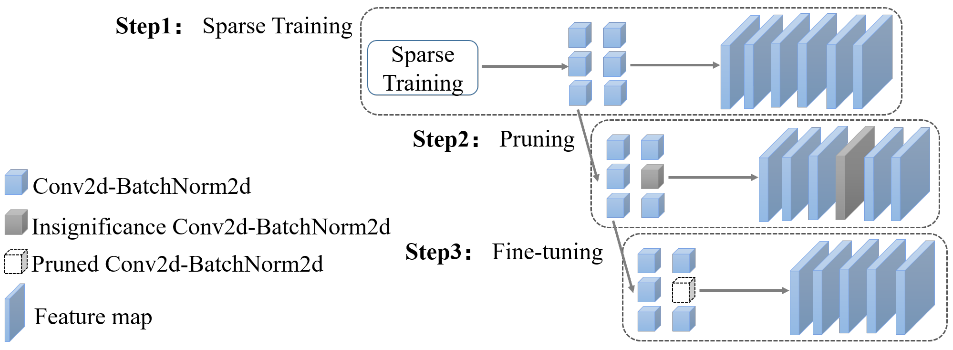


Figure 6. The structured pruning process.

Step 2: After the sparse training is completed, the trained model starts pruning. FP sorts the γ of all BN layers and prunes the corresponding ratio of the BN layer and its corresponding convolution kernel according to the pruning ratio ε from small to large.

Because the residual block in ResNet has a shortcut, the number of input channels must equal the number of output channels to sum the elements pairwise. Pruning directly deletes all the convolution kernels and BN layers below the threshold, as shown in the residual block pruning part in Figure 7. If the channels of different dimensions of the first layer of *Pruned_Feature_map_A* and the last layer of *Pruned_Feature_map_B* of the residual block are pruned, it may not be possible to sum elementwise owing to the unequal number of channels.

In this FP algorithm, a new solution to this problem is proposed. As shown in Figure 7, when *Pruned_Feature_map_A* and *Pruned_Feature_map_B* are pruned in the same channel, the corresponding weight of the BN layer is low. Therefore, as shown in Equation (5), for a given channel $C = i$, the new feature map generated by the shortcut of two feature maps approaching zero will also approach zero. Similarly, if any feature map is not zero, the new feature map is not necessarily zero. The channel approaching zero in the new feature map has no effect in the next convolution layer; therefore, it is pruned.

$$F_C(x,y,i) = \lim_{\substack{(x,y)\to 0,\\ c=i}} F_A(x,y,c) + \lim_{\substack{(x,y)\to 0,\\ c=i}} F_B(x,y,c)$$
$$= \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & 0 & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & 0 & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & 0 & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{5}$$

This method prunes the channel of the feature map close to zero generated by the residual block. As shown in Figure 7, the algorithm takes the union of the unpruned channels of the two feature maps. This prunes the unimportant BN layer and its corresponding convolution kernel under the same channel and obtains *Pruned_Feature_map_C*. According to the channels reserved by the previous residual block, the algorithm determines the same channels; the next residual block should be kept before pruning.

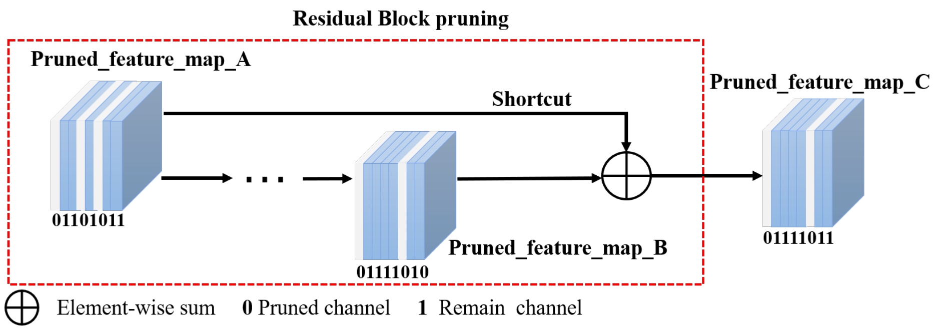


Figure 7. Residual block pruning strategy. "0" represents the pruned feature map, and "1" the retained feature map.

Step 3: Conducting fine-tuning. Loading all the parameters of the pruned model and then training it overall on the fine-tuned dataset. The network can be restored to its original performance by training only a few epochs (after pruning the network, the accuracy may temporarily decrease). The fine-tuned network may occasionally perform better than the original network during the experimental process of this FP algorithm.

5. Experiments

The visual navigation system of UAV is an important application scenario for the proposed algorithm. However, because the system involves multiple fields, in order to independently verify the performance of the proposed tracking algorithm without system interference, we only test the tracking algorithm objectively, subjectively, and on the computing platforms that can be mounted on UAVs for efficiency testing. Without specified, the experiments were conducted on a PC with 16 GB RAM, 3.40 GHz CPU, and NVIDIA 2080Ti GPU. All experimental frameworks were accelerated by PyTorch, CUDA 9.0, and cuDNN 7.6. The general dataset VOT2018 [33], OBT100 and the UAV image dataset UAV123 [34] were used to verify the effectiveness of the SiamPRA method.

5.1. Experimental Environment

5.1.1. Dataset Description

This study used the GOT10K dataset [35], TrackingNet dataset [36], and COCO dataset [37] as the training datasets. The datasets were cropped to obtain 255 × 255 search images and 127 × 127 template images which were input into the network as training images.

5.1.2. Evaluation Criteria

The objective indicators of the datasets VOT, UAV123 and OTB100 were used to evaluate the performance of the algorithm. Acc. in the VOT denotes the tracking accuracy calculated by the overlap rate between the predicted box and the ground truth box; Rob. denotes the stability of the tracking calculated by the number of tracking failures in the video; Expected Average Overlap (EAO) denotes the tracking expected average coverage rate calculated by Acc. and Rob., and is negatively correlated with the tracking failure time. An efficient algorithm generally has a low Rob. value and high EAO. and Acc. values. Prec. in UAV123 and OTB100 denotes the tracking accuracy calculated by the distance between the center of the predicted box and the center of the ground-truth box below the threshold (set as 20); Succ. denotes the success rate of the tracking calculated by IoU above the threshold (set as 0.6). FPS denotes the tracking speed calculated by the amount of tracking completed per second; MACs denotes the memory usage which is calculated per second; Params and model size denote the number of parameters and the size of the model, respectively.

5.1.3. Implementation Details

The pre-trained model on the ImageNet [38] dataset was used to initialize the ResNet in the SiamRA network and fine-tune the visual tracking task on the TrackingNet, GOT10K, and COCO datasets. Because the spatial-channel attention module is not part of the standard ResNet structure, there is no corresponding pre-training weight. Therefore, we initialized these two modules randomly. The input search images were randomly shifted within ±64 pixels and the template images within ±8 pixels. The COCO dataset was used to generate positive and negative datasets of the same type.

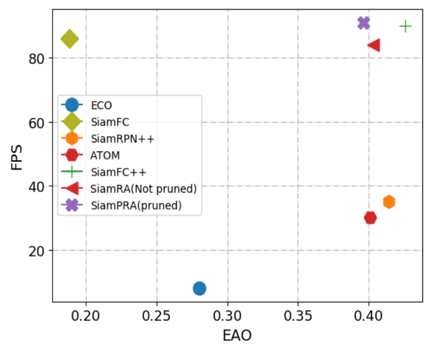
5.2. Comparison of System Effectiveness

Embedded vision systems have very limited computing power and storage resources, such as UAVs. Therefore, it is required that the algorithms ensure accuracy, reliability, and speed while minimizing the model size and computational complexity. The proposed method was compared with state-of-the-art methods in terms of accuracy, speed, and model size on the VOT2018 and UAV123 datasets to verify its effectiveness. As can be seen in Table 1, the proposed algorithms achieved the best performance in terms of robustness and running speed, respectively, and greatly reduces the model size and computational complexity while ensuring accuracy. The use of spatial-channel attention makes SiamPRA more robust. However, to reduce the number of parameters and the algorithm complexity, RPN and anchor-free algorithms are not used for bounding-box regression. This results in a lower overlap rate performance than those with bounding box regression. When using the UAV for tracking, these algorithms have high computational costs and more parameters, making it difficult to perform real-time operations on resource-constrained embedded platforms. The Model Size and the Params of SiamPRA are only 6 MB and 802.56 K, which is a great advantage. Among them, the 802.56 K Params are respectively composed of 749.1 K for the backbone network and 53.46 K for the spatial-channel attention module, which also proves that the attention mechanism can improve the performance of the network under the condition of low parameters.

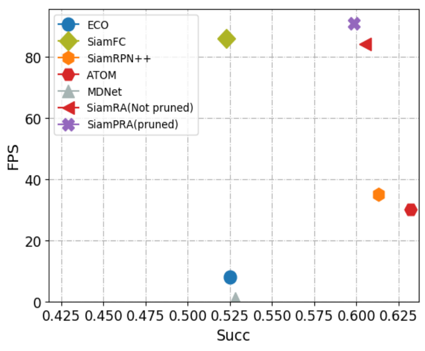
Figure 8 shows the advantages of the algorithms in this paper more visually, where the horizontal coordinates represent EAO/Succ and the vertical coordinates represent FPS/GMAC. For target tracking algorithms applied to UAV systems, it is necessary to improve the algorithm's speed and reduce the parameters while maintaining a relatively high accuracy rate. An algorithm with better overall performance should be in the upper right corner of the coordinate system, demonstrating that it can achieve high accuracy target tracking with high speed, low computation and storage space. From the Figure 8, it can be seen that SiamPRA achieves the best FPS and GMAC with high accuracy and is the algorithm with the best overall performance. To further evaluate the effectiveness of the proposed algorithm, this study conducted comparisons with mainstream methods on the UAV images in Figure 9. The results show that the proposed algorithm can effectively handle partial occlusion, small targets, and the influence of similar objects. This demonstrates that the SiamPRA can effectively solve the challenges in UAV image tracking and the performance is superior to that of the other algorithms.

Table 1. Comparison of the proposed method with state-of-the-art methods on the VOT2018, UAV123 and OTB100 datasets.

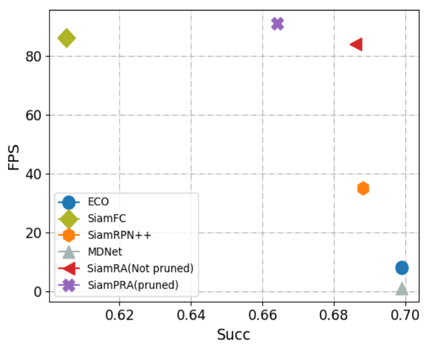
Dataset / Methods	VOT2018			UAV123		OTB100		FPS	MACs	Params	Model Size
	EAO	**Rob.**	**Acc.**	**Succ.**	**Prec.**	**Succ.**	**Prec.**				
MDNet [39]	-	-	-	0.528	-	0.699	0.932	1	0.26 G	4.43 M	366 MB
ECO [22]	0.28	0.276	0.484	0.525	0.741	0.699	0.913	8	-	-	329 MB
SiamFC [6]	0.188	0.585	0.503	0.523	0.731	0.605	0.799	86	3.2 G	2.34 M	8.9 MB
SiamRPN++ [9]	0.414	0.234	0.60	0.613	0.807	0.688	0.899	35	34.648 G	32.979 M	206 MB
ATOM [11]	0.401	0.204	0.59	0.632	0.844	-	-	30	-	-	109 MB
SiamFC++ [10]	0.426	0.183	0.587	-	-	0.688	0.911	90	19.793 G	13.887 M	53.14 MB
SiamRA	0.403	0.15	0.58	0.605	0.767	0.686	0.913	84	2.59 + 0.16 G	1.3 + 0.32 M	7 MB
SiamPRA	0.396	0.193	0.571	0.598	0.761	0.664	0.881	91	1.786 + 0.03 G	749.1 + 53.46 K	6 MB



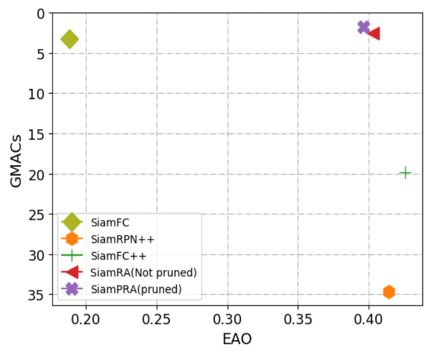
(**a**) FPS



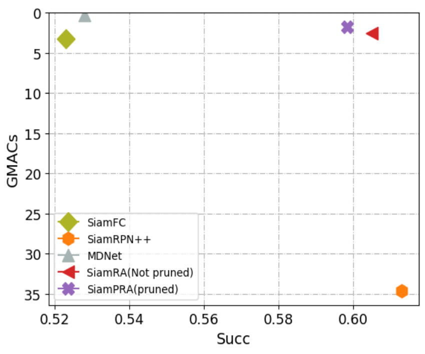
(**b**) FPS



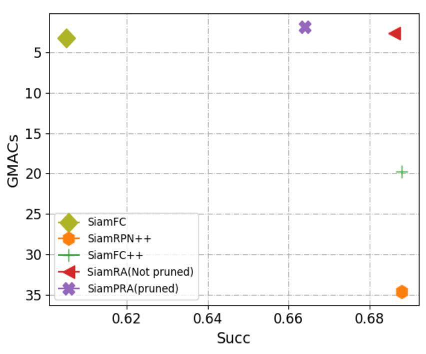
(**c**) FPS



(**d**) GMACs



(**e**) GMACs



(**f**) GMACs

Figure 8. Comparison results of FPS and GMACS on the VOT2018, UAV123 and OTB100 datasets. Column one represents the results from VOT2018, column two represents the results from UAV123, and column three represents the results from OTB100.

In addition, to better analyze the effectiveness and efficiency of the algorithm, experiments were conducted on the two most commonly used embedded platforms: FPGA and embedded GPU. The FPGA platform was Zynq UltraScale+MPSoC ZCU102 with XCZU9EG-2FFVB1156 CPU, which is suitable for industrial applications requiring high reliability. The experimental environment was Vitis 1.0, with Python 3.6, TensorFlow 1.12, OpenCV-Python, and OpenBLAS. The embedded GPU experiments were conducted on NVIDIA's Jetson Nano, Xavier NX, and Xavier AGX. The Jetson Nano can run at a low power consumption of 5 W, which is suitable for power-sensitive tasks in industrial scenarios. The Xavier AGX is a high-performance platform for scenarios requiring higher processing speeds. The power consumption and performance of the Xavier NX lie between those of the above two. The experimental environment was JetPack 4.5.1, with Python

3.6, PyTorch 1.7, and OpenCV 3.4.3. All experiments were conducted without engineering optimization. The results are shown in Table 2.

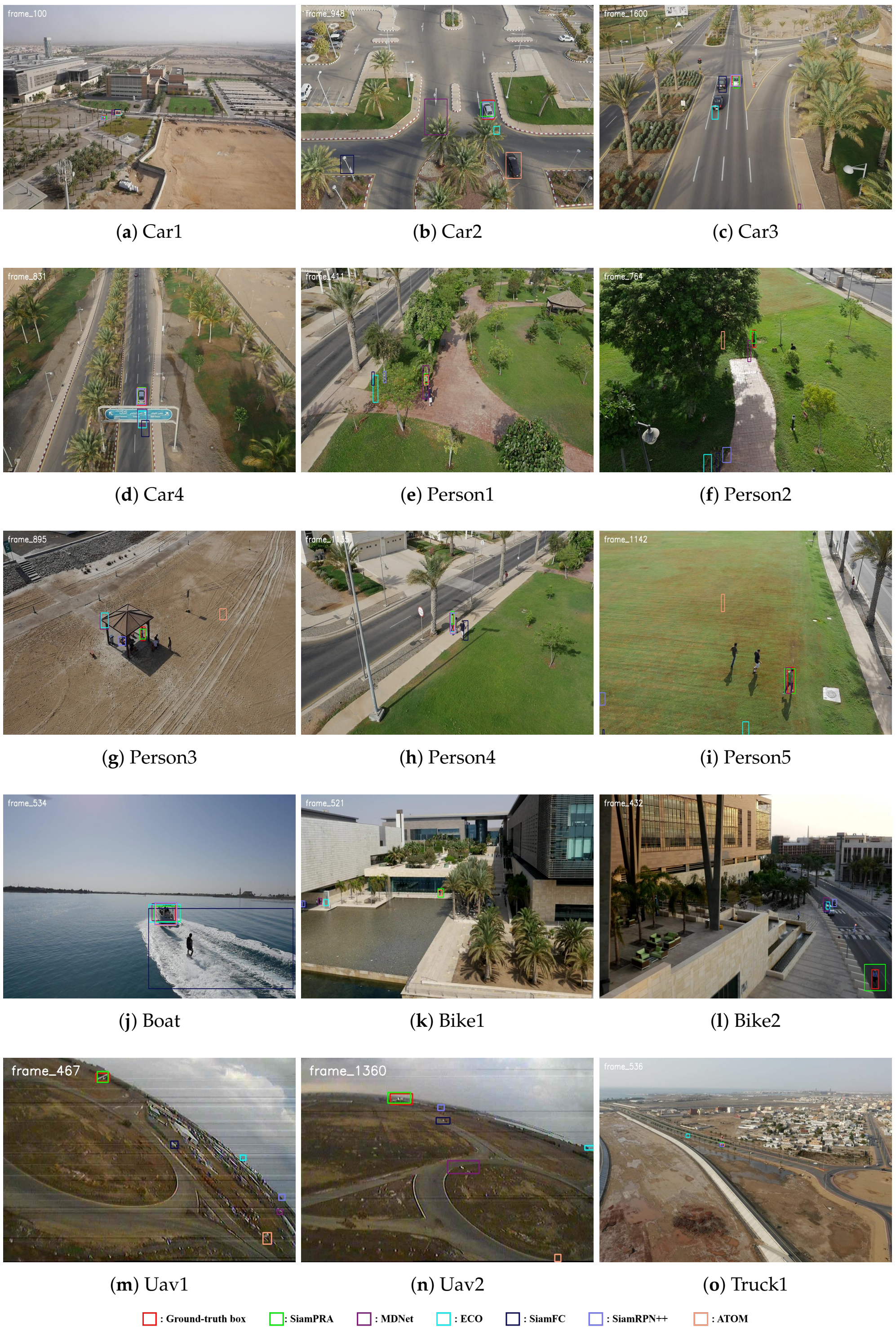


Figure 9. Comparison of target tracking results in UAV images. The targets in UAV images present challenges such as small size, similar targets, partial occlusion, object size change, camera motion, and fast movement.

In this study, the pre-processing, feature extraction, and post-processing of SiamPRA were tested separately in the embedded system. Among them, pre-processing mainly included the data loading and scaling modules, feature extraction included the backbone network and attention mechanism modules, and post-processing corresponded to the tracking result return and target localization modules. As can be seen in Table 2, the embedded GPU-based algorithm implementation was the most time-consuming in this module owing to it being the most computation-intensive in feature extraction. However, in the ZCU102 platform, the feature extraction part was parallelized by the programmable logic (PL) side, and thus took only 0.02 s. As shown in Figure 10, the SiamPRA algorithm consumed only approximately 26.64% of the total PL hardware resources on average. For the other two modules, ZCU102 used the programmable system (PS) side for data scheduling and processing; thus, it requires a longer time and has a high potential for speedup. In the above experiments, the Xavier AGX had the fastest average inference time of 0.057 s, achieving real-time processing speed without engineering optimization. The above experimental results and analysis demonstrate the effectiveness of SiamPRA in vision tracking tasks.

Table 2. Results of SiamPRA running on the embedded platform.

	ZCU102	Xavier-AGX	Xavier-NX	Jetson-Nano
Pre-processing	0.051 s (PS)	0.022 s	0.024 s	0.030 s
Features Extraction	0.020 s (PL)	0.029 s	0.030 s	0.034 s
Post-processing	0.092 s (PS)	0.006 s	0.009 s	0.012 s
Total time	0.163 s	0.057 s	0.063 s	0.076 s

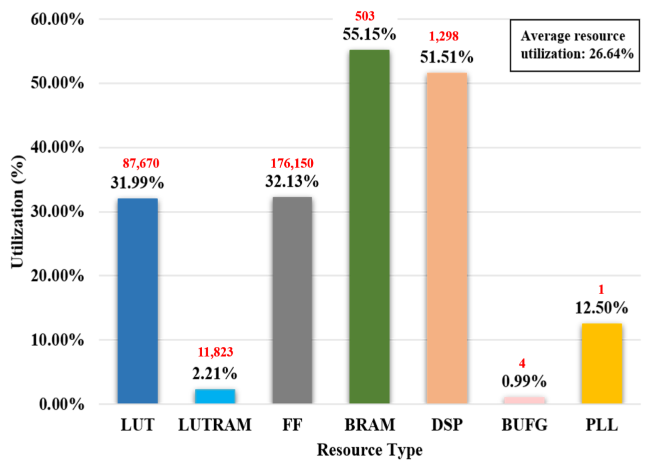


Figure 10. FPGA resource utilization of SiamPRA algorithm.

5.3. Network Architecture Performance Analysis

Ablation experiments on the attention module were conducted to ascertain the effectiveness of the network architecture. Because the proposed algorithm employs similarity-based target tracking, it requires the backbone network to have good translational invariance. The network without and with the attention module on the two datasets were compared by ablation studies to evaluate the effectiveness of the spatial attention module and the channel attention module. Table 3 shows the comparison results. Using SiamRA, which only contains the basic network ResNet, an EAO of 0.338 was obtained on the VOT2018 dataset. Adding the attention module after extracting the depth features, the EAO increased to 0.403, an improvement of 19.2%. This shows that spatial information and channel information can be aggregated through the spatial-channel attention module, thus focusing on the target area, reducing attention to similar scenes or similar targets, and improving the robustness.

Table 3. Comparisons of SiamRA for different feature extraction configurations.

Dataset Models	VOT2018			UAV123	
	EAO	Acc.	Rob.	Succ.	Prec.
SiamRA(ResNet)	0.338	0.587	0.245	0.543	0.701
SiamRA(channel attention)	0.344	0.576	0.23	0.564	0.715
SiamRA(spatial attention)	0.373	0.562	0.172	0.594	0.759
SiamRA(spatial-channel attention)	0.403	0.58	0.15	0.605	0.767

5.4. System Optimization Performance Analysis

The results before and after pruning were also compared to evaluate the effectiveness of network-structured pruning based on BN layer coefficients. The results without pruning the last residual block of the network and those without sparse training along with the experimental comparison are shown in Table 4.

Table 4. Comparisons of SiamPRA for different system optimization configurations.

	Sparse	Full Pruning	VOT2018		MACs	Ratio
			EAO	ΔEAO		
Original SiamRA	–	–	0.403	-	2.75 G	-
Sparse SiamRA	✓	✗	0.409	+0.006	2.75 G	-
Pruned SiamRA	✗	✗	0.329	−0.074	2.45 G	14%
Pruned SiamRA	✗	✓	0.375	−0.028	1.98 G	37%
SiamPRA	✓	✓	0.396	−0.007	1.82 G	44%

In this study, the $L1$ norm regular item of the BN coefficient γ in Equation (3) makes the γ coefficient sparse. By comparing no sparsity regularization ($\lambda = 0$) and sparsity regularization ($\lambda = 10^{-5}$) the sparse γ coefficient will fall into a small region near zero and the γ coefficients that are not pruned will be evenly distributed as shown in Figure 11. Therefore, the addition of the $L1$ regular item improves the pruning efficiency of the system for the convolution kernels and BN layers, accelerates the pruning speed, and increases the proportion of the final pruning. It can also be seen from Table 4 that the addition of the $L1$ norm regular item does not affect the accuracy of the tracking and results in a higher pruning rate and EAO after pruning.

Current pruning algorithms such as [17,32] assume that networks containing Residual Blocks such as MobileNet and ResNet need the same number of channels in the first and last layers to perform elementwise addition. Usually, the last layer of the residual block is not pruned. Experimental results show that as the degree of pruning deepens, this type of algorithm will cause the number of channels in some layers to become very small, resulting in a significant decline in tracking performance. Because the number of channels in the last layer of the residual block is four times that of previous channels, it is difficult to sufficiently optimize the amount of computation. It can be seen from Figure 12 that the network pruned by the FP method at a similar pruning ratio is smoother, without some channels being close to zero, and has a higher EAO.

It can be seen from Table 4 that the FP algorithm has a higher pruning ratio and accuracy. It can achieve a pruning ratio of 44% when the EAO is only reduced by 0.007. This indicates that the tracking algorithm after pruning through the FP method can have fewer calculations and parameters while maintaining high-accuracy tracking and can perform better on the embedded vision platform.

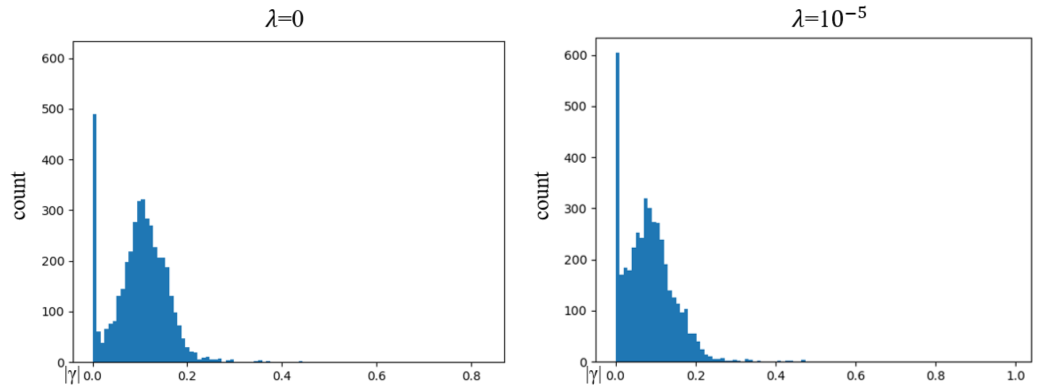


Figure 11. Statistical results of γ under $\lambda = 0$ and $\lambda = 10^{-5}$.

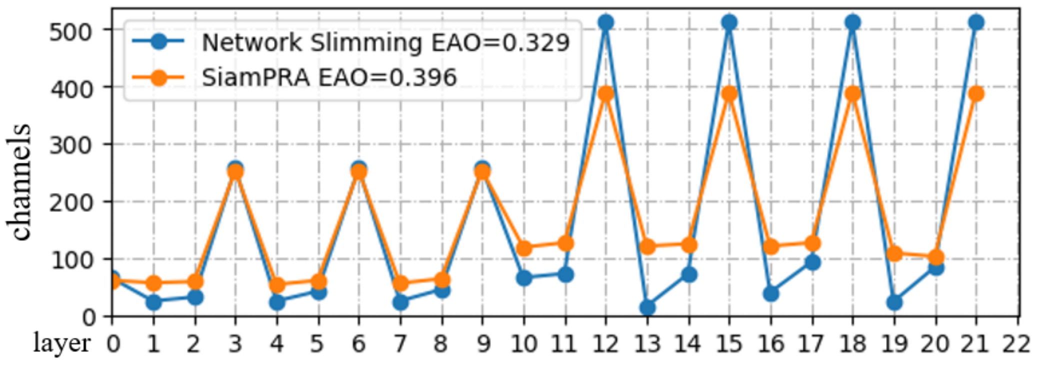


Figure 12. Remaining channels for Network Slimming and the SiamPRA algorithm.

6. Conclusions

This paper proposed a visual tracking algorithm architecture for visual navigation systems involving resource-constrained embedded platform in UAVs. The main contributions of this study are twofold. First, a spatial-channel attention mechanism is integrated into the Siamese-based visual tracking network for the first time. The attention mechanism enhances the feature extraction ability of the network, which in turn improves the tracking accuracy and robustness. Second, considering the limited resources of the embedded platform, this study proposes an algorithm that can fully prune the ResNet and reduce the computation and parameters of the tracking algorithm.

The limitation of SiamPRA lies in its boundary box estimation. Related research results show that although the RPN or anchor-free module can improve the bounding box efficiency, it can create high computing and memory overheads. In addition, the algorithm does not consider the application scenario of long-time tracking and does not introduce a recovery mechanism after tracking loss to improve accuracy. In future work, we will develop corresponding algorithm optimization methods for RPN or anchorless modules and introduce adaptive mechanisms to quickly re-detect the tracking target in case of tracking failure, achieving faster and more accurate for long-time target tracking for UAV images.

Author Contributions: Methodology, J.L., L.Y. and L.Z.; Software, K.Z.; Validation, Z.G. and L.Y.; Writing—original draft, J.L.; Writing—review & editing, J.L., K.Z. and Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the General Program of Beijing Municipal Education Commission under Grant KM202110005027, and in part by the Beijing Natural Science Foundation under Grant L211017, and in part by National Natural Science Foundation of China under Grant 61971016 and 61701011, and in part by Beijing Municipal Education Commission Cooperation Beijing Natural Science Foundation under Grant KZ201910005077.

Data Availability Statement: The datasets generated and analysed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiao, Y.; Kamat, V.R.; Menassa, C.C. Human tracking from single RGB-D camera using online learning. *Image Vis. Comput.* **2019**, *88*, 67–75. [CrossRef]
2. Zhu, M.; Zhang, H.; Zhang, J.; Zhuo, L. Multi-level prediction Siamese network for real-time UAV visual tracking. *Image Vis. Comput.* **2020**, *103*, 104002. [CrossRef]
3. Shan, Y.; Zhou, X.; Liu, S.; Zhang, Y.; Huang, K. SiamFPN: A deep learning method for accurate and real-time maritime ship tracking. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *31*, 315–325. [CrossRef]
4. Zhou, Z.; Zhang, C.; Chen, X.; Fei, X.; Umer, T. Energy-Efficient Industrial Internet of UAVs for Power Line Inspection in Smart Grid. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2705–2714. [CrossRef]
5. Qingbo, J.; Chao, R.; Lewei, Q.; Chang, W. The implementation of an improve infrared dim-small target track before detect based on DSP. In Proceedings of the 2013 International Conference on Machine Learning and Cybernetics (ICMLC), Tianjin, China, 14–17 July 2013; Volume 4, pp. 1514–1518.
6. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H.S. Fully-Convolutional Siamese Networks for Object Tracking. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 850–865.
7. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High Performance Visual Tracking with Siamese Region Proposal Network. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8971–8980.
8. Zhu, Z.; Wang, Q.; Li, B.; Wu, W.; Yan, J.; Hu, W. Distractor-aware Siamese Networks for Visual Object Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 101–117.
9. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4282–4291.
10. Xu, Y.; Wang, Z.; Li, Z.; Yuan, Y.; Yu, G. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 12549–12556. [CrossRef]
11. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. ATOM: Accurate Tracking by Overlap Maximization. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4655–4664.
12. Yang, M.; Yuan, J.; Wu, Y. Spatial selection for attentional visual tracking. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
13. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
14. Fu, J.; Liu, J.; Tian, H.; Li, Y.; Bao, Y.; Fang, Z.; Lu, H. Dual Attention Network for Scene Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3146–3154.
15. Wu, Y.; Liu, Z.; Zhou, X.; Ye, L.; Wang, Y. ATCC: Accurate tracking by criss-cross location attention. *Image Vis. Comput.* **2021**, *111*, 104188. [CrossRef]
16. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
17. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the Value of Network Pruning. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
18. Vojir, T.; Noskova, J.; Matas, J. Robust scale-adaptive mean-shift for tracking. *Pattern Recognit. Lett.* **2014**, *49*, 250–258. [CrossRef]
19. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 583–596. [CrossRef] [PubMed]
20. Lukezic, A.; Vojir, T.; Čehovin, L.; Matas, J.; Kristan, M. Discriminative Correlation Filter with Channel and Spatial Reliability. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4847–4856.
21. Bertinetto, L.; Valmadre, J.; Golodetz, S.; Miksik, O.; Torr, P. Staple: Complementary Learners for Real-Time Tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1401–1409.
22. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. ECO: Efficient Convolution Operators for Tracking. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6931–6939.
23. Zhang, Z.; Peng, H. Deeper and wider siamese networks for real-time visual tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4591–4600.

24. Nousi, P.; Tefas, A.; Pitas, I. Dense convolutional feature histograms for robust visual object tracking. *Image Vis. Comput.* **2020**, *99*, 103933. [CrossRef]
25. Tan, H.; Zhang, X.; Zhang, Z.; Lan, L.; Zhang, W.; Luo, Z. Nocal-Siam: Refining Visual Features and Response with Advanced Non-Local Blocks for Real-Time Siamese Tracking. *IEEE Trans. Image Process.* **2021**, *30*, 2656–2668. [CrossRef] [PubMed]
26. Tang, F.; Ling, Q. Learning to Rank Proposals for Siamese Visual Tracking. *IEEE Trans. Image Process.* **2021**, *30*, 8785–8796. [CrossRef] [PubMed]
27. Yao, S.; Han, X.; Zhang, H.; Wang, X.; Cao, X. Learning Deep Lucas-Kanade Siamese Network for Visual Tracking. *IEEE Trans. Image Process.* **2021**, *30*, 4814–4827. [CrossRef] [PubMed]
28. Yang, K.; He, Z.; Pei, W.; Zhou, Z.; Li, X.; Yuan, D.; Zhang, H. SiamCorners: Siamese Corner Networks for Visual Tracking. *arXiv* **2021**, arXiv:2104.07303.
29. Zhou, W.; Wen, L.; Zhang, L.; Du, D.; Luo, T.; Wu, Y. SiamCAN: Real-Time Visual Tracking Based on Siamese Center-Aware Network. *IEEE Trans. Image Process.* **2021**, *30*, 3597–3609. [CrossRef] [PubMed]
30. Zhao, F.; Zhang, T.; Song, Y.; Tang, M.; Wang, X.; Wang, J. Siamese Regression Tracking with Reinforced Template Updating. *IEEE Trans. Image Process.* **2021**, *30*, 628–640. [CrossRef] [PubMed]
31. Chan, S.; Tao, J.; Zhou, X.; Bai, C.; Zhang, X. Siamese Implicit Region Proposal Network with Compound Attention for Visual Tracking. *IEEE Trans. Image Process.* **2022**, *31*, 1882–1894. [CrossRef] [PubMed]
32. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning Efficient Convolutional Networks through Network Slimming. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2755–2763.
33. Kristan, M.; Leonardis, A.; Matas, J.; Felsberg, M.; Pflugfelder, R.; Zajc, L.Č.; Vojír, T.; Bhat, G.; Lukežič, A.; Eldesokey, A.; et al. The Sixth Visual Object Tracking VOT2018 Challenge Results. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
34. Mueller, M.; Smith, N.; Ghanem, B. A Benchmark and Simulator for UAV Tracking. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 445–461.
35. Huang, L.; Zhao, X.; Huang, K. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 1562–1577. [CrossRef] [PubMed]
36. Müller, M.; Bibi, A.; Giancola, S.; Al-Subaihi, S.; Ghanem, B. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 310–327.
37. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
38. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
39. Nam, H.; Han, B. Learning Multi-domain Convolutional Neural Networks for Visual Tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4293–4302.