ORIGINAL PAPER

Multiple convolutional features in Siamese networks for object tracking

Zhenxi Li[1] · Guillaume-Alexandre Bilodeau[1] · Wassim Bouachir[2]

Abstract

Siamese trackers demonstrated high performance in object tracking due to their balance between accuracy and speed. Unlike classification-based CNNs, deep similarity networks are specifically designed to address the image similarity problem and thus are inherently more appropriate for the tracking task. However, Siamese trackers mainly use the last convolutional layers for similarity analysis and target search, which restricts their performance. In this paper, we argue that using a single convolutional layer as feature representation is not an optimal choice in a deep similarity framework. We present a Multiple Features-Siamese Tracker (MFST), a novel tracking algorithm exploiting several hierarchical feature maps for robust tracking. Since convolutional layers provide several abstraction levels in characterizing an object, fusing hierarchical features allows to obtain a richer and more efficient representation of the target. Moreover, we handle the target appearance variations by calibrating the deep features extracted from two different CNN models. Based on this advanced feature representation, our method achieves high tracking accuracy, while outperforming the standard siamese tracker on object tracking benchmarks.The source code and trained models are available at https://github.com/zhenxili96/MFST.

1 Introduction

Visual object tracking (VOT) is a fundamental task in computer vision. Given a target object in the first frame, the objective of VOT is to determine the object state, typically its bounding box, in the following frames. With the rapid development of computer vision, visual object tracking has been employed in many applications, such as autonomous driving, visual analysis and video surveillance. For example, with the help of visual object tracking, autonomous driving systems can analyze obstacle movements and decide where to go.

Nowadays, most successful state-of-the-art trackers are based on correlation filters (e.g., [31]), deep neural networks

(e.g., [24]) or on a combination of both techniques (e.g., [29]). In this work, we are particularly interested in deep learning trackers, that achieved impressive performance while bringing new ideas to VOT. This paradigm has become successful mainly due the use of convolutional neural network (CNN)-based features for appearance modeling and their discriminative ability to represent target objects. While several tracking methods use classification-based CNN models that are built following the principals of visual classification tasks, another approach [2] formulates the tracking task as a deep similarity learning problem, where a Siamese network is trained to locate the target within a search image region. This method uses feature representations extracted by CNNs and performs correlation operation with a sliding window to calculate a similarity map for finding the target location. Rather than detecting by correlation, other deep similarity trackers [7,16,28,32] generate the bounding box for the target object with regression networks. For example, GOTURN [7] predicts the bounding box of the target object with a simple CNN model. The trackers [32] and [16] generate a number of proposals for the target after extracting feature representations. Classification and regression procedures are then applied to produce the final object location. The SPLT tracker [28] uses

✉ Guillaume-Alexandre Bilodeau
   gabilodeau@polymtl.ca

   Zhenxi Li
   zhenxili96@gmail.com

   Wassim Bouachir
   wassim.bouachir@teluq.ca

[1] LITIV lab, Polytechnique Montreal, Montreal H3T 1J4, Canada

[2] TELUQ University, Montreal H2S 3L5, Canada

a similar approach, but includes also a re-detection module for long-term tracking.

By formulating object tracking as a deep similarity learning problem, Siamese trackers achieved significant progress in terms of both speed and accuracy. However, one weakness of the siamese trackers is that they typically use only features from the last convolutional layers for similarity analysis and target state prediction. Therefore, the object representation is not as robust as it could be to target appearance variations, and tracking can be lost in more difficult scenarios. To address this weakness, we argue that using the last convolutional layers is not the optimal choice, and we demonstrate in this work that features from earlier layers are also beneficial for more accurate tracking with Siamese trackers.

Indeed, the combination of several convolutional layers was shown to be efficient for robust tracking [19,20]. As we go deeper in a CNN, the receptive field becomes wider, therefore, features from different layers contain different levels of information. In this way, the last convolutional layers retain general characteristics represented in a summarized fashion, while the first convolutional layers provide low-level features. These latter are extremely valuable for precise localization of the target as they are more object-specific and capture spatial details. Furthermore, instead of using features from a single CNN model, we propose to exploit different models within the deep similarity framework. Diversifying feature representations significantly improves tracking performance. Such strategy is shown to ensure a better robustness against target appearance variations, one of the most challenging tracking difficulties [18].

Based on these principles, we propose a Multiple Features-Siamese Tracker (MFST). Our tracker utilizes diverse features from several convolutional layers, two models and a proper feature fusing strategies to improve tracking performance. Our contributions can be summarized as follows:

- We propose a new tracking method that exploits feature representations from several hierarchical convolutional layers as well as different CNN models for object tracking.
- We propose feature fusing strategies with a feature recalibration module to make a better use of the feature representations.
- We show that our two previous contributions improve tracking by testing our MFST tracking algorithm on popular OTB benchmarks. We show that our method improves over the SiamFC base model and that our method achieves strong performance with respect to recent state-of-the-art trackers on popular OTB benchmarks.

The paper is organized as follows. We present the related work in Sect. 2, the proposed MFST tracker in Sect. 3, and the experimental results in Sect. 4 respectively. Finally, Sect. 5 concludes the paper.

2 Related work

2.1 Siamese trackers

VOT can be formulated as a similarity learning problem. Once the deep similarity network is trained during an offline phase to learn a general similarity function, the model is applied for online tracking by analyzing the similarity between the two network inputs: the target template and the current frame. The pioneering work, SiamFC [2], applied two identical branches made up of fully convolutional neural networks to extract the feature representations, on which cross-correlation is computed to generate the tracking result. SiamFC outperformed most of the best trackers at that time, while achieving real-time speed. Rather than performing correlation on deep features directly, CFNet [23] trains a correlation filter based on the extracted features of the object to speed up tracking without accuracy drop. MBST [18] improved the tracking performance by using multiple siamese networks as branches to enhance the diversity of the feature representation. SA-Siam [6] encodes the target by a semantic branch and an appearance branch to improve the robustness of tracking. However, since these siamese trackers only take the output of the last convolutional layers, more-detailed target specific information from earlier layers is not used. In contrast, in our work, we adopt a Siamese architecture to extract deep features for the target and search region, but combine features from different layers of the networks for tracking.

2.2 Hierarchical convolutional features in tracking

Most CNN-based trackers only use the output of the last convolutional layer that contains semantic information represented in a summarized fashion. However, different convolutional layers embed different levels of visual abstraction. In fact, convolutional layers provide several detail levels in characterizing an object, and the combination of different convolutional levels is demonstrated to be efficient for robust tracking [17,19]. In this context, the pioneering algorithm, HCFT [19], tracks the target using correlation filters learned on several layers. With HCFT, the representation ability of hierarchical convolutional features is demonstrated to be better than features from a single layer. Subsequently, [20] presented a visualization of features extracted from different convolutional layers. In their work, they employed three convolutional layers as the target object representations, which are then convolved with the learned correlation filters to generate the response map and a long-term memory fil-
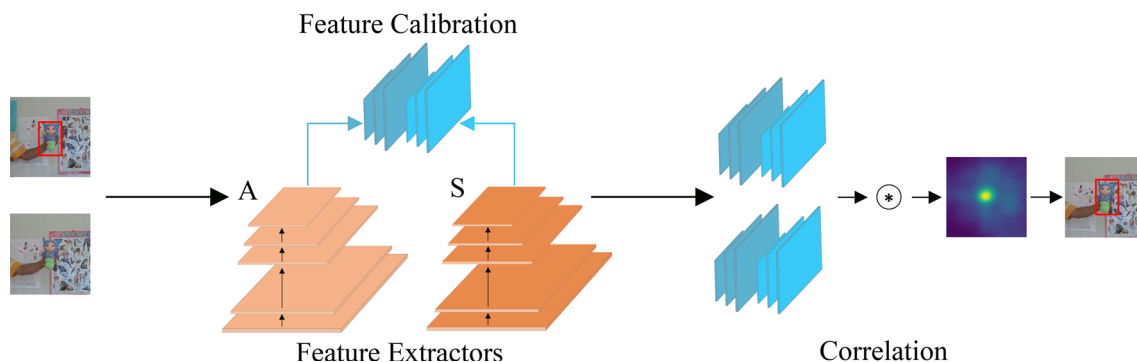
Fig. 1 The architecture of our MFST tracker. Two CNN models are used as feature extractors, and their features are calibrated by Squeeze-and-Excitation (SE) blocks. Then, correlations are applied over the features of the search region with the features of the exemplar patch, and the output response maps are fused to calculate the new position of the target. Bright orange and blue: SiamFC (S) and dark orange and blue: AlexNet (A)

ter to correct results. The use of hierarchical convolutional features is shown to make their trackers much more robust. In a similar way, the SiamRPN++ [15] tracker uses features from several layers of a very deep network to regress the target location. The regression results obtained with several SiamRPN blocks [16], applied each on a selected layer, are combined to obtain the final object location.

2.3 Multi-branch tracking

One of the most challenging problem in object tracking is the varying appearance of the tracked objects. A single fixed networks cannot guarantee to generate discriminative feature representations in all tracking situations. To handle the problem of target appearance variations, TRACA [3] trained multiple auto-encoders, each for different appearance categories. These auto-encoders compress the feature representation for each category. The best expert auto-encoder is selected by a pretrained context-aware network. By selecting a specific auto-encoder for the tracked object, a more robust representation can be generated. MDNet [22] applied a fixed CNN for feature extraction, but used multiple regression branches for objects belonging to different tracking scenarios. More recently, MBST [18] extracted the feature representation for the target object through multiple branches and selected the best branch according to their response maps. With multiple branches, MBST can obtain diverse feature representations and select the most discriminative one under the prevailing circumstance. In their study, we can observe that the greater the number of branches, the more robust the tracker is. However, this is achieved at the cost of a higher computational time. In this work, we can get a diverse feature representation of a target at lower cost because some of the representations are extracted from the many layers of the same CNN. Therefore, we do not need a large number of siamese branches.

3 Multiple features-siamese tracker

We propose a Multiple Features-Siamese Tracker (MFST) for object tracking. For the design of our method, we considered that features from different convolutional layers contain different level of abstractions and that the different channels of the features play different roles in tracking. Furthermore, we recalibrate the deep features extracted from the CNN models and combine hierarchical features to make a more robust representation. Besides, since models trained for different tasks can diversify the feature representation as well, we build our siamese architecture with two CNN models to achieve better performance. The code of our tracker can be found at https://github.com/zhenxili96/MFST.

3.1 Network architecture

As many recent object tracking approaches [2,18,23], we formulate the tracking problem as a similarity learning problem and utilize a siamese architecture to address it. The network architecture of our tracker is shown in Fig. 1. It uses two pretrained CNN models as feature extractors, SiamFC [2] and AlexNet [14], as indicated in Fig. 1. The two models are denoted as S and A, respectively, in the following. Both of them are five layers fully convolutional neural networks.

The input of our method consists of an exemplar patch z cropped according to the initial bounding box or the result of last frame and search region x. The exemplar patch has a size of $W_z \times H_z \times 3$, and the search region has a size of $W_x \times H_x \times 3$ ($W_z < W_x$ and $H_z < H_x$), representing the width, height and the color channels of the image patches.

With the two CNN models, we obtain the deep features S_{l_i}, A_{l_i} ($l = c3, c4, c5, i = z, x$) from the conv3, conv4 and conv5 layers of each model. These are the preliminary deep feature representations of the inputs. Then, these features are recalibrated through Squeeze-and-Excitation blocks (SE-
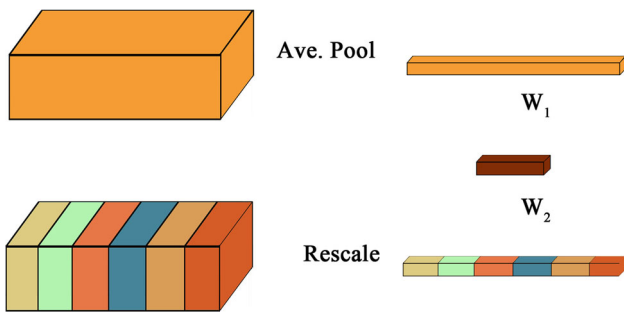
Fig. 2 Illustration of a SE-block. It consists of two step, squeeze step and excitation step. The squeeze step uses average pooling operation to generate the channel descriptor, and the excitation step uses a two layers MLP to capture the channel-wise dependencies

blocks) [10]. The recalibrated features are denoted as $S_{l_i}^*$, $A_{l_i}^*$, respectively, for the two models. The details of a SE-block are illustrated in Fig. 2. These blocks are trained to explore the importance of the different channels for tracking. They learn weights for the different channels to recalibrate features extracted from the preliminary feature representations.

Once the recalibrated feature representations $S_{l_i}^*$ and $A_{l_i}^*$ are generated, we apply cross-correlation operations for each recalibrated feature map pairs to generate response maps. The cross-correlation operation can be implemented by a convolution layer using the features of the exemplar as a filter. Then we fuse these response maps to produce the final response map. The corresponding location of the maximum value in the response map is the new position of the target object.

Similarly to [2], the SiamFC feature extractor, as well as the SE-blocks for both feature extractors are trained with a logistic loss. For a pair of patches z, x, the total loss for a response map r is

$$L(y, v) = \frac{1}{|r|} \sum_{u \in r} l(y[u].v[u]), \tag{1}$$

with

$$l(y, v) = log(1 + exp(-yv)), \tag{2}$$

where y is a ground truth label (1 or -1, for positive and negative pairs) and v is a cross-correlation score at coordinate u in response map r.

3.2 Feature extraction

Hierarchical Convolutional Features. It is well known that the last convolutional layer encodes more semantic information that is invariant to significant appearance variations, compared to earlier layers. However, its resolution is coarse due to the large receptive field, and it is not the most appropriate for precise localization as required in tracking. On the

contrary, features from earlier layers contain less semantic information, but they retain more spatial details and they are more precise in localization. Thus, we propose to exploit multiple hierarchical levels of features to build a better representation of the target.

In our work, we use the convolutional layers of two CNN models as feature extractors, that is SiamFC [2] and AlexNet [14]. Each model is trained for a different task, object tracking for SiamFC and image classification for AlexNet. We take features extracted from the 3rd, 4th, 5th layers as the preliminary target representations.

Feature Recalibration. Considering that different channels of deep features play different roles in tracking, we apply SE-blocks [10] over the raw deep features extracted with the base feature extractors. An illustration of a SE-block is shown in Fig. 2. The SE-block consists of two steps: 1) *squeeze* and 2) *excitation*. The *squeeze* step corresponds to an average pooling operation. Given a 3D feature map, this operation generates the channel descriptor ω_{sq} with

$$\omega_{sq} = \frac{1}{W \times H} \sum_{m=1}^{W} \sum_{n=1}^{H} v_c(m, n), (c = 1, ..., C), \tag{3}$$

where W, H, C are the width, height and the number of channels of the deep feature, and $v_c(m, n)$ is the corresponding value in the feature map. The subsequent step is the *excitation* through a two-layer multi-layer perceptron (MLP). Its goal is to capture the channel-wise dependencies that can be expressed as

$$\omega_{ex} = \sigma(\mathbf{W_2}\delta(\mathbf{W_1}\omega_{sq})), \tag{4}$$

where σ is a sigmoid activation, δ is a ReLU activation, $\mathbf{W_1} \in \mathbb{R}^{\frac{C}{b} \times C}$ and $\mathbf{W_2} \in \mathbb{R}^{C \times \frac{C}{b}}$ are the weights for each layer, and b is the channel reduction factor used to change the dimension. After the *excitation* operation, we obtain the channel weight ω_{ex}. The weight is used to rescale the feature maps extracted by the base feature extractors with

$$F_{l_i}^* = \omega_{ex} \cdot F_{l_i}, \tag{5}$$

where $\cdot$ is a channel-wise multiplication and $F = (S, A)$. Note that ω_{ex} is learned for each layer in a base feature extractor, but the corresponding layers for the CNN branches of the exemplar patch and the search region share the same channel weights. We train the SE-blocks to obtain six ω_{ex} in total (see Fig. 1).

3.3 Response maps combination

Once the recalibrated feature representations from the convolutional layers of each model are obtained, we apply a

cross-correlation operation, which is implemented by convolution, over the corresponding feature maps to generate the response map r with

$$r(z, x) = corr(F^*(z), F^*(x)), \qquad (6)$$

where F^* is a recalibrated feature map from SiamFC or AlexNet.

The response maps are then combined. For a pair of image inputs, six response maps are generated, denoted as r_{c3}^S, r_{c4}^S, r_{c5}^S, r_{c3}^A, r_{c4}^A and r_{c5}^A. Note that we do not need to rescale the response maps for combination, since they have the same size (see Sect. 4.1, Data Dimensions). The response maps are combined hierarchically. After fusing r^S and r^A for each of the CNN models, we combine the two resulting response maps to get the final map. The combination is performed by considering three strategies: hard weight (HW), soft mean (SM) and soft weight (SM) [20], defined as

$$\text{Hard weight: } r^* = \sum_{t=1}^{N} w_t r_t, \qquad (7)$$

$$\text{Soft mean: } r^* = \sum_{t=1}^{N} \frac{r_t}{max(r_t)}, \qquad (8)$$

$$\text{Soft weight: } r^* = \sum_{t=1}^{N} \frac{w_t r_t}{max(r_t)}, \qquad (9)$$

where r^* is the combined response map, N is the number of response maps to be combined together, and w_t is an empirical weight for each response map.

The optimal weights w_t for HW and SW are obtained experimentally. Finally, the corresponding location of the maximum value in the final response map is the new location of the target.

4 Experiments

The first objective of our experiments is to investigate the contribution of each module in order to find the best response map combination strategy for optimal representations. For this purpose, we perform an ablation analysis. Secondly, we compare our method with the reference SiamFC method and recent state-of-the-art trackers. The experimental results show that our method significantly outperforms SiamFC, while obtaining competitive performance with respect to the recent state-of-the-art trackers.

We performed our experiments on a PC with an Intel i7-3770 3.40 GHz CPU and a Nvidia Titan X GPU. We benchmarked our method on the OTB benchmarks [26] and on the VOT2018 benchmark [11]. The benchmark results are

calculated using the provided toolkits. The average testing speed of our tracker is 39 fps.

4.1 Implementation details

Network Structure. We used SiamFC [2] and AlexNet [14] as deep feature extractors. The SiamFC network is a fully convolutional neural network, containing five convolutional layers. It has an AlexNet-like architecture, but it is trained on a video dataset for object tracking. The AlexNet network consists of five convolutional layers and three fully connected layers trained on an image classification dataset. We slightly modified the stride of AlexNet to obtain the same dimensions for the outputs of both CNN models. Since only deep features are needed to represent the target, we removed the fully connected layers of AlexNet and only kept the convolutional layers to extract features.

Data Dimensions. The inputs of our method are the exemplar patch z and the search region x. The size of z is 127×127 and the size of x is 255×255. The output feature maps of z have sizes of $10 \times 10 \times 384$, $8 \times 8 \times 384$ and $6 \times 6 \times 256$ respectively. The output feature maps of x have sizes of $26 \times 26 \times 384$, $24 \times 24 \times 384$ and $22 \times 22 \times 256$, respectively. Taking the features of z as filters to perform a convolution on the features of x, the size of the output response maps are all the same, 17×17. The final response map is resized to the size of the input to locate the target. Since the two feature extractors that we are using are fully convolutional neural networks, the size of inputs can also be adapted to any other dimension.

Training. The SiamFC model is trained on the ImageNet dataset [4] and only color images are considered. The ImageNet dataset contains more that 4,000 video sequences with about 1.3 million frames and 2 million tracked objects with ground truth bounding boxes. For the input, we take a pair of images and crop the exemplar patch z in the center and the search region x in another image. The SiamFC model is trained with the loss of Eq. 1 for 50 epochs with an initial learning rate of 0.01. The learning rate decays with a factor of 0.86 after each epoch. The AlexNet model is pretrained on the ImageNet dataset for the image classification task. We just remove the fully connected layers before training the SE-blocks.

After the training of the base feature extractors, we add the SE-blocks in the two models and train them separately in the same manner. For each model, the original parameters are fixed. We then apply SE-blocks on the output of each selected layer (c3, c4 and c5) and take the recalibrated output of each layer as the output feature maps to generate the result for training. The SE-blocks are trained with the videos of the ImageNet dataset with the loss of Eq. 1 for 50 epochs with an initial learning rate of 0.01. The learning rate decays with a factor of 0.86 after each epoch.

Tracking. We first initialize our tracker with the initial frame and the coordinates of the bounding box of the target object. After we scaled and cropped the initial frame and obtained the exemplar patch, it is fed into the SiamFC model and AlexNet model to generate the preliminary feature representations S_{l_z}, A_{l_z} ($l = c3, c4, c5$). Then, the SE-blocks are applied to produce the recalibrated feature maps $S_{l_z}^*$, $A_{l_z}^*$, which are then used to produce response maps for tracking the target object for all the following frames.

After the feature maps of the target object are obtained, to track the target, the next frame is fed into the tracker. The tracker crops the region centered on the last center position of the target object, generate the feature representations and output the response maps by a correlation operation with the feature maps of the target object. The corresponding position of the maximum value in the final combined response map indicates the center point of the new position of the target object and the bounding box keeps the same size unless other scales obtain higher response value.

Hyperparameters. The channel reduction factor b in the SE-blocks is 4. The empirical weights w_t for r_{c3}^S, r_{c4}^S, r_{c5}^S, r_{c3}^A, r_{c4}^A and r_{c5}^A are 0.1, 0.3, 0.7, 0.1, 0.6 and 0.3. The empirical weights w_t for r^S and r^A are 0.3 and 0.7. To handle scale variations, we search the target object over three scales $1.025^{\{-1,0,1\}}$ during evaluation and testing.

4.2 Dataset and evaluation metrics

OTB Benchmarks. We evaluate our method on the OTB benchmarks [26,27], which consist of three datasets, OTB50, OTB2013 and OTB100. They contain 50, 51, 100 video sequences with ground truth target labels for object tracking. Two evaluation metrics are used for quantitative analysis, the center location error and the overlap score, which are used to produce precision plots and success plots respectively. To obtain the precision plot, we calculate the average euclidean distance between the center location of the tracking results and the ground truth labels. The threshold of 20 pixels is used to rank the results. For the success plot, we compute the IoU (intersection over union) between the tracking results and the ground truth labels for each frame. The AUC (area-under-curve) is used to rank the results.

VOT 2018 Benchmark. VOT2018 short-term benchmark consists of 60 video sequences, the target in the sequences are annotated by a rotated bounding box. The benchmark takes three primary measures to evaluate the tracking performance: accuracy (A), robustness (R) and expected average overlap (EAO). The accuracy is calculated by the average overlap between the tracker predictions and the ground truth bounding boxes, while the robustness is how many times the target get lost during tracking. The last evaluation metric, expected average overlap, measures the expected average overlap of a

tracker when given sequences with the same visual properties. The VOT benchmark utilizes a reset-based methodology, which means that the tracker is re-initialized when its prediction has no overlap with the ground truth.

4.3 Ablation study

To investigate the contributions of each module and the optimal strategies to combine representations, we performed an ablation study with several variations of our method. We first studied the combination strategy that achieves the best performance on the OTB benchmarks to generate the combined response maps for each model, which are denoted as r^S and r^A (see Table 1). After that, as illustrated in Table 2, we test the three different strategies again to find the best strategy to combine r^S and r^A.

A proper combination of features is better than features from single layer. As illustrated in Table 1, we experimented using features from a single layer as the target representation and combined features from several layers with different combination strategies for the two CNN models. The results show that, taken separately, c3, c4, c5 give results that are approximately similar. Since object appearance changes, c3 that should give the most precise location does not always achieve good performance. However, with a proper combination, the representation power of the combined feature gets much improved.

Features get enhanced with recalibration. Due to the *squeeze* and *excitation* operations, recalibrated features achieves better performance than the preliminary features. Recalibration through SE-blocks thus improves the representation power of features from single layer, which results in a better representation of the combined features.

Multiple models are better than a single model. Our approach utilizes two CNN models as feature extractors. Therefore, we also conducted experiments to verify the benefit of using two CNN models. As illustrated in Table 2, we evaluated the performance of using each CNN model separately and using the combination of two CNN models. The results show that the combination of two models is more discriminative than only one model regardless of the use SE-blocks.

A proper strategy is important for the response map combination. We applied three strategies to combine the response maps: hard weight (HW), soft mean (SM) and soft weight (SW). Since the two CNN models are trained for different tasks and features from different layers embed different level of information, different types of combination strategies should be applied to make the best use of the features. The experimental results show that generally, combined features are more discriminative than independent features, while a proper strategy can improve the performance significantly

Table 1 Experiments with several variations of our method, where A and S denote using AlexNet or using SiamFC as the base feature extractor. **Boldface** indicates best results

Model	Layers				SE	OTB-2013		OTB-50		OTB-100	
	c3	c4	c5	Fusion		AUC	Prec.	AUC	Prec.	AUC	Prec.
A	✓					0.587	0.740	0.474	0.618	0.559	0.712
A	✓				✓	0.603	0.755	0.504	0.642	0.587	0.747
A		✓				0.632	0.789	0.536	0.692	0.614	0.778
A		✓			✓	**0.637**	0.801	0.544	0.707	0.623	0.795
A			✓			0.582	0.763	0.496	0.665	0.557	0.735
A			✓		✓	0.573	0.762	0.507	0.696	0.575	0.769
A	✓	✓	✓	HW		0.623	0.774	0.515	0.657	0.605	0.763
A	✓	✓	✓	SM		0.633	0.797	0.542	0.705	0.616	0.784
A	✓	✓	✓	SW		0.630	0.795	0.538	0.699	0.616	0.786
A	✓	✓	✓	HW	✓	0.627	0.798	0.537	0.700	0.617	0.790
A	✓	✓	✓	SM	✓	0.631	0.799	0.542	0.706	0.621	0.792
A	✓	✓	✓	SW	✓	0.635	**0.811**	**0.545**	**0.716**	**0.627**	**0.803**
S	✓					0.510	0.661	0.439	0.574	0.512	0.656
S	✓				✓	0.545	0.709	0.465	0.608	0.532	0.687
S		✓				0.584	0.757	0.507	0.666	0.570	0.742
S		✓			✓	0.592	0.772	0.518	0.686	0.581	0.758
S			✓			0.600	0.791	0.519	0.698	0.586	0.766
S			✓		✓	0.606	0.801	0.535	**0.722**	0.588	0.777
S	✓	✓	✓	HW		0.614	0.794	0.532	0.692	0.602	0.776
S	✓	✓	✓	SM		0.612	0.787	0.539	0.697	0.607	0.777
S	✓	✓	✓	SW		0.615	0.808	0.534	0.705	0.600	0.780
S	✓	✓	✓	HW	✓	**0.627**	**0.823**	**0.542**	0.716	**0.606**	**0.787**
S	✓	✓	✓	SM	✓	0.591	0.761	0.501	0.649	0.575	0.736
S	✓	✓	✓	SW	✓	0.603	0.780	0.518	0.673	0.590	0.759

Table 2 Experiments on combining the response maps of the two CNN models. A_{c5} is only taking features from the last convolutional layer of AlexNet network, S_{c5} is only taking features from the last convolutional layer of SiamFC network. A_{com} is the combined response maps from AlexNet network by soft weight combining, S_{com} is the combined response maps from SiamFC network by hard weight combining. **Boldface** indicates best results

A	S	Fusion	SE	OTB-2013		OTB-50		OTB-100	
				AUC	Prec.	AUC	Prec.	AUC	Prec.
A_{c5}				0.582	0.763	0.496	0.665	0.557	0.735
A_{com}				0.630	0.795	0.538	0.699	0.616	0.786
A_{com}			✓	0.635	0.811	0.545	0.716	0.627	0.803
	S_{c5}			0.600	0.791	0.519	0.698	0.586	0.766
	S_{com}			0.614	0.794	0.532	0.692	0.602	0.776
	S_{com}		✓	0.627	0.823	0.542	0.716	0.606	0.787
A_{com}	S_{com}	HW		0.637	0.815	0.555	0.720	0.625	0.801
A_{com}	S_{com}	SM		0.647	0.819	0.560	0.728	0.638	0.816
A_{com}	S_{com}	SW		0.647	0.818	0.564	0.734	0.637	0.813
A_{com}	S_{com}	HW	✓	0.667	0.852	**0.583**	0.761	0.644	0.824
A_{com}	S_{com}	SM	✓	0.640	0.810	0.557	0.718	0.632	0.804
A_{com}	S_{com}	SW	✓	**0.667**	**0.854**	0.581	**0.764**	**0.647**	**0.831**

as illustrated in Table 1 and Table 2. In addition, we observe that the soft weight strategy is generally the most appropriate, except for combining hierarchical features from the SiamFC model.

4.4 Comparisons

We compare our tracker MFST with MBST [18], LMCF [25], CFNet [23], SiamFC [2], Staple [1], Struck [5], MUSTER [9], LCT [21], MEEM [30] on OTB benchmarks [26,27]. The precision plot and success plot are shown in Fig. 3. Both plots
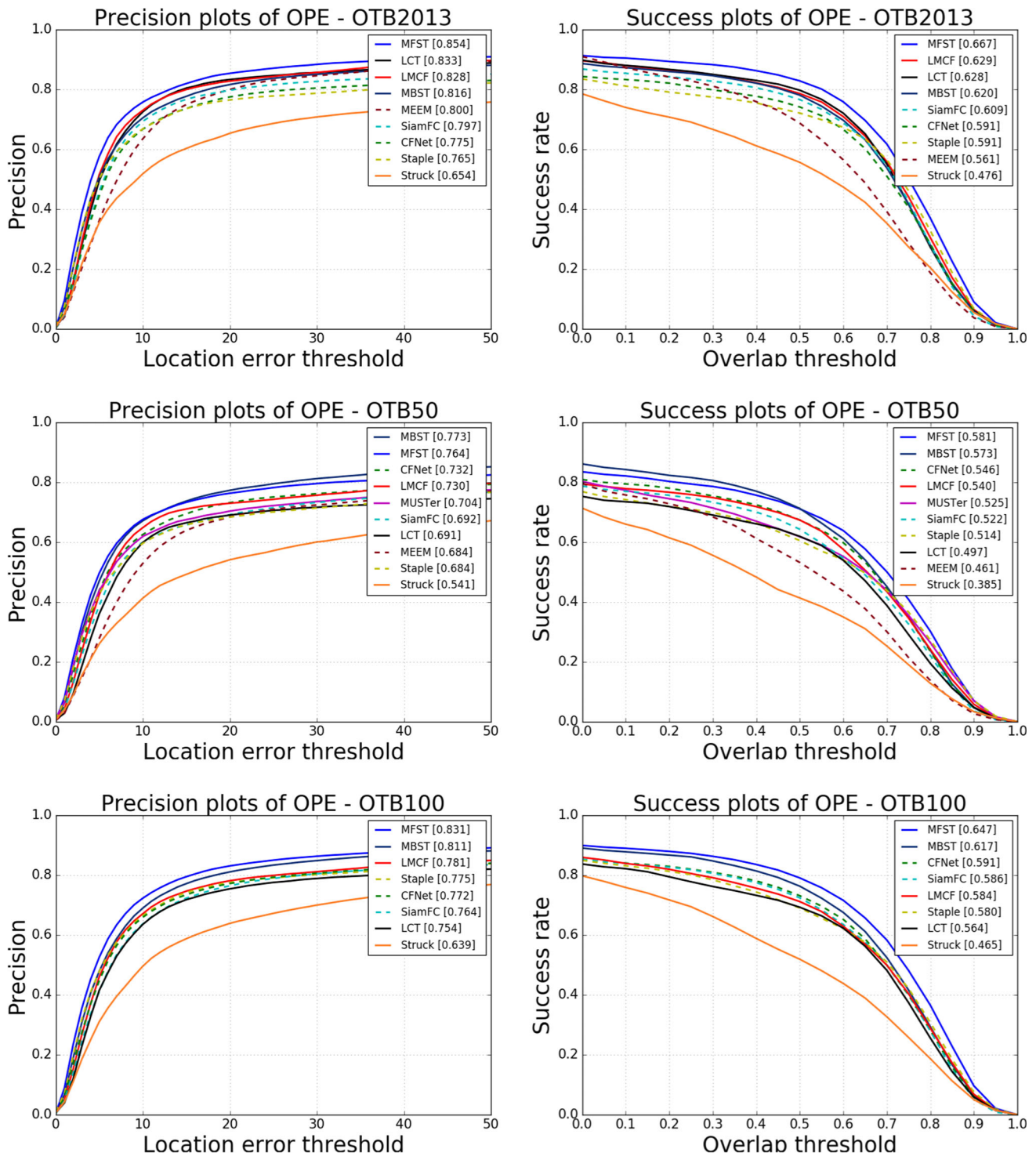
Fig. 3 The evaluation results on OTB benchmarks. The plots are generated by the Python implemented OTB toolkit

show that our tracker MFST achieves the best performance among these recent state-of-the-art trackers on OTB benchmarks, except on the OTB-50 benchmark precision plot. It demonstrates that by using the combined features, the target representation of our method is more robust then our

base tracker SiamFC. The feature calibration mechanism we employed is beneficial for tracking as well. Although we use siamese networks to address the tracking problem as for SiamFC, and take SiamFC as one of our feature extractor, our tracker achieves much improved performance over SiamFC.

Table 3 The speed evaluation results on OTB benchmarks

Tracker	MFST	MBST	LCT	LMCF	MEEM	CFNet	SiamFC	Staple	Struck	MUSTer
Speed(fps)	39	17	27	66	22	75	86	41	10	5

Table 4 Evaluation results of our trackers and some recent the state-of-the-art trackers on VOT2018 benchmark. Bold: best, Italic: second best, Bold Italic: third best. ↑: higher is better, ↓: lower is better. A: Accuracy, R: Robustness, AO: Average overlap, EAO: Expected AO. For the unsupervised experiment, the tracker is not re-initialized when it fails. For the real-time experiment, frames are skipped if the tracker is not fast enough

Tracker	Baseline			Real-time			Unsupervised	
	EAO↑	A↑	R↓	EAO↑	A↑	R↓	AO↑	Speed (FPS)↑
MEEM	0.192	0.463	*0.534*	0.072	0.407	1.592	0.328	4.9
KCF	0.094	0.417	1.726	0.088	0.428	1.926	0.174	N/A
Staple	0.169	*0.530*	0.688	0.170	*0.530*	0.688	0.335	54.3
ANT	0.168	0.464	0.632	0.059	0.403	1.737	0.279	4.1
CFNet	*0.188*	0.503	0.585	***0.182***	0.502	***0.604***	***0.345***	42.6
SiamFCOSP	0.171	***0.508***	1.194	0.166	***0.503***	1.254	0.241	N/A
ALTO	0.182	0.358	0.818	0.172	0.365	0.888	0.252	42.9
SiamRPN++	**0.285**	**0.599**	*0.482*	**0.285**	**0.599**	*0.482*	**0.482**	36.3
MFST	*0.200*	0.497	**0.428**	*0.200*	0.488	**0.455**	*0.348*	33.2

Besides, despite the fact that MBST tracker employs diverse feature representations from many CNN models, our tracker achieves better results with only two CNN models, in terms of both tracking accuracy and speed.

A speed comparison is shown in Table 3 and Table 4 (Speed). Because we use two feature extractor networks, our MFST is slower then SiamFC. Still, it is faster than several trackers in the literature that are less robust. Our method shows a better speed vs accuracy compromise than MBST that combines features from several base feature extractor networks.

In addition to the OTB benchmarks [26,27], we evaluate our MFST tracker on the VOT2018 benchmark [11,13] and compared it with some recent and classic state-of-the-art trackers, including MEEM [30], some correlation-based trackers: KCF [8], Staple [1], ANT [33], and several Siamese-based trackers: CFNet [23], SiamFCOSP [12], ALTO [12] and SiamRPN++ [15]. The results are produced by the VOT toolkit [13] and reported in Table 4. These results show that our method is more robust to the compared trackers with less failures as depicted by the R value. On that aspect, our tracker does better than SiamRPN++, demonstrating that our feature and fusion approach helps in better representing the target. However, it seems that using proposal, like in SiamRPN++ can lead to better accuracy (higher A value). Proposals could be included in our method. Our method ranks a little better for EAO compared to A for the baseline and real-time scenarios. This shows that our features generalize better then the one used by other trackers. Moreover, it is interesting to note that our tracker also performs well in the unsupervised scenario where the tracker is not reset after failure, showing the robustness of our siamese tracker compared to CFNet, SiamFCOSP and ALTO, which, like our tracker, do not use

a region proposal network. Although our tracker is not the fastest siamese tracker, it is fast enough to maintain good performance in the real-time scenario, where frames are skipped if the tracker is not fast enough to process a video at 20 FPS.

5 Conclusion

In this paper, we presented a Multiple Features-Siamese Tracker (MFST) that exploits diverse features at different convolutional layers within the Siamese tracking framework. We utilize features from different hierarchical levels and from different models using three combination strategies. Based on the feature combination, different levels of abstraction of the target are encoded into a fused feature representation. Moreover, the tracker greatly benefits from the new feature representation due to a calibration mechanism applied to different channels to recalibrate features. As a result, MFST achieved strong performance with respect to recent state-of-the-art trackers on object tracking benchmarks.

References

1. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1401–1409 (2016)
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: Hua, G., Jégou, H. (eds.) Computer Vision - ECCV 2016 Workshops, pp. 850–865. Springer, Cham (2016)

3. Choi, J., Chang, H.J., Fischer, T., Yun, S., Lee, K., Jeong, J., Demiris, Y., Choi, J.Y.: Context-aware deep feature compression for high-speed visual tracking. In: IEEE conference on computer vision and pattern recognition, pp. 479–488 (2018)

4. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255 (2009)

5. Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M., Hicks, S.L., Torr, P.H.S.: Struck: structured output tracking with kernels. IEEE Trans Pattern Anal Mach Intell **38**(10), 2096–2109 (2016)

6. He, A., Luo, C., Tian, X., Zeng, W.: A twofold siamese network for real-time object tracking. In: CVPR, pp. 4834–4843 (2018)

7. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision - ECCV 2016, pp. 749–765. Springer, Cham (2016)

8. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans Pattern Anal Mach Intell **37**(3), 583–596 (2015). https://doi.org/10.1109/TPAMI.2014.2345390

9. Hong, Z., Chen, Z., Wang, C., Mei, X., Prokhorov, D., Tao, D.: Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp. 749–758 (2015)

10. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR, pp. 7132–7141. IEEE computer society (2018)

11. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Zajc, L.Č., Vojír, T., Bhat, G., Lukežič, A., Eldesokey, A., Fernández, G., García-Martín, Á., Iglesias-Arias, Á., Alatan, A.A., González-García, A., Petrosino, A., Memarmoghadam, A., Vedaldi, A., Muhič, A., He, A., Smeulders, A., Perera, A.G., Li, B., Chen, B., Kim, C., Xu, C., Xiong, C., Tian, C., Luo, C., Sun, C., Hao, C., Kim, D., Mishra, D., Chen, D., Wang, D., Wee, D., Gavves, E., Gundogdu, E., Velasco-Salido, E., Khan, F.S., Yang, F., Zhao, F., Li, F., Battistone, F., De Ath, G., Subrahmanyam, G.R.K.S., Bastos, G., Ling, H., Galoogahi, H.K., Lee, H., Li, H., Zhao, H., Fan, H., Zhang, H., Possegger, H., Li, H., Lu, H., Zhi, H., Li, H., Lee, H., Chang, H.J., Drummond, I., Valmadre, J., Martin, J.S., Chahl, J., Choi, J.Y., Li, J., Wang, J., Qi, J., Sung, J., Johnander, J., Henriques, J., Choi, J., van de Weijer, J., Herranz, J.R., Martínez, J.M., Kittler, J., Zhuang, J., Gao, J., Grm, K., Zhang, L., Wang, L., Yang, L., Rout, L., Si, L., Bertinetto, L., Chu, L., Che, M., Maresca, M.E., Danelljan, M., Yang, M.H., Abdelpakey, M., Shehata, M., Kang, M., Lee, N., Wang, N., Miksik, O., Moallem, P., Vicente-Moñivar, P., Senna, P., Li, P., Torr, P., Raju, P.M., Ruihe, Q., Wang, Q., Zhou, Q., Guo, Q., Martín-Nieto, R., Gorthi, R.K., Tao, R., Bowden, R., Everson, R., Wang, R., Yun, S., Choi, S., Vivas, S., Bai, S., Huang, S., Wu, S., Hadfield, S., Wang, S., Golodetz, S., Ming, T., Xu, T., Zhang, T., Fischer, T., Santopietro, V., Štruc, V., Wei, W., Zuo, W., Feng, W., Wu, W., Zou, W., Hu, W., Zhou, W., Zeng, W., Zhang, X., Wu, X., Wu, X.J., Tian, X., Li, Y., Lu, Y., Law, Y.W., Wu, Y., Demiris, Y., Yang, Y., Jiao, Y., Li, Y., Zhang, Y., Sun, Y., Zhang, Z., Zhu, Z., Feng, Z.H., Wang, Z., He, Z.: The sixth visual object tracking vot2018 challenge results. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision - ECCV 2018 Workshops, pp. 3–53. Springer, Cham (2019)

12. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Pflugfelder, R., Kamarainen, J.K., Čehovin Zajc, L., Drbohlav, O., Lukezic, A., Berg, A., Eldesokey, A., Kapyla, J., Fernandez, G.: The seventh visual object tracking vot2019 challenge results (2019)

13. Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. IEEE Trans Pattern Anal Mach Intell **38**(11), 2137–2155 (2016). https://doi.org/10.1109/TPAMI.2016.2516982

14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)

15. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR) pp. 4277–4286 (2019)

16. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: The IEEE conference on computer vision and pattern recognition (CVPR), pp. 8971–8980 (2018)

17. Li, P., Wang, D., Wang, L., Lu, H.: Deep visual tracking: review and experimental comparison. Pattern Recognit **76**, 323–338 (2018)

18. Li, Z., Bilodeau, G., Bouachir, W.: Multi-branch siamese networks with online selection for object tracking. In: ISVC, Lecture notes in computer science, vol. 11241, pp. 309–319. Springer (2018)

19. Ma, C., Huang, J., Yang, X., Yang, M.: Hierarchical convolutional features for visual tracking. In: 2015 IEEE international conference on computer vision (ICCV), pp. 3074–3082 (2015)

20. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Robust visual tracking via hierarchical convolutional features. IEEE transactions on pattern analysis and machine intelligence (2018)

21. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. 2015 IEEE conference on computer vision and pattern recognition (CVPR) pp. 5388–5396 (2015)

22. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: The IEEE conference on computer vision and pattern recognition (CVPR), pp. 4293–4302 (2016)

23. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 5000–5008 (2017)

24. Wang, G., Luo, C., Sun, X., Xiong, Z., Zeng, W.: Tracking by instance detection: A meta-learning approach. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6288–6297 (2020)

25. Wang, M., Liu, Y., Huang, Z.: Large margin object tracking with circulant feature maps. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp. 4800–4808 (2017)

26. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Trans Pattern Anal Mach Intell **37**(9), 1834–1848 (2015)

27. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: IEEE conference on computer vision and pattern recognition (CVPR), pp. 2411–2418 (2013)

28. Yan, B., Zhao, H., Wang, D., Lu, H., Yang, X.: 'skimming-perusal' tracking: A framework for real-time and robust long-term tracking. In: IEEE international conference on computer vision (ICCV) (2019)

29. Zgaren, A., Bouachir, W., Ksantini, R.: Coarse-to-fine object tracking using deep features and correlation filters. In: the international symposium on visual computing (ISVC). Springer (2020)

30. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: Proc. of the European conference on computer vision (ECCV), pp. 188–203 (2014)

31. Zheng, Y., Song, H., Zhang, K., Fan, J., Liu, X.: Dynamically spatiotemporal regularized correlation tracking. IEEE Trans Neural Netw Learn Syst **31**(7), 2336–2347 (2020). https://doi.org/10.1109/TNNLS.2019.2929407

32. Zhu, Z., Wang, Q., Bo, L., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: European conference on computer vision, pp. 103–119 (2018)

33. čehovin, L., Leonardis, A., Kristan, M.: Robust visual tracking using template anchors. In: 2016 IEEE Winter conference on appli-

cations of computer vision (WACV), pp. 1–8 (2016). https://doi.org/10.1109/WACV.2016.7477570

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
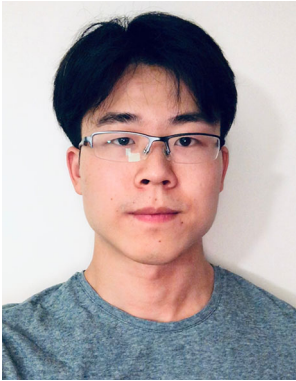
Zhenxi Li received his M.Sc.A. degree in computer engineering from Polytechnique Montréal in 2019 (Québec, Canada) and B.E. degree in software engineering from Xiamen University in 2017 (Xiamen, China). He is currently working at Sensetime as a Computer Vision Researcher, focusing on image processing and machine learning.

Guillaume–Alexandre Bilodeau received the B.Sc.A. degree in computer engineering and the Ph.D. degree in electrical engineering from Université Laval, Canada, in 1997 and 2004, respectively. In 2004, he was appointed Assistant Professor at Polytechnique Montréal, Canada, where he is now full professor since 2014. His research interests encompass image and video processing, video surveillance, object tracking and detection, segmentation, and medical applications of computer vision. Dr. Bilodeau is a member of the REPARTI research network.

Wassim Bouachir received his Ph.D. degree in computer engineering from Polytechnique Montréal in 2015 (Québec, Canada) and the M.Sc. in computer science from the University of Moncton (New Brunswick, Canada). He is currently a professor of computer science at TELUQ University. His research interests include image analysis, video processing, and machine learning. His research works aim to develop intelligent computer vision systems for several application areas, such as security, environment sciences, and health–care applications.