Signal Processing 199 (2022) 108628

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

More robust object tracking via shape and motion cue integration

Bo Gao*, Michael W. Spratling

The Department of Informatics, King's College London, Strand, London WC2R 2LS, UK

ARTICLE INFO

Article history:
Received 31 August 2021
Revised 5 May 2022
Accepted 21 May 2022
Available online 22 May 2022

Keywords:
Tracking-by-detection trackers
Distractor-suppression
Neural style transfer
Kalman filter

ABSTRACT

Most current trackers utilise an appearance model to localise the target object in each frame. However, such approaches often fail when there are similar looking distractor objects in the surrounding background. This paper promotes an approach that can be combined with many existing trackers to tackle this issue and improve tracking robustness. The proposed approach makes use of two additional cues to target location: shape cues which are exploited through offline training of the appearance model, and motion cues which are exploited online to predict the target object's future position based on its history of past locations. Combining these additional mechanisms with the existing trackers SiamFC, SiamFC++, Super_DiMP and ARSuper_DiMP all resulted in an increase in the tracking accuracy compared to that achieved by the corresponding underlying tracker alone. When combined with ARSuper_DiMP the resulting tracker is shown to outperform all popular state-of-the-art trackers on three benchmark datasets (OTB-100, NFS, and LaSOT), and produce performance that is competitive with the state-of-the-art on the UAV123, Trackingnet, GOT-10K and VOT2020 datasets.

1. Introduction

Visual tracking is a fundamental task in computer vision with numerous applications in surveillance [1,2], self-driving vehicles [3,4], and UAV-based monitoring [5,6]. It is the task of locating the same moving object in each frame of a video sequence, given only the initial appearance of the target object. Traditional trackers [7–10] often employed a motion prior, obtained by a Kalman filter or particle filter, as an additional cue to infer the location of the target. However, modern tracking datasets contain many sequences with random camera motion, which causes most motion models to fail [11]. This has resulted in most modern trackers simply searching for the target around the location at which it appeared in the previous frame, and otherwise relying entirely on appearance modelling. The availability of discriminative appearance descriptors, such as Histogram of Oriented Gradients [12], color names [13] and deep features, mean that most modern trackers treat visual tracking as a classification problem. By learning an appearance model of the target from the initial frame, the trackers distinguish the target from background by a cross-correlation operation and predict its location in the following frames. Although they ignore additional information such as motion, these Tracking-by-Detection methods achieve impressive performance. However, they can fail when the appearance model misidentifies a similar-looking object (a "distractor") as the target. Even the recent state-of-the-art tracker, Super_DiMP[1], which fully exploits (through end-to-end offline training and online meta-learning) both target and background appearance information, is still fooled by distractors as shown in Fig. 1(b).

It is well-known that deep models are data-hungry. Training trackers with bigger datasets could potentially improve performance as the new trained model could extract more discriminative features of the target and distractors, allowing the target to be more likely matched to the correct location. One method to increase the amount, and diversity, of available training data for deep models is data augmentation. A recent study [16] demonstrated that ImageNet-trained CNNs are biased toward making categorisation decisions based on texture cues rather than shape cues. This work also showed that CNNs could be trained to increase sensitivity to shape cues using a data augmentation technique based on neural style transfer and that this would improve accuracy and robustness both of object classification and detec-

* Corresponding author.
E-mail addresses: bo.gao@kcl.ac.uk (B. Gao), michael.spratling@kcl.ac.uk (M.W. Spratling).

[1] Super_DiMP combines the bounding-box regressor of PrDiMP [14] with the standard DiMP classifier [15]. Code for this tracker is available at https://github.com/visionml/pytracking/

https://doi.org/10.1016/j.sigpro.2022.108628

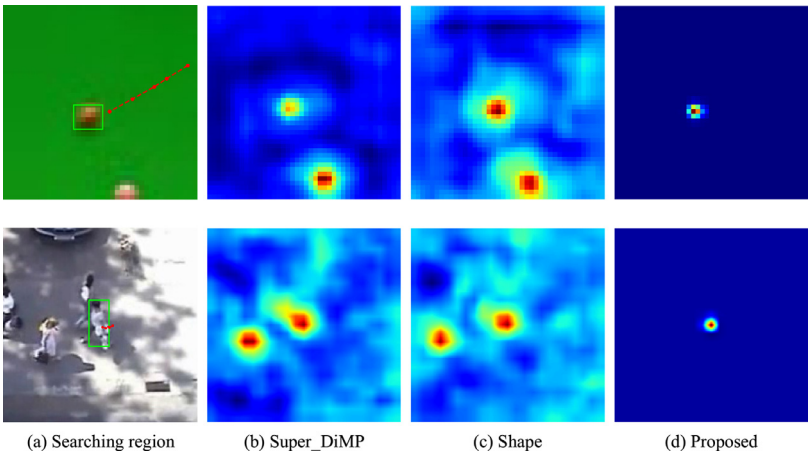


Fig. 1. Comparison of our approach with Super_DiMP on two hard scenarios. (a) shows the search region of the current video frame. The green rectangle is the ground truth location of the tracked target object and the red dashed line is the trajectory of the target in last five frames generated by its previous locations (red points) predicted by Super_DiMP. (b), (c) and (d) are score maps produced by Super_DiMP, by Super_DiMP trained to be more sensitive to shape cues, and by Super_DiMP trained to be more sensitive to shape cues and using the proposed motion method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

tion. A VGG19 [17] model trained with the same data augmentation method was found to extract more discriminative features that improve the performance of template matching methods [18]. Considering the Tracking-by-Detection based trackers use template matching technique to classify the target, these results suggest that the same data augmentation method could potentially also improve the matching performance in these trackers. In this article we verified this assumption by training trackers with different degrees of shape sensitivity. Our results show that training them to learn about texture cues while biasing them to be more sensitive to shape cues, can improve tracking performance.

However, only using the appearance model to identify the tracked object is insufficient to achieve robust results. As shown in Fig. 1(c), increasing shape sensitivity of Super_DiMP can improve the match between the appearance model and the target, but it is still not always possible to distinguish the target from similar looking distractors. In these examples the previous velocities of both targets are stable and their movements are nearly linear (Fig. 1(a)), and hence, such motion information can be used to identify the true location of the target. Inspired by this observation, this paper proposes a sophisticated location predictor for online inference using motion cues to improve the robustness of tracking. A motion module is developed to generate a probabilistic map that the object is present at a certain location in the searching region, based on the state estimated by a Kalman filter. Different from previous approaches, we estimate the movement of the background between two consecutive frames and use this global movement to compensate for the motion of the camera, and improve the robustness of the Kalman filter. The resulting probabilistic map narrows-down the range of locations where the target might appear in the current frame.

Our main contributions are summarised as follows:

- We demonstrate that current state-of-the-art trackers are biased towards texture cues. Furthermore, we show that performance can be improved by allowing trackers to learn about texture cues while biasing them to be more sensitive to shape information.
- We propose a novel motion predictor, that can cope with random camera movements, to provide an estimate of the target location based on its previous locations. We demonstrate that this motion model can make the identification of the target more reliable.
- We demonstrate the effectiveness of our approach by integrating shape and motion cues with the recent state-of-the-art tracker ARsuper_DiMP [19]. The resulting tracker achieves new state-of-the-art performance on three benchmark datasets: OTB-100 [20], NFS [21], and LaSOT [22].
- When integrated with three alternative trackers, Super_DiMP [23], SiamFC [24] and SiamFC++ [25], in all cases the performance of the underlying tracker is improved. This indicates that the proposed method is, potentially, a general approach that could be used to improve the performance of most current visual trackers.

2. Related work

The Tracking-by-Detection architecture is widely used in recent trackers [14,15,19,24–43]. These algorithms exploit the representations produced by deep learning using end-to-end offline training, and can be sub-divided into generative trackers [24–35] and discriminative trackers [14,15,19,36–43]. The former formulate object tracking as a cross correlation problem in deep feature space and learn a similarity metric offline by training a Y-shaped network containing two branches, one for the object template and the other for the search region. This approach is exemplified by Siamese network based trackers which have gained significant attention due to their promising performance and efficiency. However, they typically employ a fixed target template and do not model background information, which consequently results in incorrect tracking when there is a similar looking object in the background or a significant change of the target appearance. Despite appearance updating strategies [26,30,33,35] that have been recently introduced into Siamese network based trackers, their performance is still below that of discriminative trackers.

The discriminative trackers are exemplified by learning a discriminative classifier online to distinguish the target from the background. These trackers [14,15,19,38,40,41] employ a meta-learning formulation to predict the weights of the classification layer. This enables discriminative trackers to achieve state-of-the-art results on many benchmarks. Despite discriminative trackers learning an appearance model using background information, the appearance model is still unable to deal with cases that contain highly similar looking distractors (as shown in Fig. 1).

Tracking failures caused by a similar-looking location being misidentified as the target, indicates that only using the appear-

ance model to identify the tracked object is insufficient to achieve robust results. Some existing methods attempt to address this issue by taking motion cues into consideration. For example, Gladh et al. [39] use deep motion features extracted from optical flow images together with appearance features to generate the target model. Wang et al. [44] predict the approximate location of the target by decoupling camera motion and object motion to create an adaptive search region.

Other methods attempt to address this issue by introducing attention mechanisms. For example, RAR [45] employed a hierarchical attention module to leverage both inter- and intra-frame attention at each convolutional layer which effectively highlighted informative representations and suppressed distractors. SiamGAT [46] employed a graph attention module to replace the cross-correlation operation, that is common in Siamese trackers, for part-to-part matching which effectively passed target information from the template to the search region. TrDiMP [41] proposed an appearance model generator using a transformer [47], the transformer-encoder promotes the previous appearance models via attention based feature reinforcement to acquire more compact target representations while the transformer-decoder generates the appearance model for the current frame. TransT [48] developed a Transformer-like fusion module to combine the template and search region features solely using attention instead of correlation. STMTrack [49] created a space-time memory network inspired by non-local self-attention [50] to fully use historical information about the target to better adapt to appearance variations during tracking [51]. presented a new tracking architecture with an encoder-decoder transformer. The encoder models the global spatio-temporal feature dependencies between target objects and search regions, while the decoder learns a query embedding to predict the spatial positions of the target objects.

Some approaches take into account the information about possible distractors. For example, DaSiamRPN [28] proposed a distractor-aware feature learning scheme to boost the discriminative power of the networks during off-line training, and also a novel distractor-aware module to suppress distractors during on-line tracking. KYS [38] presented an end-to-end learning architecture, where the encoding of image regions is learned and propagated by appearance-based dense tracking between frames. The final prediction is then obtained by combining the explicit background representation with the appearance model output. Nocal-Siam [52] proposed a target-aware non-local attention module to jointly refine visual features of the target and search branches which suppressed distractors [53]. developed a novel tracking architecture that detects distractors in every frame. These are represented as additional appearance models with the same size as the target appearance model. The predicted location of the target in the next frame takes into consideration not only the tracked object, but also the distractors using a inference process known as explaining away. As a consequence, matches between the target appearance model and the surrounding background are suppressed, and the identification of the target is more reliable.

Other distractor-suppression techniques had been proposed for specific tracking architectures, but would be difficult to incorporate in modern Tracking-by-Detection approaches. For example, [54] developed an on-line feature ranking mechanism to select the top-ranked appearance features for the trackers based on color histogram. TLD [55] proposed the Tracking-Learning-Detection architecture which implemented a P-N learning mechanism to exploit spatio-temporal relationships in the video. Siam-RCNN [56] proposed a Siamese re-detection architecture with a novel Tracklet Dynamic Programming Algorithm to simultaneously track all potential objects and select the best object in the current timestep based on the complete history of all target and distractor object tracklets[57]. proposed a novel hard negative mining method to suppress distractors for long-term tracking which enhanced the target identification ability of a verification network.

Table 1
The different training strategies used to produce four Super_DiMP models with different degrees of shape bias. DS and SDS are abbreviations for the original training dataset and the Stylised-training dataset respectively. Ranking describes the relative shape sensitivity.

Name	Training	Fine-tuning	Ranking
Model_A	DS	-	4
Model_B	SDS	-	1
Model_C	DS+SDS	-	2
Model_D	DS+SDS	DS	3

In contrast, we present distractor-suppression methods that are applicable to all modern Tracking-by-Detection based trackers. A recent study showed that ImageNet-trained CNNs are strongly biased towards recognising textures cue rather than shapes cues [16]. This study also demonstrated that the same standard architecture (ResNet-50 [58]) that learns a texture-based representation on ImageNet is able to learn a shape-based representation when trained on Stylised-ImageNet: a version of ImageNet that replaces the texture in the original image with the style of a randomly selected painting through AdaIN style transfer [59]. This new shape-sensitive model was found to be more accurate and robust in both object classification and detection tasks. A VGG-19 [17] model trained with same data augmentation method was found to extract more discriminative features that improve the performance of template matching methods [18]. Yingwei et al [60] proposed a new training method that provides the corresponding supervisions from shape and texture simultaneously which improved model performance on several image recognition benchmarks and enhanced adversarial robustness. Inspired by these findings, in this paper we trained four trackers with different degrees of shape sensitivity. The new shape sensitive models were found to have better tracking performance.

We further use motion cues, ignored in current approaches, with our tracker. We estimate the movement of the background between two frames and a Kalman filter takes the global movement as input every frame to compensate for the motion of the camera, and thus improves its robustness. The state estimated by the filter is used to generate a probabilistic map of the predicted target location in the search region. This is combined with the score map produced by the tracker's appearance model, to improve the estimate of the targets position in the current frame.

3. Proposed method

We first trained four Super_DiMP models with different shape sensitivity to determine the best balance between texture and shape bias, as described in Section 3.1. The resulting, shape-biased, version of Super_DiMP was then modified to incorporate the proposed motion module, as described in Section 3.3.

3.1. Training super_DiMP with stylised-dataset

Super_DiMP was trained with the training splits of LaSOT [22], GOT10k [61], Trackingnet [62] and MSCOCO [63]. The corresponding styled training datasets were generated by applying AdaIN style transfer [16]. As shown in Fig. 2, after applying AdaIN style transfer local texture cues tend to be modified while the global shape tends to be retained. We followed the procedure used in [16] to define four training regimes, see Table 1, that would result in different degrees of shape selectivity.

Model_A was trained using the standard training dataset (we used the pretrained Super_DiMP model from pytracking [23] model zoo) which has the least shape bias. Model_B was

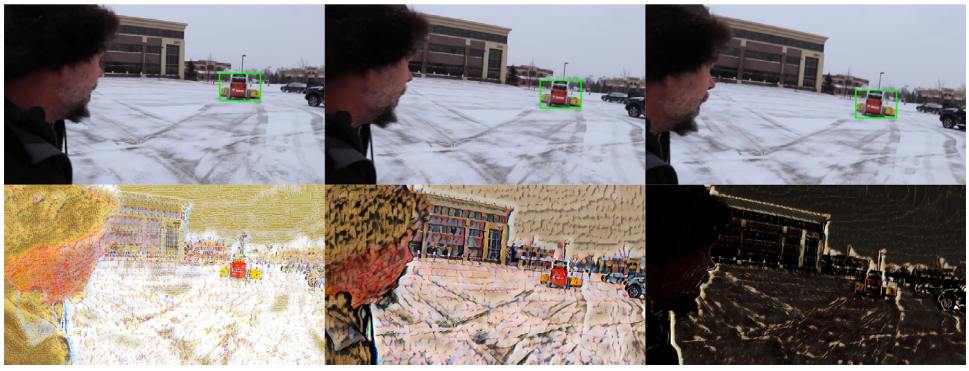


Fig. 2. Visualisation of Stylised-training dataset. Top row: three randomly selected consecutive frames. Bottom row: corresponding images in the stylised-training dataset, with textures selected randomly from ten different painting styles. The tracking object is within the green bounding box. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

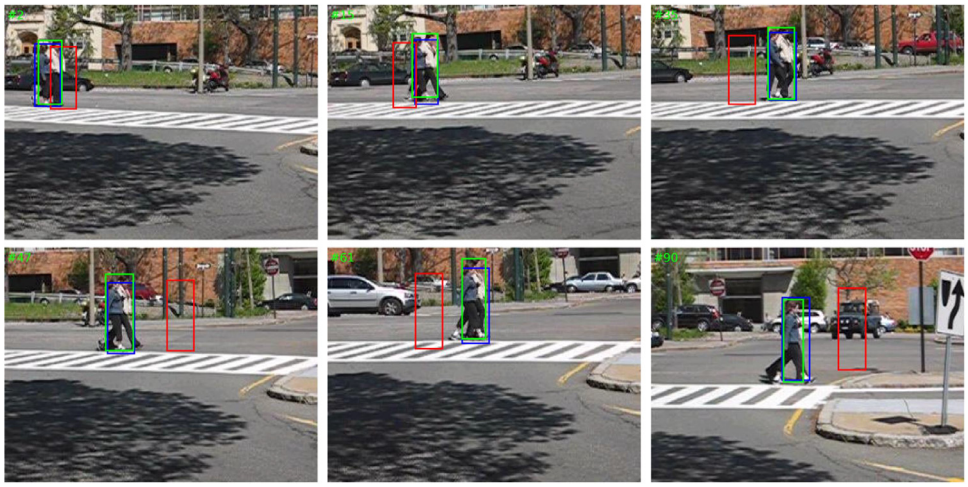


Fig. 3. An example of a sequence recorded using a unsteady hand-held camera. The green box is the ground truth bounding box of the target. The blue box is predicted by a Kalman filter with the proposed global movement estimation method, while the red box is predicted by the same Kalman filter but without the proposed global movement estimation method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

trained on the Stylizsed dataset and thus has the most shape bias. Model_C was trained on a dataset containing the images from both the standard and Stylised dataset. Model_D was initialised with the weights of Model_C and then fine-tuning on the standard dataset for 34 epochs using a learning rate of 4×10^{-5} multiplied by 0.2 after 7 epochs. Therefore Model_C and Model_D have intermediate levels of shape bias, with model_D being less selective to shape than Model_C. The learning rate was 2×10^{-4} multiplied by 0.2 after every 15 epochs for Model_B and after every 7 epochs for Model_C. The number of training epochs was 50 for Model_B and 25 for Model_C (as the dataset used to train Model_C was twice as large as that used to train Model_B the number of weight updates was the same for both models). The official settings were used for all other training hyper-parameters for each model [23].

3.2. Motion prediction

Background movements are inevitable in some tracking sequences, especially for videos recorded using hand-held or vehicle-mounted cameras, where camera shake can result in large and unpredictable global movements between frames. In such circumstances most motion models fail to produce an accurate estimate of the object location, as illustrated for a Kalman filter in Fig. 3. To solve this problem, the proposed motion module estimates, and discounts, the global motion caused by camera movement first. This is achieved by matching keypoints across successive frames to estimate the global image motion caused by the camera. A pair of matched keypoints between frame t and $t-1$ is the one which the similarity of their descriptors is more than 0.8. The global movement between the two frames is estimated as:

$$(x_{t|t-1}, y_{t|t-1}) = \frac{\sum_{i=1}^{n} (\mathbf{p}_{t,i} - \mathbf{p}_{t-1,i})}{n} \quad (1)$$

Where $x_{t|t-1}$ and $y_{t|t-1}$ are component movements in the $x-$ and $y-$ directions, i is the index over the number of matched pairs n, $\mathbf{p}_{t,i}$ represents the coordinates of a keypoint in frame t and its matched location in frame $t-1$ is $\mathbf{p}_{t-1,i}$. Keypoints and descriptors are obtained using self-evolving keypoint detection and description (SEKD) which is the current state-of-the-art method [64]. This method is based on a deep neural network and runs much faster than hand-crafted methods such SIFT [65], SURF [66] and ORB [67]. 500 keypoints were detected in every frame. Note, that all matched keypoints were used to estimate global motion, as it was found that the additional computational burden imposed by finding those matches that were consistent with the same transformation, for example using the RANSAC algorithm, was significant while the resulting improvement in accuracy was insignificant.

The state of the target is formulated as a 4-dimensional vector $\mathbf{s} = (x, y, v_x, v_y)$. Where (x, y) represents the centre of the target, and v_x and v_y are component velocities in the $x-$ and $y-$ directions. The center is initialised to the ground-truth values of the first frame, while the velocities are initialised to zero. We use a Kalman filter with the Constant Velocity Model [68] to predict the

object state. A Kalman filter was preferred to a particle filter because of the computational complexity, and issues with degeneracy and divergence, of a particle filter. The estimated function of the Kalman filter is shown in Eqs. (2) and (3):

$$\bar{\mathbf{s}}_t = \mathbf{F}\mathbf{s}_{t-1} \tag{2}$$

$$\bar{\mathbf{s}}_t(x_t, y_t) = \bar{\mathbf{s}}_t(x_t + x_{t|t-1}, y_t + y_{t|t-1}) \tag{3}$$

where $\bar{\mathbf{s}}_t$ is the estimated state given observations up to time $t-1$, $\mathbf{s}_{t-1}$ is the optimal result of its previous state, and $\mathbf{F}$ is the transition matrix. $x_{t|t-1}$ and $y_{t|t-1}$ are the component movements in the $x-$ and $y-$ directions calculated by Eq. (1), and are used in Eq. (3) to remove the influence of background movement.

The covariance corresponding to the process result is predicted using:

$$\bar{\mathbf{P}}_t = \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}^{\mathbf{T}} + \mathbf{Q} \tag{4}$$

where $\bar{\mathbf{P}}_t$ is a prediction of the covariance corresponding to the state $\mathbf{s}_t$ and $\mathbf{P}_{t-1}$ is the covariance for the previous optimised state $\mathbf{s}_{t-1}$. $\mathbf{Q}$ is the covariance matrix of system noise which is assumed to be the Discrete Constant White Noise Model with variance 0.04^2.

The estimated state $\bar{\mathbf{s}}_t$ is then optimised using the corresponding observation, i.e. the result estimated by the estimater of the tracker for frame t:

$$\mathbf{s}_t = \bar{\mathbf{s}}_t + \mathbf{K}_t(\mathbf{O}_t - \mathbf{H}\bar{\mathbf{s}}_t) \tag{5}$$

where $\mathbf{O_t}$ is the observation obtained from the underlying tracker i.e the object location $(\dot{x}_t, \dot{y}_t)$ made by using the appearance model of tracked target. $\mathbf{H}$ is the observation matrix. $\mathbf{K}_t$ is the Kalman gain at frame t defined by:

$$\mathbf{K}_t = \frac{\bar{\mathbf{P}}_t\mathbf{H}^T}{\mathbf{H}\bar{\mathbf{P}}_t\mathbf{H}^T + \mathbf{R}} \tag{6}$$

where $\mathbf{R}$ is the covariance matrix of the measurement noise (assumed to be Gaussian with variance 0.35^2).

The covariance is updated as follows:

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\bar{\mathbf{P}}_t \tag{7}$$

where $\mathbf{I}$ is an identity matrix.

A Kalman filter can only handle linear and Gaussian systems, however the behaviour of most tracking targets are nonlinear and unknown. Therefore, in some sequences where the velocity of the object changes significantly, the Kalman filter is unable to predict the target location for a period. To solve this problem, Wang et al [44] proposed a method that separates a long video into several short pieces. Following the same approach, we reset the filter every n_s (a value of 20 was used) frames. At the end of each twentieth frame, the target's state is set to $(\dot{x}_t, \dot{y}_t, v_{xt}, v_{yt})$, where (v_{xt}, v_{yt}) are the velocities in $\mathbf{s}_t$. The whole process of the motion module is summarised by Algorithm 1.

3.3. Predictor with motion guide

The motion module is integrated with a tracker as illustrated in Fig. 4. During online inference, both generative and discriminative trackers (see Section 2) predict a scalar confidence score $\mathbf{S}(\mathbf{X}) \in \mathbb{R}$ given a search region of an image $\mathbf{X}$. The score map is defined as:

$$\mathbf{S}(\mathbf{X}) = \mathbf{A} * \phi(\mathbf{X}) \tag{8}$$

Here, $\phi(\mathbf{X})$ are the features extracted from the search region of the image, commonly by a CNN. $\mathbf{A}$ is the target appearance model. For Siamese trackers it represents the features of the template $\mathbf{Z}$, i.e. $\mathbf{A} = \phi(\mathbf{Z})$. For DCF trackers it is the convolution kernel which is trained online. $*$ represents the cross-correlation operation.

Algorithm 1 Motion module loop.

Require: Current frame im_t and target center location estimated by the tracker $(\dot{x}_t, \dot{y}_t)$;
Ensure: The estimated state $\bar{\mathbf{s}}_t$;
1: Detect keypoints in first frame;
2: Initialise Kalman filter with the ground truth center location;
3: **for** $t = 2$ to n **do**
4: Detect keypoints in im_t;
5: Calculate matched points $\mathbf{p}_t, \mathbf{p}_{t-1}$ between im_t and im_{t-1};
6: Estimate background movement using Eq. 1;
7: Estimate state $\bar{\mathbf{s}}_t$ and update with the background movement using Eq. 2 and 3
8: Estimate $\bar{\mathbf{P}}_t$ using Eq. 4;
9: **if** $t \nmid n_s$ **then**
10: Update the Kalman filter with $(\dot{x}_t, \dot{y}_t)$ using Eq 5,6 and 7;
11: **else**
12: Reset the Kalman filter with initial state $\bar{\mathbf{s}}_t = (\dot{x}_t, \dot{y}_t, v_{xt}, v_{yt})$;
13: **end if**
14: **end for**

The score map measures the similarity between an appearance model and the deep features extracted from the current video frame. The tracker estimates the target object's location by finding the location of the maximum in the score map. If the appearance of the target is distinctive there will only be one peak in the score map. However, if there are similar looking distractors in the search region, the score map will have multiple peaks. To be specific, a peak is defined as a local maximum (within a 3-by-3 neighbourhood) that has a value over a global threshold which is set to g (a value of 0.85 was used) times the max value of the score map.

Based on the state $\bar{\mathbf{s}}_t(x_t, y_t, v_{xt}, v_{yt})$ of frame t estimated by the motion module, its corresponding location $(\hat{x}_t, \hat{y}_t)$ within search region $\mathbf{X}$ is determined. A motion probabilitic map $\mathbf{P}(\mathbf{X})$ with the same size as $\mathbf{X}$ is generated using a two-dimensional circular-symmetric Gaussian function centred at $(\hat{x}_t, \hat{y}_t)$ and with standard deviation σ_g (a value of 0.5 was used). If the score map, $\mathbf{S}(\mathbf{X})$, of frame t has multiple peaks, it is filtered by:

$$\mathbf{S}(\mathbf{X}) = \mathbf{S}(\mathbf{X}) \odot \mathbf{P}(\mathbf{X}) \tag{9}$$

$$\mathbf{S}(\mathbf{X}) = \mathbf{S}(\mathbf{X}) \times \frac{a}{b} \tag{10}$$

Where $\odot$ represents element-wise multiplication, and a and b are the maximum of $\mathbf{S}(\mathbf{X})$ before and after applying Eq. (9) respectively. Super_DiMP checks the maximum of $\mathbf{S}(\mathbf{X})$ every frame, if the value is below a threshold that is interpreted as a tracking failure, the tracker does not update the location of the target, and outputs the same predicted bounding box as the last frame. To prevent the maximum value being effected by the element-wise multiplication used in Eq. (9), Eq. (10) is used to ensure the combined score map has the same maximum value as it did before applying the multiplication.

Occasionally, when the target rapidly changes velocity, the Kalman filter may fail to accurately predict its location. To avoid this causing a reduction in tracking performance, we only combine the appearance score map and the motion probabilistic map using Eqs. (9) and Eq. (10) if the previous motion of the target has been near linear. Specifically, we require the standard deviation of the angles between the velocities and the $x-$ direction, over the last n_p (default value is 5) frames to be within σ_a (a value of 10 was used) degrees.

The complete proposed tracker is summarised by Algorithm 2.

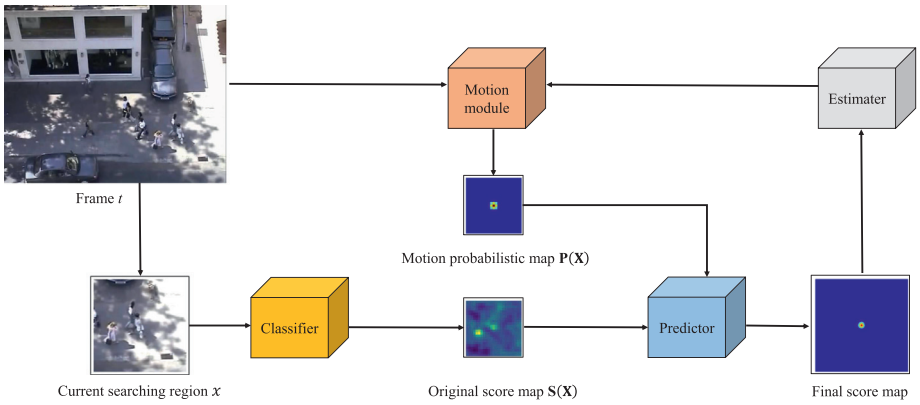


Fig. 4. An overview of the proposed tracking architecture. The motion module detects keypoints every frame and the movement of the background between frame t and $t-1$ is estimated by averaging the distances between the matched keypoints of the two frames. A Kalman filter then updates using the global movement and current state from the estimater. Based on the predicted state, the motion module outputs a probabilistic map. The predictor uses the motion probabilistic map to suppress distractors. Note: the original score map generated by the tracker is the same size as the image features $\phi(\mathbf{X})$, while the final score map is upsampled to the size of the input image.

Algorithm 2 Tracking loop.

Require: The ground-truth bounding box of the target in first frame, current frame im_t;
Ensure: The predicted bounding box of target in every frame;
1: Initialise underlying tracker and Algorithm 1 using the ground-truth bounding box;
2: **for** $t = 2$ to n **do**
3: Estimate current state $\bar{\mathbf{s}}_t$ by executing the step 4 to 8 of Algorithm 1;
4: Generate the score map $\mathbf{S}(\mathbf{X})$ by the classifier in the underlying tracker;
5: Find the number of peaks, m, in $\mathbf{S}(\mathbf{X})$;
6: **if** $m > 1$ **then**
7: Calculate the angles, a, over the last n_p frames;
8: **if** $a.std() < \sigma_a$ **then**
9: Modify $\mathbf{S}(\mathbf{X})$ using Eq. 9 and 10;
10: **end if**
11: **end if**
12: Estimate the predicted bounding box by the estimator in the underlying tracker using $\mathbf{S}(\mathbf{X})$;
13: Execute the step 9 to 13 of Algorithm 1 using the predicted bounding box;
14: **end for**

4. Experiments

To evaluate the proposed methods, we use Super_DiMP [23] and ARSuper_DiMP [19] as baseline trackers. Super_DiMP was initially trained with both standard and styled datasets, to find the degree of shape bias that produced the best performance (see results in Section 4.1). This re-trained version of Super_DiMP was then integrated with the proposed motion module. As ARSuper_DiMP uses an Alpha-Refine module to improve the accuracy of the bounding boxes predicted by Super_DiMP, the re-trained Super_DiMP was also employed in it and the hyper-parameters used by the motion module were tuned using Super_DiMP and re-used for ARSuper_DiMP. The resulting trackers, termed as Super_Cues_DiMP and ARSuper_Cues_DiMP, were evaluated using the following seven tracking benchmarks (see results in Section 4.2):

OTB-100 [20] This is the first modern benchmark for single object tracking and has been used extensively to evaluate visual trackers. It contains 100 videos with more than 59K frames.

UAV123 [69] This is a large dataset captured from low-altitude UAVs. It contains over 110K frames and 123 videos. It is quite challenging due to small tracked objects and fast motion.

NFS [21] This dataset contains 100 videos captured using a high frame rate (240 FPS) camera. We perform our experiments using the 30 FPS version of this dataset in which videos have an average length of 479 frames.

LaSOT Fan et al. [22] The large-scale LaSOT dataset contains 280 videos in its test set. The video sequences, which have an average length of 2500 frames, are longer than those in other datasets, testing not only the accuracy of the tracker but also its robustness.

Trackingnet Muller et al. [62] This dataset provides over 30K videos sampled from YouTube. We report results on its test set, consisting of 511 videos with an average of 441 frames per sequence.

GOT-10K Huang et al. [61] This is a recent large-scale dataset consisting of 10K video sequences. With this dataset trackers are evaluated on 180 videos with 84 object classes and 32 motions that cover a wide range of common moving objects in the wild.

VOT2020 Kristan et al. [70] The VOT challenge, held yearly, provides a precisely defined and repeatable way of comparing short-term trackers. VOT2020 contains 60 videos with binary segmentation masks as the ground-truth.

The intersection-over-union (IoU) is calculated between the predicted bounding box and the ground truth box. The overall success score is determined by calculating the area under the curve (AUC) of a success curve produced by varying the threshold (between 0 to 1) of IoU that counts as success. The precision score is defined as the percentage of frames whose estimated location is within 20 pixels of the ground truth. The AUC and precision score are used for ranking trackers on OTB-100, UAV123, NFS, and Trackingnet. Trackingnet also use normalised precision score (P_{norm}), please refer to Muller et al. [62] for details. AUC and P_{norm} are commonly used for LaSOT. For GOT-10K, the results are evaluated in terms of average overlap (AO) and success rates ($SR_{0.50}$ and $SR_{0.75}$) the percentage of successfully tracked frames where overlap rates are above the given threshold. VOT2020 uses an expected average overlap (EAO) as the main metric to rank trackers while their accuracy and robustness are also considered.

Due to the stochastic nature of DCF trackers, the results reported for DiMP-based trackers [14,15,38] are an average over multiple runs. For OTB-100, NFS, UAV123, and LaSOT, the results were averaged over five runs. As Trackingnet results are obtained using

Table 2
Results of four Super_DiMP models with different training strategies on three benchmarks.

Name	OTB-100 AUC	UAV123 AUC	GOT-10K AO
Model_A	70.1	66.5	67.5
Model_B	70.4	65.3	62.0
Model_C	69.3	65.0	64.1
Model_D	**70.6**	**67.7**	**68.4**

Table 3
Comparison with State-of-the-art trackers on NFS dataset. The best two results are highlighted by red and green.

Arch.	Tracker	Succ.	Prec.
Tracking-By-Detection	ATOM [36]	58.0	70.0
	FCOT [40]	63.2	76.1
	DiMP50 [15]	61.5	74.1
	PrDiMP50 [14]	63.5	75.9
	KYS [38]	63.3	76.1
	SiamBAN [71]	59.5	70.0
	TrDiMP [41]	65.8	79.1
	Super_DiMP [23]	64.7	78.1
	ARSuper_DiMP [19]	66.3	80.5
	Super_Cues_DiMP (ours)	65.6	79.5
	ARSuper_Cues_DiMP (ours)	66.5	81.2
Others	STMTrack [49]	-	-
	TransT [48]	65.7	-
	Siam R-CNN [56]	63.9	-

an online evaluation server, only a single run was used. For GOT-10K three runs were used. For a fair comparison, we follow the same approach to test our methods. Super_Cues_DiMP and ARSuper_Cues_DiMP were evaluated on a single Nvidia Tesla V100 GPU and the running speed are 17 FPS and 12 FPS respectively. Our code is available at: https://github.com/iminfine/trackingmulticues.

4.1. Training with stylised-dataset

Super_DiMP models were trained using different combinations of datasets (as summarised in Table 1) to have different degrees of shape sensitivity. The performance of these trackers on three tracking benchmarks, OTB-100, UAV123 and GOT-10K, is shown in Table 2. Except for Model_B on OTB-100, the results for Model_B and Model_C are worse than those for Model_A, and the results for Model_D are better than for Model_A. This demonstrates that Super_DiMP model relies more heavily on texture cues than shape cues, but that slightly increasing its shape bias can highly improve its robustness.

4.2. state-of-the-art comparison[2]

Fig. 5 a and b show results on the OTB-100 dataset [20]. Despite performance on this dataset becoming saturated during recent years, ARSuper_Cues_DiMP produces state-of-the-art result with an AUC (success score) of 73.0%. Super_Cues_DiMP outperforms the baseline Super_DiMP by 1.4% in terms of AUC and 2% in terms of precision.

Fig. 5 c and d show results on the UAV123 dataset [69]. The proposed tracker Super_Cues_DiMP outperforms the previous best approaches, and achieves the new state-of-the-art results with an AUC of 69.4% and precision of 91.8%.

Results for NFS [21] are shown in Table 3. It can be seen that the performance ARSuper_Cues_DiMP is similar with the underline tracker, ARSuper_DiMP. Super_Cues_DiMP achieves a success score of 65.6% which is 0.9 percentage points higher than the underlying tracker Super_DiMP.

Results for the LaSOT dataset [22] are shown in Table 4. ARSuper_Cues_DiMP achieves the state-of-the-art results with 66.7% and 75.2% in success score and normalised precision score and Super_Cues_DiMP outperforms the underlying tracker Super_DiMP with relative gains of 1.5% and 2.2% in these two metrics.

For the results of Trackingnet dataset [62], as shown in Table 5, ARSuper_Cues_DiMP outperforms the baseline ARSuper_DiMP by 0.4% in terms of AUC and precision respectively which is comparable with the state-of-the-art results.

Results for the GOT-10K dataset [61] are shown in Table 6. ARSuper_Cues_DiMP and Super_Cues_DiMP outperform their baseline

[2] Note that using the raw results provided by the authors, we were unable to reproduce the scores reported for TrDiMP in [41] for the OTB-100, UAV123 and NFS datasets. Our different results are shown in Fig. 5 and Table 3.

Table 4
Comparison with State-of-the-art trackers on LaSOT dataset.

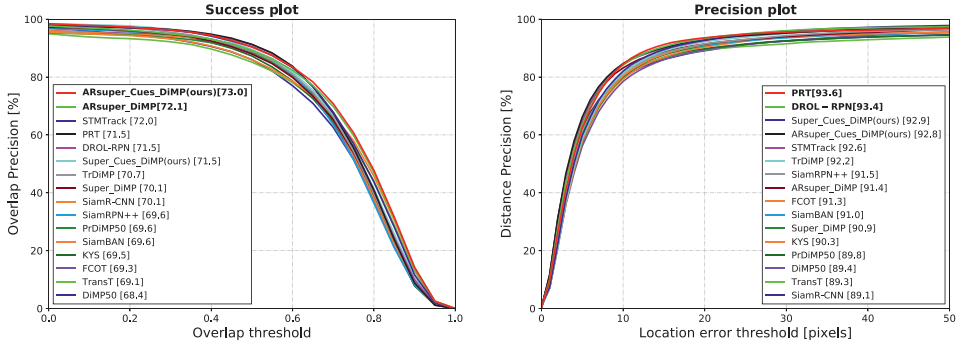
Arch.	Tracker	Succ.	P_{norm}
Tracking-By-Detection	SiamFC+ [25]	55.7	58.9
	ATOM [36]	51.5	57.6
	Nocal-Siam [52]	53.3	-
	FCOT [40]	56.9	67.8
	DiMP50 [15]	56.9	65.0
	PrDiMP50 [14]	59.8	68.8
	SiamBAN [71]	51.4	59.8
	TrDiMP [41]	63.9	-
	Super_DiMP [23]	63.0	71.9
	ARSuper_DiMP [19]	65.3	73.6
	Super_Cues_DiMP	64.5	74.1
	ARSuper_Cues_DiMP	66.7	75.2
Others	STMTrack [49]	60.6	69.3
	TransT [48]	64.9	73.8
	Siam R-CNN [56]	64.8	72.2

Table 5
Comparison with State-of-the-art trackers on Trackingnet dataset.

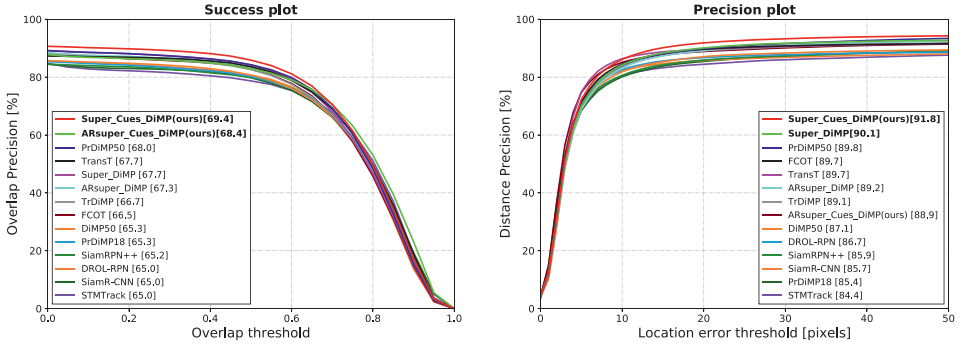
Arch.	Tracker	Prec.	Pnorm	Succ.
Tracking-By-Detection	SiamFC+ [25]	70.5	80.0	75.4
	ATOM [36]	64.8	77.1	70.3
	SiamRPN+ [29]	69.4	80.0	73.3
	FCOT [40]	72.3	82.8	75.1
	DiMP50 [15]	68.7	80.1	74.0
	PrDiMP50 [14]	70.4	81.6	75.8
	KYS [38]	68.8	80.0	74.0
	TrDiMP [41]	73.1	83.3	78.4
	Super_DiMP [23]	73.3	83.4	78.1
	ARSuper_DiMP [19]	78.3	85.6	80.5
	Super_Cues_DiMP	74.1	83.7	78.5
	ARSuper_Cues_DiMP	78.7	86.0	80.9
Others	STMTrack [49]	76.7	85.1	80.3
	TransT [48]	80.3	86.7	81.4
	Siam R-CNN [56]	80.0	85.4	81.2

Table 6
Comparison with State-of-the-art trackers on GOT-10K dataset.

Arch.	Tracker	AO	$SR_{0.5}$	$SR_{0.75}$
Tracking-By-Detection	Nocal-Siam [52]	60.1	68.8	-
	FCOT [40]	64.0	76.3	51.7
	DiMP50 [15]	61.1	71.7	49.2
	PrDiMP50 [14]	63.4	73.8	54.3
	KYS [38]	63.6	75.1	51.5
	TrDiMP [41]	68.8	80.5	59.7
	Super_DiMP [23]	67.5	78.8	59.5
	ARSuper_DiMP [19]	70.1	80.0	64.2
	Super_Cues_DiMP	68.5	79.7	60.7
	ARSuper_Cues_DiMP	70.5	80.0	65.3
Others	STMTrack [49]	64.2	73.7	57.5
	TransT [48]	72.3	82.4	68.2
	Siam R-CNN [56]	64.9	72.8	59.7



(a) OTB-100 (b) OTB-100



(c) UAV123 (d) UAV123

Fig. 5. Success and Precision measured using the OTB-100 and UAV123 datasets.

Table 7
Comparison with State-of-the-art trackers on VOT2020.

Arch.	Tracker	EAO	A	R
Tracking-By-Detection	ATOM [36]	27.1	46.2	73.4
	DiMP50 [15]	27.4	45.7	74.0
	OceanPlus [72]	49.1	70.0	74.2
	PRT [34]	53.0	70.0	86.9
	Ocean [73]	43.0	69.3	75.4
	Siammask [74]	32.1	62.4	64.8
	D3S [75]	43.9	69.9	76.9
	Super_DiMP [23]	30.5	47.7	78.6
	ARSuper_DiMP [19]	48.2	75.4	77.7
	Super_Cues_DiMP	30.3	47.9	77.5
	ARSuper_Cues_DiMP	49.6	72.7	80.7
Others	STMTrack [49]	-	-	-
	TransT [48]	48.5	-	-
	Siam R-CNN [56]	35.5	69.9	58.6

trackers with relative gains of 0.4% and 1% respectively in term of AUC.

For the results of VOT2020 [70], as shown in Table 7, ARSuper_Cues_DiMP outperforms the baseline ARSuper_DiMP by 1.4% and 3% in terms of EAO and robustness respectively.

Of all the Tracking-By-Detection trackers tested, ARSuper_Cues_DiMP was the best performing on five out of seven datasets (OTB-100, NFS, LaSOT, Trackingnet, and GOT-10k). No other Tracking-By-Detection method was best performing on more than one dataset, and for one of those benchmarks (UAV123) the best performing tracker was our other method, Super_DIM_DiMP. The results also show the proposed method to be more effective than other recent distractor-suppression methods such as KYS [38] and Nocal-Siam [52].

Table 8
Results of ablation experiments using Super_Cues_DiMP on OTB-100 dataset. Shape means increasing shape sensitivity by adding the styled-training dataset in training. Motion means using the proposed motion method.

Component	Super_DiMP	Proposed		
Shape		√		√
Motion			√	√
AUC	70.1	70.6	70.7	**71.5**
FPS	**27**	**27**	17	17

5. Ablation study

5.1. Influence of different components

To verify the contributions of each component in the proposed approach, we conducted ablation experiments using Super_Cues_DiMP on the OTB-100 dataset. As shown in Table 8, the AUC increases to 70.6 form 70.1 when styled-training data is used. Similarity, when the motion method is adopted in inference, the performance increases to 70.7. The combination of the two parts achieves the best performance with AUC equal to 71.5.

5.2. Parameter sensitivity

The proposed motion method employs a number of parameters:

- The number of frames, n_s, between resets of the Kalman filter, see Section 3.2;
- The global threshold, g, applied to the score map to detect peaks caused by distractors, see Section 3.3;
- The number of previous velocities, n_p, used to detect near-linear motion of the target, see Section 3.3;

Table 9
Evaluation of the sensitivity of the proposed motion predictor to parameter values using the OTB-100 dataset. Note the value of g can not be more than 1, thus the factors in brackets are used to change its value. n_p needs to be an integer, so its modified value is rounded up to the nearest integer value. Super_Cues_DiMP without the motion module had an AUC equal to 70.6. Using the proposed motion module with the standard parameter values the AUC was equal to 71.5.

Parameter	Standard	AUC when value changed by:			
	value	÷5	÷2	×2(1.05)	×5(1.15)
n_s	20	71.2	71.4	71.0	70.8
g	0.85	70.4	70.6	71.3	70.8
n_p	5	70.6	70.9	71.5	70.9
σ_a	10	70.6	71.4	71.2	70.9
σ_g	0.5	70.8	71.4	71.3	71.2

- The standard deviation of the angles, σ_a, used to detect near-linear motion of the target, see Section 3.3;
- The standard deviation of two-dimensional gaussian, σ_g, used to define the $\mathbf{P}(\mathbf{X})$, see Section 3.3;

The effect these parameters have on the performance of the tracker was evaluated by varying the value of one parameter while keeping the other parameters fixed at their default values. This experiment was conducted using the OTB-100 dataset, and the results are shown in Table 9.

The algorithm was tolerant to large changes in n_s, and σ_g. However, increasing the value of n_s by a factor of 5 from its default value resulted in the prediction of the Kalman filter being unreliable due to the changes in trajectory of some targets. This in turn resulted in only a minor performance gain compared to the underlying tracker alone. Using a vary small σ_g resulted in an AUC only marginally above that of the underlying tracker, which indicates that when $\mathbf{P}(\mathbf{X})$ is restricted to a very small region it can suppress the value of $\mathbf{S}(\mathbf{X})$ corresponding to the true target location.

It can be seen that when the value of g was increased by a factor of 1.05, or n_p was increased by a factor of 5, from its default value, the algorithm still produced state-of-the art performance. However, increasing either parameter further resulted in only a minor gain in performance. This is not surprising as the modification of the score map by the motion module (using Eq. (9) and (10)) is less likely to be performed if n_p or g is too large. Decreasing g also reduced performance, and an extreme reduction in g could lead to worse performance than the underlying tracker alone. The score map may contain small amplitude peaks, especially if the velocity of the target changes suddenly. In such circumstance a small value of g means that the proposed method is performed frequently. However, due to the changing velocity of the target, the motion module predictions may be poor and not help to identify the true target location. When the value of n_p was decreased by a factor of 5, only the last velocity was used to calculate the standard deviation, which was therefore always equal to zero. Consequently, the non-linear motion of the target was not detected and excluded, causing a poor result.

The tracker's performance was tolerant to increases and decreases in the value of σ_a. However when σ_a was too small the modification of the score map by the motion module (using Eq. (9) and (10)) was rarely performed even for targets with predictable, linear, trajectories. In contrast, when the value of σ_a was too large, the modification of the score map by the motion module was performed even when the target trajectory was erratic and the prediction unreliable. In both cases this lead to a reduction in performance.

Table 10
The results of SiamFC and SiamFC++ trained with the same strategies used in Table 1.

(a) Results of four SiamFC models on three benchmarks.

Name	OTB-100 AUC	UAV123 AUC	GOT-10K AO
Model_A	**58.1**	50.2	34.6
Model_B	54.9	51.0	30.4
Model_C	56.2	50.0	30.8
Model_D	58.0	**51.1**	**35.1**

(b) Results of four SiamFC+ models on three benchmarks.

Name	OTB-100 AUC	UAV123 AUC	GOT-10K AO
Model_A	62.8	58.9	53.8
Model_B	**65.3**	59.3	51.7
Model_C	63.5	57.6	52.9
Model_D	64.2	**60.3**	**54.1**

5.3. Qualitative evaluation

Fig. 6 provides a qualitative comparison of Super_Cues_DiMP with the baseline tracker Super_DiMP. The effects of training Super_DiMP to be more sensitive to shape cues are illustrated on the first row. When the target is fully occluded the classifier of Super_DiMP regards a distractor as the target, as shown in frame 248. In contrast, the classifier of proposed tracker, due to the use of more discriminative learned features, is not fooled by the distractor and is able to relocate the target after it reappears. Similarly, in the second row of Fig. 6, even though the target object is not occluded, Super_DiMP begins to track a distractor object, while the proposed tracker, due to its increased shape bias is able to track the target accurately. The effects of the motion module are illustrated on the third row of Fig. 6. In this example, Super_DiMP incorrectly starts to track a similar looking distractor in frame 32. However, this distractor is moving left-to-right, while the true target is moving right-to-left. The proposed tracker is able to use the motion of the target to avoid making the same mistake.

6. Transferability

The experiments above are based on the discriminative tracker, Super_DiMP [23]. To test the transferability of our methods, we also combined them with the famous generative tracker SiamFC [24] and a current state-of-the-art generative tracker, SiamFC++ [25] (the AlexNet version). The two trackers were trained using the training split of GOT-10K with the same strategies used in Table 1. The training details are shown in Appendix A. The resulting tracking performance, summarised in Table 10, is generally better for Model_D, which again demonstrates that slightly increasing the shape bias of a visual tracker can improve its robustness. In contrast to the results for Super_DiMP (Table 2), the worst results were not always produced by Model_B and Model_C, suggesting that the two generative trackers can make use of more shape information than the discriminative tracker Super_DiMP.

Motion cues were integrated with the two new generative trackers trained using Model_D, using the methods set out in Section 3.1. The global threshold g was set to 0.75 for SiamFC and 0.65 for SiamFC++ and other hyper-parameters were kept at the standard values. Optimising the parameters carefully for each tracker may result in better performance. The resulting methods were evaluated on the OTB-100 and UAV123 dataset using a single Nvidia Tesla V100 GPU. The speed of SiamFC and DaSiamRPN are around 210 FPS and 186 FPS respectively. When integrating the motion module, their speeds are 75 FPS and 64 FPS respectively. Different from the two proposed discriminative trackers, the

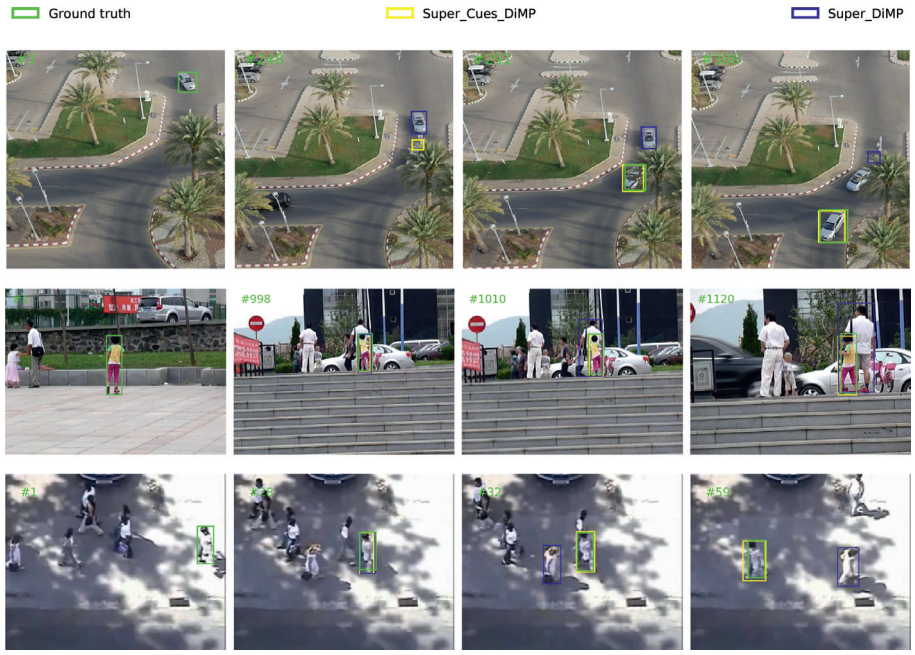


Fig. 6. Qualitative comparison of Super_DiMP and Super_Cues_DiMP on hard scenarios. Each row shows frames from a video: *uav_car7* from UAV123, *Girl2* from OTB-100, and *Crowds* from OTB-100. Note that the annotations provided with the UAV123 data, define the bounding box coordinates as NaNs when the target is out-of-frame or fully occluded, thus the ground truth bounding box is not shown in frame 248 of the video *uav_car7*. Note that for *uav_car7* and *Crowds* cropped regions of each video are shown to improve the visibility of the bounding boxes.

Table 11
Evaluation of proposed motion method with shape-sensitive generate trackers.

Tracker		OTB-100		UAV123	
		Succ.	Prec.	Succ.	Prec.
SiamFC [24]	Model_D	58.0	77.9	51.1	71.4
	Proposed	58.2	78.0	51.4	72.7
	Increment	0.2	0.1	0.3	1.3
SiamFC+ [25]	Model_D	64.2	85.5	60.3	77.2
	Proposed	64.3	85.8	60.6	77.8
	Increment	0.1	0.3	0.3	0.6

decrements of their speed are significant. The Kalman filter is executed in CPU while the SEKD and underlying trackers run on GPU, the data transfer between CPU and GPU spends more time as the speed of underlying tracker increases. A GPU implementation of the Kalman filter may speed these trackers. As shown in Table 11, the motion module improves the performance of both underlying trackers. The above results indicate our methods have the potential to be a general approach that could improve the performance for most visual trackers.

7. Conclusions and future work

This study focuses on improving the robustness of Tracking-by-Detection trackers against distractors. We demonstrate that training visual trackers to be more sensitive to shape cues can improve their performance and that motion prediction used in traditional trackers can still play an important role for modern trackers. By making the appearance model more sensitive to shape cues in offline training and using the historical locations of the target to predict its future position during online inference, a novel distractor suppression method was proposed which can be combined with many existing trackers. Extensive experiments indicate the proposed methods have the potential to improve the performance of many exiting tracking algorithms, and when combined with a state-of-the-art tracker the resulting tracker is shown to outperform all popular state-of-the-art methods on the OTB-100, NFS, and LaSOT datasets, and to produce performance that is competitive with the state-of-the-art on the UAV123, Trackingnet, GOT-10K, and VOT2020 datasets.

Although the proposed approach has shown favorable results, the proposed motion method can only handle situations where the movement of tracking target is near-linear and it is based on the Kalman filter which doesn't employ the power of deep learning. Future work might explore a sophisticated network for both appearance and motion information. A two stage network might be useful where the first stage is used to learn target appearance and output an appearance score map while the second stage exploit information about the past motion of the tracked target and other objects in the surrounding scene to verify the result of first stage and thus avoid wrongly identifying distractors as targets. For the second stage, RC-VAE [76] might be useful as this method infers continuous time latent position and velocity trajectories given past frames and locations of objects.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Bo Gao: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Michael W. Spratling:** Writing – original draft, Writing – review & editing.

Acknowledgement

This research was funded by the China Scholarship Council.The authors acknowledge use of the research computing facility at Kings College London, Rosalind (https://rosalind.kcl.ac.uk), and the Joint Academic Data science Endeavour (JADE) facility.

Appendix A. Training details for SiamFC and SiamFC++

The four SiamFC models, and the four SiamFC++ models, were trained by the same approach that we trained Super_DiMP models as summarised in Table 1. Model_A was trained using the standard GOT-10K training set. Model_B was trained on the Stylised-GOT-10K training set. Model_C was trained on a dataset containing the images from both GOT-10K and Stylised-GOT-10K. Model_D was initialised with the weights of Model_C and then fine-tuning on GOT-10K.

For SiamFC Model_D was fine-tuned for 35 epochs with learning rate annealed geometrically at each epoch from 10^{-3} to 10^{-6}. The number of epochs was 50 for Model_A and Model_B and 25 for Model_C.

For SiamFC++ Model_D was fine-tuned with 1 warm up epoch with learning rate linearly increased from 10^{-7} to 2×10^{-3}, then using a cosine annealing learning rate schedule for the rest of 13 epochs. The number of epochs was 20 for Model_A and Model_B and 10 for Model_C.

The other training hyperparameters used for each model were the same as the official settings [24,25].

References

[1] A. Mangawati, M. Leesan, H.R. Aradhya, et al., Object tracking algorithms for video surveillance applications, in: 2018 International Conference on Communication and Signal Processing (ICCSP), IEEE, 2018, pp. 0667–0671.

[2] Z. Pan, S. Liu, A.K. Sangaiah, K. Muhammad, Visual attention feature (VAF): a novel strategy for visual tracking based on cloud platform in intelligent surveillance systems, J. Parallel Distrib. Comput. 120 (2018) 182–194.

[3] G. Prabhakar, B. Kailath, S. Natarajan, R. Kumar, Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving, in: 2017 IEEE Region 10 Symposium (TENSYMP), IEEE, 2017, pp. 1–6.

[4] H. Cho, Y.-W. Seo, B.V. Kumar, R.R. Rajkumar, A multi-sensor fusion system for moving object detection and tracking in urban driving environments, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 1836–1843.

[5] I.F. Mondragón, P. Campoy, C. Martinez, M.A. Olivares-Méndez, 3d pose estimation based on planar object tracking for UAVs control, in: 2010 IEEE International Conference on Robotics and Automation, Ieee, 2010, pp. 35–41.

[6] M. Tarhan, E. Altuğ, A catadioptric and pan-tilt-zoom camera pair object tracking system for UAVs, J. Intell. Robot. Syst. 61 (1–4) (2011) 119–134.

[7] S.-K. Weng, C.-M. Kuo, S.-K. Tu, Video object tracking using adaptive kalman filter, J. Vis. Commun. Image Represent. 17 (6) (2006) 1190–1208.

[8] Z. Fu, Y. Han, Centroid weighted kalman filter for visual object tracking, Measurement 45 (4) (2012) 650–655.

[9] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 564–577.

[10] E. Maggio, A. Cavallaro, Hybrid particle filter and mean shift tracker with adaptive transition model, in: Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005., volume 2, IEEE, 2005, pp. ii–221.

[11] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, A. Khajenezhad, A. Salahledin, A. Soltani-Farani, A. Zarezade, A. Petrosino, A. Milton, B. Bozorgtabar, B. Li, C.S. Chan, C. Heng, D. Ward, D. Kearney, D. Monekosso, H.C. Karaimer, H.R. Rabiee, J. Zhu, J. Gao, J. Xiao, J. Zhang, J. Xing, K. Huang, K. Lebeda, L. Cao, M.E. Maresca, M.K. Lim, M. El Helw, M. Felsberg, P. Remagnino, R. Bowden, R. Goecke, R. Stolkin, S.Y. Lim, S. Maher, S. Poullot, S. Wong, S. Satoh, W. Chen, W. Hu, X. Zhang, Y. Li, Z. Niu, The visual object tracking vot2013 challenge results, in: 2013 IEEE International Conference on Computer Vision Workshops, 2013, pp. 98–111, doi:10.1109/ICCVW.2013.20.

[12] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), volume 1, IEEE, 2005, pp. 886–893.

[13] J. Van De Weijer, C. Schmid, J. Verbeek, D. Larlus, Learning color names for real-world applications, IEEE Trans. Image Process. 18 (7) (2009) 1512–1523.

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

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

[16] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F.A. Wichmann, W. Brendel, Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, International Conference on Learning Representations (2018).

[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, International Conference on Learning Representations, (ICLR) (2015).

[18] G. Bo, M.W. Spratling, Robust template matching via hierarchical convolutional features from a shape biased CNN, The International Conference on Image, Vision and Intelligent Systems (ICIVIS 2021), 2022, pp. 333–344.

[19] B. Yan, X. Zhang, D. Wang, H. Lu, X. Yang, Alpha-refine: boosting tracking performance by precise bounding box estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 5289–5298.

[20] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: a benchmark, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2013, pp. 2411–2418.

[21] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, S. Lucey, Need for speed: A benchmark for higher frame rate object tracking, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1125–1134.

[22] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, H. Ling, Lasot: a high-quality benchmark for large-scale single object tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5374–5383.

[23] M. Danelljan, G. Bhat, Pytracking: Visual tracking library based on pytorch, 2019, https://github.com/visionml/pytracking/,accessed: 6/01/2020.

[24] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, P.H. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 850–865.

[25] Y. Xu, Z. Wang, Z. Li, Y. Yuan, G. Yu, SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines, in: The Association for the Advancement of Artificial Intelligence (AAAI), 2020, pp. 12549–12556.

[26] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, S. Wang, Learning dynamic siamese network for visual object tracking, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1763–1771.

[27] A. He, C. Luo, X. Tian, W. Zeng, Towards a better match in siamese network based visual object tracker, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018. 0–0

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

[29] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, J. Yan, SiamRPN++: evolution of siamese visual tracking with very deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4282–4291.

[30] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P.H.S. Torr, End-to-end representation learning for correlation filter based tracking, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[31] L. Bo, Y. Junjie, W. Wei, Z. Zheng, H. Xiaolin, High performance visual tracking with siamese region proposal network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 8971–8980.

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

[33] J. Zhou, P. Wang, H. Sun, Discriminative and robust online learning for siamese visual tracking, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34(07), 2020, pp. 13017–13024.

[34] Z. Ma, L. Wang, H. Zhang, W. Lu, J. Yin, RPT: Learning point set representation for siamese visual tracking, European Conference on Computer Vision (ECCV), 2020, pp. 653-665.

[35] J. Zhou, B. Li, L. Qiao, P. Wang, W. Gan, W. Wu, J. Yan, W. Ouyang, Higher performance visual tracking with dual-modal localization, arXiv:2103.10089(2021).

[36] M. Danelljan, G. Bhat, F.S. Khan, M. Felsberg, ATOM: accurate tracking by overlap maximization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4660–4669.

[37] C. Ma, J.-B. Huang, X. Yang, M.-H. Yang, Robust visual tracking via hierarchical convolutional features, IEEE Trans Pattern Anal Mach Intell (2018).

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

[39] S. Gladh, M. Danelljan, F.S. Khan, M. Felsberg, Deep motion features for visual tracking, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 1243–1248.

[40] Y. Cui, C. Jiang, L. Wang, G. Wu, Fully convolutional online tracking, arXiv:2004.07109 (2020).

[41] N. Wang, W. Zhou, J. Wang, H. Li, Transformer meets tracker: Exploiting temporal context for robust visual tracking, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 1571-1580.

[42] X.-F. Zhu, X.-J. Wu, T. Xu, Z.-H. Feng, J. Kittler, Complementary discriminative correlation filters based on collaborative representation for visual object tracking, IEEE Trans. Circuit. Syst. Video Technol. 31(2) (2020) 557–568.

[43] T. Xu, Z. Feng, X.-J. Wu, J. Kittler, Adaptive channel selection for robust visual object tracking with discriminative correlation filters, Int. J. Comput. Vis. 129(5) (2021) 1359–1375.

[44] J. Wang, Y. He, Motion prediction in visual object tracking, In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 10374-10379.

[45] P. Gao, Q. Zhang, F. Wang, L. Xiao, H. Fujita, Y. Zhang, Learning reinforced attentional representation for end-to-end visual tracking, Inf. Sci. (Ny) 517 (2020) 52–67.
[46] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, C. Shen, Graph attention tracking, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems, 2017, 30.
[48] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, H. Lu, Transformer tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8126–8135.
[49] Z. Fu, Q. Liu, Z. Fu, Y. Wang, Stmtrack: Template-free visual tracking with space-time memory networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 13774–13783.
[50] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2018, pp. 7794–7803.
[51] B. Yan, H. Peng, J. Fu, D. Wang, H. Lu, Learning spatio-temporal transformer for visual tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10448–10457.
[52] H. Tan, X. Zhang, Z. Zhang, L. Lan, W. Zhang, Z. Luo, Nocal-siam: refining visual features and response with advanced non-local blocks for real-time siamese tracking, IEEE Trans. Image Process. (2021).
[53] B. Gao, M.W. Spratling, Explaining away results in more robust visual tracking, Vis. Comput. (2022) 1–15.
[54] R.T. Collins, Y. Liu, M. Leordeanu, Online selection of discriminative tracking features, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1631–1643.
[55] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE Trans. Pattern Anal. Mach. Intell. 34 (7) (2011) 1409–1422.
[56] P. Voigtlaender, J. Luiten, P.H. Torr, B. Leibe, Siam R-CNN: Visual tracking by re-detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 6578–6588.
[57] S. Xuan, S. Li, Z. Zhao, L. Kou, Z. Zhou, G.-S. Xia, Siamese networks with distractor-reduction method for long-term visual object tracking, Pattern Recognit. (2020) 107698.
[58] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[59] X. Huang, S. Belongie, Arbitrary style transfer in real-time with adaptive instance normalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1501–1510.
[60] Y. Li, Q. Yu, M. Tan, J. Mei, P. Tang, W. Shen, A. Yuille, C. Xie, Shape-texture debiased neural network training, International Conference on Learning Representations (ICLR) (2020).
[61] L. Huang, X. Zhao, K. Huang, GOT-10k: a large high-diversity benchmark for generic object tracking in the wild, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[62] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, B. Ghanem, Trackingnet: a large-scale dataset and benchmark for object tracking in the wild, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 300–317.
[63] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European conference on computer vision, Springer, 2014, pp. 740–755.
[64] Y. Song, L. Cai, J. Li, Y. Tian, M. Li, Sekd: Self-evolving keypoint detection and description, arXiv:2006.05077 (2020).
[65] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[66] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: European conference on computer vision, Springer, 2006, pp. 404–417.
[67] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: 2011 International conference on computer vision, Ieee, 2011, pp. 2564–2571.
[68] K. Saho, Kalman filter for moving object tracking: performance analysis and filter design, Kalman Filters-Theory for Advanced Applications, IntechOpen, 2017.
[69] M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for UAV tracking, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 445–461.
[70] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L.Č. Zajc, A. Lukežič, O. Drbohlav, et al., The eighth visual object tracking vot2020 challenge results, in: European Conference on Computer Vision, Springer, 2020, pp. 547–601.
[71] Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2020, PP. 6668-6677.
[72] Z. Zhang, B. Li, W. Hu, H. Peng, Towards accurate pixel-wise object tracking by attention retrieval, IEEE Trans. Image Process 30 (2021) 8553–8566.
[73] Z. Zhipeng, P. Houwen, F. Jianlong, L. Bing, H. Weiming, Ocean: Object-aware anchor-free tracking, in: European Conference on Computer Vision (ECCV), 2020.
[74] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, P.H. Torr, Fast online object tracking and segmentation: A unifying approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1328–1338.
[75] A. Lukezic, J. Matas, M. Kristan, D3s-a discriminative single shot segmentation tracker, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7133–7142.
[76] M. Qi, J. Qin, Y. Wu, Y. Yang, Imitative non-autoregressive modeling for trajectory forecasting and imputation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.