# Know Your Surroundings: Exploiting Scene Information for Object Tracking

Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte

CVL, ETH Zürich, Switzerland

**Abstract.** Current state-of-the-art trackers rely only on a target appearance model in order to localize the object in each frame. Such approaches are however prone to fail in case of e.g. fast appearance changes or presence of distractor objects, where a target appearance model alone is insufficient for robust tracking. Having the knowledge about the presence and locations of other objects in the surrounding scene can be highly beneficial in such cases. This scene information can be propagated through the sequence and used to, for instance, explicitly avoid distractor objects and eliminate target candidate regions.
In this work, we propose a novel tracking architecture which can utilize scene information for tracking. Our tracker represents such information as dense localized state vectors, which can encode, for example, if a local region is target, background, or distractor. These state vectors are propagated through the sequence and combined with the appearance model output to localize the target. Our network is learned to effectively utilize the scene information by directly maximizing tracking performance on video segments. The proposed approach sets a new state-of-the-art on 3 tracking benchmarks, achieving an AO score of 63.6% on the recent GOT-10k dataset.

## 1 Introduction

Generic object tracking is one of the fundamental computer vision problems with numerous applications. The task is to estimate the state of a target object in each frame of a video sequence, given only its initial appearance. Most current approaches [3, 8, 16, 25, 30, 33, 36] tackle the problem by learning an appearance model of the target in the initial frame. This model is then applied in subsequent frames to localize the target by distinguishing its appearance from the surrounding background. While achieving impressive tracking performance [24, 28], these approaches rely *only* on the appearance model, and do not utilize any other information contained in the scene.

In contrast, humans exploit a much richer set of cues when tracking an object. We have a holistic view of the scene and take into consideration not only the target object, but also other objects in the scene. Such information is helpful when localizing the target, e.g. in case of cluttered scenes with distractor objects, or when the target undergoes fast appearance change. Consider the example in Figure 1. Given only the initial target appearance, it is hard to confidently locate

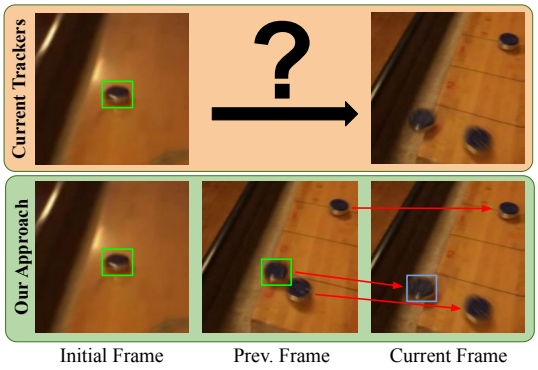



**Fig. 1.** Current approaches (top) only utilize an appearance model to track the target object. However, such a strategy fails in the above example. Here, the presence of distractor objects makes it virtually impossible to localize the target based on appearance only, even if the appearance model is continuously updated using previous frames. In contrast, our approach (bottom) is also aware of other objects in the scene. This scene information is propagated through the sequence by computing a dense correspondence (red arrows) between consecutive frames. The propagated scene knowledge greatly simplifies the target localization problem, allowing us to reliably track the target.

the target due to the presence of distractor objects. However, if we also utilize the previous frame, we can easily detect the presence of distractors. This knowledge can then be propagated to the next frame in order to reliably localize the target. While existing approaches update the appearance model with previously tracked frames, such a strategy by itself cannot capture the locations and characteristics of the other objects in the scene.

In this work, we aim to go beyond the conventional frame-by-frame detection-based tracking. We propose a novel tracking architecture which can propagate valuable scene information through the sequence. This information is used to achieve an improved *scene aware* target prediction in each frame. The scene information is represented using a dense set of localized state vectors. These state vectors encode valuable information about the local region, e.g. whether the region corresponds to the target, background or a distractor object. As the regions move through a sequence, we propagate the corresponding state vectors by utilizing dense correspondence maps between frames. Consequently, our tracker is 'aware' of every object in the scene and can use this information in order to e.g. avoid distractor objects. This scene knowledge, along with the target appearance model, is used to predict the target state in each frame using a learned predictor module. The scene information captured by the state representation is then updated using a recurrent neural network module.

We perform comprehensive experiments on six challenging benchmarks: VOT-2018 [28], GOT-10k [24], TrackingNet [35], OTB-100 [45], NFS [14], and LaSOT [13]. Our approach achieves state-of-the-art results on five datasets. On the challenging GOT-10k dataset, our tracker obtains an average overlap (AO)

score of 63.6%, outperforming the previous best approach by 2.5%. We also provide an ablation study analyzing the impact of key components in our tracker.

## 2 Related Work

Most tracking approaches tackle the problem by learning an appearance model of the target in the first frame. A popular method to learn the target appearance model is the discriminative correlation filters (DCF) [5, 9, 10, 22, 26, 32]. These approaches exploit the convolution theorem to efficiently train a classifier in the Fourier domain using the circular shifts of the input image as training data. Another approach is to train or fine-tune a few layers of a deep neural network in the first frame to perform target-background classification [3, 8, 36, 39]. MDNet [36] fine-tunes three fully-connected layers online, while DiMP [3] employs a meta-learning formulation to predict the weights of the classification layer. In recent years, Siamese networks have received significant attention [2, 19, 30, 31, 44]. These approaches address the tracking problem by learning a similarity measure, which is then used to locate the target.

The discriminative approaches discussed above exploit the background information in the scene to learn the target appearance model. A number of attempts have also been made to integrate background information into the appearance model in Siamese trackers [29, 51, 53]. However, in many cases, the distractor object is indistinguishable from a previous target appearance. Thus, a single target model is insufficient to achieve robust tracking in such cases. Further, in case of fast motion, it is hard to adapt the target model quickly to new distractors. In contrast to these works, our approach explicitly encodes localized information about different image regions and propagates this information through the sequence via dense matching. More related to our work, [46] aims to exploit the locations of distractors in the scene. However, it employs hand-crafted rules to classify image regions into background and target candidates independently in each frame. In contrast, we present a fully *learnable* solution, where the encoding of image regions is learned and propagated between frames. Further, our final prediction is obtained by combining the explicit background representation with the appearance model output.

In addition to appearance cues, a few approaches have investigated the use of optical flow information for tracking. Gladh et al [17] utilize deep motion features extracted from optical flow images to complement the appearance features when constructing the target model. Zhu et al [54] use optical flow to warp the feature maps from the previous frames to a reference frame and aggregate them in order to learn the target appearance model. However, both these approaches utilize optical flow to only improve the robustness of the target model. In contrast, we explicitly use dense motion information to propagate information about background objects and structures in order to complement the target model.

Some works have also investigated using recurrent neural networks (RNN) for object tracking. Gan et al [15] use a RNN to directly regress the target location using image features and previous target locations. Ning et al [37] utilize the

YOLO [38] detector to generate initial object proposals. These proposals, along with the image features, are passed through an LSTM [23] to obtain the target box. Yang et al [49, 50] use an LSTM to update the target model to account for changes in target appearance through a sequence.

## 3 Proposed Method

We develop a novel tracking architecture capable of exploiting scene information to improve tracking performance. While current state-of-the-art methods [3, 8, 30] rely only on the target appearance model to process every frame independently, our approach also propagates information about the scene from previous frames. This provides rich cues about the environment, e.g. the location of distractor objects, which greatly aids the localization of the target.

A visual overview of our tracking architecture is provided in Figure 2. Our tracker internally tracks *all* regions in the scene, and propagates any information about them that helps localization of the target. This is achieved by maintaining a state vector for every region in the target neighborhood. The state vector can, for instance, encode whether a particular patch corresponds to the target, background, or a distractor object that is likely to fool the target appearance model. As the objects move through a sequence, the state vectors are propagated accordingly by estimating a dense correspondence between consecutive frames. The propagated state vectors are then fused with the target appearance model output in order to predict the final target confidence values used for localization. Lastly, the outputs of the predictor and the target model are used to update the state vectors using a convolutional gated recurrent unit (ConvGRU) [1].

### 3.1 Tracking with Scene Information

Our tracker predictions are based on two cues: (i) appearance in the current frame and (ii) scene information propagated over time. The appearance model $\tau$ aims to distinguish the target object from the background. By taking the deep feature map $x_t \in \mathbb{R}^{W \times H \times D}$ extracted from frame $t$ as input, the appearance model $\tau$ predicts a score map $s_t = \tau(x_t) \in \mathbb{R}^{W \times H}$. Here, the score $s_t(\mathbf{r})$ at every spatial location $\mathbf{r} \in \Omega := \{0, \dots, W-1\} \times \{0, \dots, H-1\}$ denotes the likelihood of that location being the target center.

The target model has the ability to recover from occlusions and provides long-term robustness. However, it is oblivious to the contents of the surrounding scene. In order to extract such information, our tracker maintains a state vector for every region in the target neighborhood. Concretely, for every spatial location $\mathbf{r} \in \Omega$ in the deep feature representation $x_t$, we maintain a $S$-dimensional state vector $h^{\mathbf{r}}$ for that cell location such that $h \in \mathbb{R}^{W \times H \times S}$. The state vectors contain information about the cell which is beneficial for single target tracking. For example, it can encode whether a particular cell corresponds to the target, background, or is in fact a distractor that looks similar to the target. Note that we

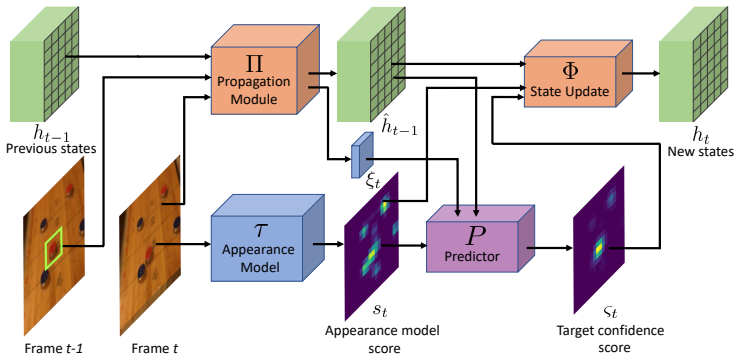



**Fig. 2.** An overview of our tracking architecture. In addition to using a target appearance model $\tau$, our tracker also exploits propagated scene information in order to track the target. The information about each image region is encoded within localized states $h$. Given the states $h_{t-1}$ from the previous frame, the propagation module $\Pi$ maps these states from the previous frame to the current frame locations. These propagated states $\hat{h}_{t-1}$, along with the propagation reliability $\xi_t$ and appearance model score $s_t$ are used by the predictor $P$ to output the final target confidence scores $\varsigma_t$. The state update module $\Phi$ then uses the current frame predictions to provide the new states $h_t$.

do not explicitly enforce any such encoding, but let $h$ be a generic representation whose encoding is trained end-to-end by minimizing a tracking loss.

The state vectors are initialized in the first frame using a small network $\Upsilon$ which takes the first frame target annotation $B_0$ as input. The network generates a single-channel label map specifying the target location. This is passed through two convolutional layers to obtain the initial state vectors $h_0 = \Upsilon(b_0)$. The state vectors contain localized information specific to their corresponding image regions. Thus, as the objects move through a sequence, we propagate their state vectors accordingly. Given a new frame $t$, we transform the states $h_{t-1}$ from the previous frame locations to the current frame locations. This is performed by our state propagation module $\Pi$,

$$(\hat{h}_{t-1}, \xi_t) = \Pi(x_t, x_{t-1}, h_{t-1}) \tag{1}$$

Here, $x_t \in \mathbb{R}^{W \times H \times D}$ and $x_{t-1} \in \mathbb{R}^{W \times H \times D}$ are the deep feature representations from the current and previous frames, respectively. The output $\hat{h}_{t-1}$ represents the spatially propagated state, compensating for the object and camera motions. The propagation reliability map $\xi_t \in \mathbb{R}^{W \times H}$ indicates the reliability of the state propagation. That is, a high $\xi_t(\mathbf{r})$ indicates that the state $\hat{h}^{\mathbf{r}}_{t-1}$ at $\mathbf{r}$ has been confidently propagated. The reliability map $\xi_t$ can thus be used to determine whether to trust a propagated state vector $\hat{h}^{\mathbf{r}}_{t-1}$ when localizing the target.

In order to predict the location of the target object, we utilize *both* the appearance model output $s_t$ and the propagated states $\hat{h}_{t-1}$. The latter captures valuable information about all objects in the scene, which complements the target-centric information contained in the appearance model. We input the

propagated state vectors $\hat{h}_{t-1}$, along with the reliability scores $\xi_t$ and the appearance model prediction $s_t$ to the predictor module $P$. The predictor combines these information to provide the fused target confidence scores $\varsigma_t$,

$$\varsigma_t = P(\hat{h}_{t-1}, \xi_t, s_t) \tag{2}$$

The target is then localized in frame $t$ by selecting the location $\mathbf{r}^*$ with the highest score: $\mathbf{r}^* = \arg\max_{\mathbf{r} \in \Omega} \varsigma_t$. Finally, we use the fused confidence scores $\varsigma_t$ along with the appearance model output $s_t$ to update the state vectors,

$$h_t = \Phi(\hat{h}_{t-1}, \varsigma_t, s_t) \tag{3}$$

The recurrent state update module $\Phi$ can use the current frame information from the score maps to e.g. reset an incorrect state vector $\hat{h}^{\mathbf{r}}_{t-1}$, or flag a newly entered object as a distractor. These updated state vectors $h_t$ are then used to track the object in the next frame. Our tracking procedure is detailed in Alg. 1.

### 3.2 State propagation

The state vectors contain localized information for every region in the target neighborhood. As these regions move through a sequence due to e.g. object or camera motion, we need to propagate their states accordingly, in order to compensate for their motions. This is done by our state propagation module $\Pi$. The inputs to this module are the deep feature maps $x_{t-1}$ and $x_t$ extracted from the previous and current frames, respectively. Note that the deep features $x$ are not required to be the same as the ones as used for the target model. However, we assume that both feature maps have the same spatial resolution $W \times H$.

In order to propagate the states from the previous frame to the current frame locations, we first compute a dense correspondence between the two frames. We represent this correspondence as a probability distribution $p$, where $p(\mathbf{r}'|\mathbf{r})$ is the probability that location $\mathbf{r} \in \Omega$ in the current frame originated from $\mathbf{r}' \in \Omega$ in the previous frame. The dense correspondence is estimated by constructing a 4D cost volume $\mathbf{CV} \in \mathbb{R}^{W \times H \times W \times H}$, as is commonly done in optical flow approaches [12, 42, 47]. The cost volume contains a matching cost between every image location pair from the previous and current frame. The element $\mathbf{CV}(\mathbf{r}', \mathbf{r})$ in the cost volume is obtained by computing the correlation between $3 \times 3$ windows centered

**Algorithm 1** Our tracking loop

**Input:** Image features $\{x_t\}_{t=0}^{N}$, initial annotation $b_0$, appearance model $\tau$

```
h_0 ← Υ(b_0)                                  # Initialize states
for i = 1, ..., N do                          # For every frame
    s_t ← τ(x_t)                              # Apply appearance model
    (ĥ_{t-1}, ξ_t) ← Π(x_t, x_{t-1}, h_{t-1}) # Propagate states
    ς_t ← P(ĥ_{t-1}, ξ_t, s_t)                # Predict target confidence scores
    h_t ← Φ(ĥ_{t-1}, ς_t, s_t)                # Update states
```

at $\mathbf{r}'$ in the previous frame features $x_{t-1}$ and $\mathbf{r}$ in the current frame features $x_t$. For computational efficiency, we only construct a partial cost volume by assuming a maximal displacement of $d_{\max}$ for every feature cell.

We process the cost volume through a network module to obtain robust dense correspondences. We pass the cost volume slice $\mathbf{CV}_{\mathbf{r}'}(\mathbf{r}) \in \mathbb{R}^{W \times H}$ for every cell $\mathbf{r}'$ in the previous frame, through two convolutional blocks in order to obtain processed matching costs $\phi(\mathbf{r}', \mathbf{r})$. Next, we take the softmax of this output over the current frame locations to get an initial correspondence $\phi'(\mathbf{r}', \mathbf{r}) = \frac{\exp\big(\phi(\mathbf{r}', \mathbf{r})\big)}{\sum_{\mathbf{r}'' \in \Omega} \exp(\phi(\mathbf{r}', \mathbf{r}''))}$. The softmax operation aggregates information over the current frame dimension and provides a soft association of locations between the two frames. In order to also integrate information over the previous frame locations, we pass $\phi'$ through two more convolutional blocks and take softmax over the previous frame locations. This provides the required probability distribution $p(\mathbf{r}'|\mathbf{r})$ at each current frame location $\mathbf{r}$.

The estimated correspondence $p(\mathbf{r}'|\mathbf{r})$ between the frames can now be used to determine the propagated state vector $\hat{h}^{\mathbf{r}}_{t-1}$ at a current frame location $\mathbf{r}$ by evaluating the following expectation over the previous frame state vectors.

$$\hat{h}^{\mathbf{r}}_{t-1} = \sum_{\mathbf{r}' \in \Omega} h^{\mathbf{r}'}_{t-1} p(\mathbf{r}'|\mathbf{r})\,. \tag{4}$$

When using the propagated state vectors $\hat{h}_{t-1}$ for target localization, it is also helpful to know if a particular state vector is valid i.e. if it has been correctly propagated from the previous frame. We can estimate this reliability $\xi^{\mathbf{r}}_t$ at each location $\mathbf{r}$ using the correspondence probability distribution $p(\mathbf{r}'|\mathbf{r})$ for that location. A single mode in $p(\mathbf{r}'|\mathbf{r})$ indicates that we are confident about the source of the location $\mathbf{r}$ in the previous frame. A uniformly distributed $p(\mathbf{r}'|\mathbf{r})$ on the other hand implies uncertainty. In such a scenario, the expectation (4) reduces to a simple average over the previous frame state vectors $h^{\mathbf{r}'}_{t-1}$, leading to an unreliable $\hat{h}^{\mathbf{r}}_{t-1}$. Thus, we use the negation of the shannon entropy of the distribution $p(\mathbf{r}'|\mathbf{r})$ to obtain the reliability score $\xi^{\mathbf{r}}_t$ for state $\hat{h}^{\mathbf{r}}_{t-1}$,

$$\xi^{\mathbf{r}}_t = \sum_{\mathbf{r}' \in \Omega} p(\mathbf{r}'|\mathbf{r}) \log(p(\mathbf{r}'|\mathbf{r})) \tag{5}$$

The reliability $\xi^{\mathbf{r}}_t$ is then be used to determine whether to trust the state $\hat{h}^{\mathbf{r}}_{t-1}$ when predicting the final target confidence scores.

### 3.3 Target Confidence Score Prediction

In this section, we describe our predictor module $P$ which determines the target location in the current frame. We utilize both the appearance model output $s_t$ and the scene information encoded by $\hat{h}_{t-1}$ in order to localize the target. The appearance model score $s^{\mathbf{r}}_t$ indicates whether a location $\mathbf{r}$ is target or background, based on the appearance in the current frame only. The state vector $\hat{h}^{\mathbf{r}}_{t-1}$ on the

other hand contains past information for *every* location $\mathbf{r}$. It can, for instance, encode whether the cell $\mathbf{r}$ was classified as target or background in the previous frame, how certain was the tracker prediction for that location, and so on. The corresponding reliability score $\xi_t^{\mathbf{r}}$ further indicates if the state vector $\hat{h}_{t-1}^{\mathbf{r}}$ is reliable or not. This can be used to determine how much weight to give to the state vector information when determining the target location.

The predictor module $P$ is trained to effectively combine the information from $s_t$, $\hat{h}_{t-1}$, and $\xi_t$ to output the final target confidence score $\varsigma_t \in \mathbb{R}^{W \times H}$. We concatenate the appearance model output $s_t$, the propagated state vectors $\hat{h}_{t-1}$, and the state reliability scores $\xi_t$ along the channel dimension, and pass the resulting tensor through two convolutional blocks. The output is then mapped to the range $[0, 1]$ by passing it through a sigmoid layer to obtain the intermediate scores $\hat{\varsigma}_t$. While it is possible to use this score directly, it is not reliable in case occlusions. This is because the state vectors corresponding to the target can leak into the occluding object, especially when two objects cross each other slowly, leading to incorrect prediction. In order to handle this, we pass $\hat{\varsigma}_t$ through another layer which masks the regions from the score map $\hat{\varsigma}_t$ where the appearance model score $s_t$ is less than a threshold $\mu$. Thus, we let the appearance model override the predictor output in case of occlusions. The final score map $\varsigma_t$ is thus obtained as $\varsigma_t = \hat{\varsigma}_t \cdot \mathbb{1}_{s_t > \mu}$. Here, $\mathbb{1}_{s_t > \mu}$ is an indicator function which evaluates to 1 when $s_t > \mu$ and is 0 otherwise and $\cdot$ denotes elementwise product. Note that the masking operation is differentiable and is implemented inside the network.

### 3.4 State update

While the state propagation described in Section 3.2 maps the state to the new frame, it does not update it with new information about the scene. This is accomplished by a recurrent neural network module, which evolves the state in each time step. As tracking information about the scene, we input the scores $s_t$ and $\varsigma_t$ obtained from the appearance model $\tau$ and the predictor module $P$, respectively. The update module can thus e.g. mark a new distractor object which entered the scene or correct corrupted states which have been incorrectly propagated. This state update is performed by the recurrent module $\Phi$ (eq. 3).

The update module $\Phi$ contains a convolutional gated recurrent unit (ConvGRU) [1, 6]. We concatenate the scores $\varsigma_t$ and $s_t$ along with their maximum values in order to obtain the input $f_t \in \mathbb{R}^{W \times H \times 4}$ to the ConvGRU. The propagated states from the previous frame $\hat{h}_{t-1}$ are treated as the hidden states of the ConvGRU from the previous time step. The ConvGRU then updates the previous states using the current frame observation $f_t$ to provide the new states $h_t$. A visualization of the representations used by our tracker is shown in Fig. 3.

### 3.5 Target Appearance Model

Our approach can be employed with any target appearance model. In this work, we use the DiMP tracker [3] as our target model component, due to its strong performance. DiMP is an end-to-end trainable tracking architecture that predicts

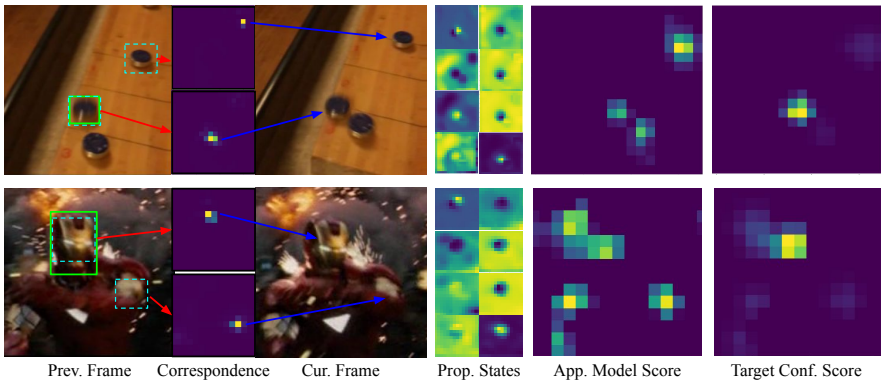



**Fig. 3.** Visualization of intermediate representations used for tracking on two example sequences. The green box in the previous frame (first column) denotes the target to be tracked. For every location in the current frame (third column), we plot the estimated correspondence with the marked region in the previous frame (second column). The states propagated to the current frame using the estimated correspondence are plotted channel-wise in the fourth column. The appearance model score (fifth column) fails to correctly localize the target in both cases due to the presence of distractors. In contrast, our approach can correctly handle these challenging scenarios and provides robust target confidence scores (last column) by exploiting the propagated scene information.

the appearance model $\tau_w$, parametrized by the weights $w$ of a single convolutional layer. The network integrates an optimization module that minimizes the following discriminative learning loss,

$$L(w) = \frac{1}{|S_{\text{train}}|} \sum_{(x,c)\in S_{\text{train}}} \|r(\tau_w(x), c)\|^2 + \|\lambda w\|^2 . \tag{6}$$

Here, $\lambda$ is the regularization parameter. The training set $S_{\text{train}} = \{(x_j, c_j)\}_{j=1}^{n}$ consists of deep feature maps $x_j$ extracted from the training images, and the corresponding target annotations $c_j$. The residual function $r(s, c)$ computes the error between the tracker prediction $s = \tau_w(x)$ and the groundtruth. We refer to [3] for more details about the DiMP tracker.

### 3.6 Offline Training

In order to train our architecture, it is important to simulate the tracking scenario. This is needed to ensure that the network can learn to effectively propagate the scene information over time and determine how to best fuse it with the appearance model output. Thus, we train our network using video sequences. We first sample a set of $N_{\text{train}}$ frames from a video, which we use to construct the appearance model $\tau$. We then sample a sub-sequence $V = \{(I_t, b_t)\}_{t=0}^{N_{\text{seq}}-1}$ consisting of $N_{\text{seq}}$ consecutive frames $I_t$ along with their corresponding target annotation $b_t$. We apply our network on this sequence data, as it would be during

tracking. We first obtain the initial state $h_0 = \Upsilon(b_0)$ using the state initializer $\Upsilon$. The states are then propagated to the next frame (Sec. 3.2), used to predict the target scores $\varsigma_t$ (Sec. 3.3), and finally updated using the predicted scores (Sec. 3.4). This procedure is repeated until the end of the sequence and the training loss is computed by evaluating the tracker performance over the whole sequence.

In order to obtain the tracking loss $L$, we first compute the prediction error $L_t^{\text{pred}}$ for every frame $t$ using the standard least-squares loss,

$$L_t^{\text{pred}} = \|\varsigma_t - z_t\|^2 \tag{7}$$

Here, $z_t$ is a label function, which we set to a Gaussian centered at the target. We also compute a prediction error $L_t^{\text{pred, raw}}$ using the raw score map $\hat{\varsigma}_t$ predicted by $P$ in order to obtain extra training supervision. To aid the learning of the state vectors and the propagation module $\mathit{\Pi}$, we add an additional auxiliary task. We use a small network head to predict whether a state vector $h_{t-1}^{\mathbf{r}}$ corresponds to the target or background. This prediction is penalized using a binary cross entry loss to obtain $L_t^{\text{state}}$. The network head is also applied on the propagated state vectors $\hat{h}_{t-1}^{\mathbf{r}}$ to get $L_t^{\text{state, prop}}$. This loss provides a direct supervisory signal to the propagation module $\mathit{\Pi}$.

Our final tracking loss $L$ is obtained as the weighted sum of the above individual losses over the whole sequence,

$$L = \frac{1}{N_{\text{seq}} - 1} \sum_{t=1}^{N_{\text{seq}}-1} L_t^{\text{pred}} + \alpha L_t^{\text{pred, raw}} + \beta (L_t^{\text{state}} + L_t^{\text{state, prop}}) \,. \tag{8}$$

The hyper-parameters $\alpha$ and $\beta$ determine the impact of the different losses. Note that the scores $s_t$ predicted by the appearance model can itself localize the target correctly in a majority of the cases. Thus, there is a risk that the predictor module only learns to rely on the target model scores $s_t$. To avoid this, we randomly add distractor peaks to the scores $s_t$ during training to encourage the predictor to utilize the scene information encoded by the state vectors.

### 3.7 Implementation details

We use a pre-trained DiMP model with ResNet-50 [20] backbone from [7] as our target appearance model. We use the block 4 features from the same backbone network as input to the state propagation module $\mathit{\Pi}$. For computational efficiency, our tracker does not process the full input image. Instead, we crop a square region containing the target, with an area $5^2$ times that of the target. The cropped search region is resized to $288 \times 288$ size, and passed to the network. We use $S = 8$ dimensional state vectors to encode the scene information. The threshold $\mu$ in the predictor $P$ is set to 0.05.

We use the training splits of TrackingNet [35], LaSOT [13], and GOT-10k [24] datasets to train our network. Within a sequence, we perturb the target position and scale in every frame in order to avoid learning any motion bias. While our network is end-to-end trainable, we do not fine-tune the weights for the backbone

network due to GPU memory constraints. Our network is trained for 40 epochs, with 1500 sub-sequences in each epoch. We use the ADAM [27] optimizer with an initial learning rate of $10^{-2}$, which is reduced by a factor of 5 every 20 epochs. We use $N_{\text{train}} = 3$ frames to construct the appearance model while the sub-sequence length is set to $N_{\text{seq}} = 50$. The loss weights are set to $\alpha = \beta = 0.1$.

During online tracking, we use a simple heuristic to determine target loss. In case the fused confidence score $\varsigma_t$ peak is smaller than a threshold (0.05), we infer that the target is lost and do not update the state vectors in this case. We impose a prior on the target motion by applying a window function on the appearance model prediction $s_t$ input to $P$, as well as the output target confidence score $\varsigma_t$. We also handle any possible drift in the target confidence scores. In case the appearance model scores $s_t$ and target confidence score $\varsigma_t$ only have small offset in their peaks, we use the appearance model score to determine the target location as it is more resistant to drift. After determining the target location, we use the bounding box estimation branch in DiMP to obtain the target box.

## 4 Experiments

We evaluate our proposed tracking architecture on six tracking benchmarks: VOT2018 [28], GOT-10k [24], TrackingNet [35], OTB-100 [45], NFS [14], and LaSOT [13]. Detailed results are provided in the supplementary material. Our tracker operates at around 20 FPS on a single Nvidia RTX 2080 GPU.

### 4.1 Ablation study

We conduct an ablation study to analyze the impact of each component in our tracking architecture. We perform experiments on the combined NFS [14] and OTB-100 [45] datasets consisting of 200 challenging videos. The trackers are evaluated using the overlap precision (OP) metric. The overlap precision $\text{OP}_T$ denotes the percentage of frames where the intersection-over-union (IoU) overlap between the tracker prediction and the groundtruth box is higher than a threshold $T$. The OP scores over a range of thresholds $[0, 1]$ are averaged to obtain the area-under-the-curve (AUC) score. We report the AUC and $\text{OP}_{0.5}$ scores for each tracker. Due to the stochastic nature of our appearance model, all results are reported as the average over 5 runs. Unless stated otherwise, we use the same training procedure and settings mentioned in Sections 3.6 and 3.7, respectively, to train all trackers evaluated in this section.

**Impact of scene information:** In order to study the impact of integrating scene information for tracking, we compare our approach with a tracker only employing target appearance model $\tau$. This version is equivalent to the standard DiMP-50 [3], which achieves state-of-the-art results on multiple tracking benchmarks. The results are reported in Tab. 1. Compared to using only the appearance model, our approach exploiting scene information provides an improvement of 1.3% in AUC score. These results clearly demonstrate that scene

**Table 1.** Impact of each component in our tracking architecture on the combined NFS and OTB-100 datasets. Compared to using only the appearance model, our approach integrating scene information, provides a significant 1.3% improvement in AUC score.

| | **Ours** | Only Appearance Model $\tau$ | No State Propagation $\Pi$ | No Propagation Reliability $\xi_t$ | No Appearance Model $\tau$ |
|---|---|---|---|---|---|
| AUC(%) | **66.4** | 65.1 | 64.9 | 66.1 | 49.2 |
| $OP_{0.5}$ | **83.5** | 81.9 | 81.2 | 82.9 | 60.1 |

knowledge contains complementary information that benefits tracking performance, even when integrated with a strong appearance model.

**Impact of state propagation:** Here, we analyze the impact of state propagation module (Sec. 3.2), which maps the localized states between frames by generating dense correspondences. This is performed by replacing the propagation module $\Pi$ in (1) and (4) with an identity mapping $\hat{h}_{t-1} = h_{t-1}$. That is, the states are no longer explicitly tracked by computing correspondences between frames. The results for this experiment are shown in Table 1. Interestingly, the approach without state propagation performs slightly worse (0.2% in AUC) than the network using only the appearance model. This shows that state propagation between frames is critical in order to exploit the localized scene information.

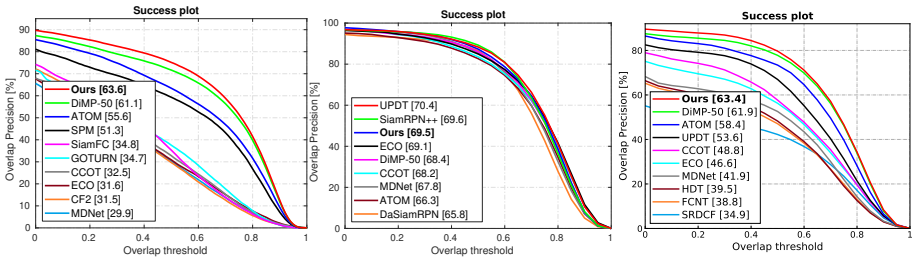
**Impact of propagation reliability:** Here, we study the impact of the propagation reliability score $\xi_t$ for confidence score prediction. We compare our approach with a baseline tracker which does not utilize $\xi_t$. The results indicate that using reliability score $\xi_t$ is beneficial, leading to a +0.3% AUC improvement.

**Impact of appearance model:** Our architecture utilizes the propagated scene information to *complement* the frame-by-frame prediction performed by the target appearance model. By design, our tracker relies on the appearance model to provide long-term robustness in case of e.g. occlusions, and thus is not suited to be used without it. However, for completeness, we evaluate a version of our tracker which does not utilize any appearance model. That is, we only use the propagated states $\hat{h}_{t-1}$, and the reliability score $\xi_t$ in order to track the target. As expected, not using an appearance model substantially deteriorates the performance by over 17% in AUC score.

### 4.2 State-of-the-art Comparison

In this section, we compare our proposed tracker with state-of-the-art approaches on six tracking benchmarks.

**VOT2018 [28]:** We evaluate our approach on the VOT2018 dataset consisting of 60 videos. The trackers are compared using the measures robustness and accuracy. Robustness indicates the number of tracking failures, while accuracy denotes the average overlap between tracker prediction and the ground-truth box. Both these measures are combined into a single expected average overlap (EAO) score. Results are shown in Tab. 2. Note that all top ranked approaches on VOT2018 utilize only a target appearance model for tracking. In contrast,




(a) GOT-10k (b) OTB-100 (c) NFS

**Fig. 4.** Success plots on GOT-10k (a), OTB-100 (b) and NFS (c). The AUC scores are shown in legend. Our approach obtains the best results on both GOT-10k and NFS datasets, outperforming the previous best method by 2.5% and 1.6% AUC, respectively.

our approach also exploits explicit knowledge about other objects in the scene. In terms of the overall EAO score, our approach outperforms the previous best method DiMP-50 with a large margin, achieving a relative gain of 5.0% in EAO.

**GOT10k [24]:** This is a recently introduced large scale dataset consisting of over 10,000 videos. In contrast to other datasets, trackers are restricted to use only the train split of the dataset in order to train networks, i.e. use of external training data is forbidden. Accordingly, we train our network using only the GOT10k train split. The success plots over all the 180 videos from the test split are shown in Fig. 4a. Among previous methods, the appearance model used by our tracker, namely DiMP-50, obtains the best results. Our approach, integrating scene information for tracking, significantly outperforms DiMP-50, setting a new state-of-the-art on this datatset. Our tracker achieves an AO (average overlap) score of 63.6, a relative improvement of 4.1% over the previous best method. These results clearly show the benefits of exploiting scene knowledge for tracking.

**TrackingNet [35]:** The large scale TrackingNet dataset consists of over 30,000 videos sampled from YouTube. We report results on the test split, consisting of 511 videos. The results in terms of normalized precision and success are shown in Table 3. The baseline approach DiMP-50 already achieves the best results with an AUC of 74.0. Our approach achieves a similar performance to the baseline, showing that it can generalize well to such real world videos.

**Table 2.** State-of-the-art comparison on the VOT2018 in terms of expected average overlap (EAO), accuracy and robustness. Our approach obtains the best EAO score, outperforming the previous best approach DiMP-50 with a EAO relative gain of 5.0%.

| | DRT [41] | RCO [28] | UPDT [4] | DaSiam-RPN [53] | MFT [28] | LADCF [48] | ATOM [8] | SiamRPN++ [30] | DiMP-50 [3] | **Ours** |
|---|---|---|---|---|---|---|---|---|---|---|
| EAO | 0.356 | 0.376 | 0.378 | 0.383 | 0.385 | 0.389 | 0.401 | 0.414 | **0.440** | **0.462** |
| Robustness | 0.201 | 0.155 | 0.184 | 0.276 | **0.140** | 0.159 | 0.204 | 0.234 | 0.153 | **0.143** |
| Accuracy | 0.519 | 0.507 | 0.536 | 0.586 | 0.505 | 0.503 | 0.590 | **0.600** | 0.597 | **0.609** |

**Table 3.** State-of-the-art comparison on the TrackingNet test set. Our approach performs similarly to previous best method DiMP-50, achieving an AUC score of 74.0%.

| | ECO [9] | SiamFC [2] | CFNet [43] | MDNet [36] | UPDT [4] | DaSiam-RPN [53] | ATOM [8] | SiamRPN++ [30] | DiMP-50 [3] | **Ours** |
|---|---|---|---|---|---|---|---|---|---|---|
| Norm. Prec. (%) | 61.8 | 66.6 | 65.4 | 70.5 | 70.2 | 73.3 | 77.1 | **80.0** | **80.1** | **80.0** |
| Success (AUC) (%) | 55.4 | 57.1 | 57.8 | 60.6 | 61.1 | 63.8 | 70.3 | **73.3** | **74.0** | **74.0** |

**Table 4.** State-of-the-art comparison on the LaSOT test set in terms of normalized precision and success. Our approach obtains competitive results with an AUC of 55.4%.

| | ECO [9] | DSiam [18] | StructSiam [52] | SiamFC [2] | VITAL [40] | MDNet [36] | SiamRPN++ [30] | ATOM [8] | DiMP-50 [3] | **Ours** |
|---|---|---|---|---|---|---|---|---|---|---|
| Norm. Prec. (%) | 33.8 | 40.5 | 41.8 | 42.0 | 45.3 | 46.0 | 56.9 | 57.6 | **65.0** | **63.3** |
| Success (AUC) (%) | 32.4 | 33.3 | 33.5 | 33.6 | 39.0 | 39.7 | 49.6 | 51.5 | **56.9** | **55.4** |

**OTB-100 [45]:** Figure 4b shows the success plots over all the 100 videos. Discriminative correlation filter based UPDT [4] tracker achieves the best results with an AUC score of 70.4. Our approach obtains results comparable with the state-of-the-art, while outperforming the baseline DiMP-50 by over 1% in AUC.
**NFS [14]:** The need for speed dataset consists of 100 challenging videos captured using a high frame rate (240 FPS) camera. We evaluate our approach on the downsampled 30 FPS version of this dataset. The success plots over all the 100 videos are shown in Fig. 4c. Among previous methods, our appearance model DiMP-50 obtains the best results. Our approach significantly outperforms DiMP-50 with a relative gain of 2.4%, achieving 63.4% AUC score.
**LaSOT [13]:** While our architecture is designed for short-term tracking, we evaluate on the long-term tracking dataset LaSOT for completeness. The results over 280 videos from the test split are shown in Tab. 4. DiMP-50 achieves the best results with an AUC score of 56.9, while our approach achieves an AUC score of 55.4. The decrease in performance compared to DiMP-50 is attributed to the fast update of the state vectors in our approach. While important for short-term tracking, the state updates can compromise the long-term re-detection capability. We believe that improving the long-term tracking performance of our approach is an interesting future work.

## 5 Conclusions

We propose a novel architecture which can exploit scene information for tracking. Our tracker represents scene information as dense localized state vectors. These state vectors are propagated through the sequence and combined with the appearance model output to localize the target. We evaluate our approach on 6 tracking benchmarks. Our tracker sets a new state-of-the-art on 3 benchmarks, demonstrating the benefits of exploiting scene information for tracking.
**Acknowledgments**: This work was supported by a Huawei Technologies Oy (Finland) project, the ETH Zürich Fund (OK), an Amazon AWS grant, and an Nvidia hardware grant.

## References


1. Ballas, N., Yao, L., Pal, C., Courville, A.C.: Delving deeper into convolutional networks for learning video representations. In: ICLR (2016)
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV workshop (2016)
3. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: ICCV (2019)
4. Bhat, G., Johnander, J., Danelljan, M., Khan, F.S., Felsberg, M.: Unveiling the power of deep tracking. In: ECCV (2018)
5. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR (2010)
6. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP (2014)
7. Danelljan, M., Bhat, G.: PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking (2019), accessed: 1/08/2019
8. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ATOM: Accurate tracking by overlap maximization. In: CVPR (2019)
9. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: efficient convolution operators for tracking. In: CVPR (2017)
10. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV (2015)
11. Danelljan, M., Robinson, A., Shahbaz Khan, F., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV (2016)
12. Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: ICCV (2015)
13. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. CoRR **abs/1809.07845** (2018), http://arxiv.org/abs/1809.07845
14. Galoogahi, H.K., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: ICCV (2017)
15. Gan, Q., Guo, Q., Zhang, Z., Cho, K.: First step toward model-free, anonymous object tracking with recurrent neural networks. ArXiv **abs/1511.06425** (2015)
16. Gao, J., Zhang, T., Xu, C.: Graph convolutional tracking. In: CVPR (2019)
17. Gladh, S., Danelljan, M., Khan, F.S., Felsberg, M.: Deep motion features for visual tracking. 2016 23rd International Conference on Pattern Recognition (ICPR) pp. 1243–1248 (2016)
18. Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., Wang, S.: Learning dynamic siamese network for visual object tracking. In: ICCV (2017)
19. He, A., Luo, C., Tian, X., Zeng, W.: Towards a better match in siamese network based visual object tracker. In: ECCV workshop (2018)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
21. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: ECCV (2016)

22. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI **37**(3), 583–596 (2015)
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**, 1735–1780 (1997)
24. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. arXiv preprint arXiv:1810.11981 (2018)
25. Kenan, D., Dong, W., Huchuan, L., Chong, S., Jianhua, L.: Visual tracking via adaptive spatially-regularized correlation filters. In: CVPR (2019)
26. Kiani Galoogahi, H., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: ICCV (2017)
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2014)
28. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pfugfelder, R., Zajc, L.C., Vojir, T., Bhat, G., Lukezic, A., Eldesokey, A., Fernandez, G., et al.: The sixth visual object tracking vot2018 challenge results. In: ECCV workshop (2018)
29. Lee, H., hankyeol: Bilinear siamese networks with background suppression for visual object tracking. In: BMVC (2019)
30. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: CVPR (2019)
31. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: CVPR (2018)
32. Li, F., Tian, C., Zuo, W., Zhang, L., Yang, M.: Learning spatial-temporal regularized correlation filters for visual tracking. In: CVPR (2018)
33. Li, X., Ma, C., Wu, B., He, Z., Yang, M.H.: Target-aware deep tracking. In: CVPR (2019)
34. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV (2015)
35. Müller, M., Bibi, A., Giancola, S., Al-Subaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: ECCV (2018)
36. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR (2016)
37. Ning, G., Zhang, Z., Huang, C., He, Z., Ren, X., Wang, H.: Spatially supervised recurrent convolutional neural networks for visual object tracking. 2017 IEEE International Symposium on Circuits and Systems (ISCAS) pp. 1–4 (2016)
38. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
39. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R., Yang, M.H.: CREST: Convolutional residual learning for visual tracking. In: ICCV (2017)
40. Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., Shen, C., Lau, R.W.H., Yang, M.H.: Vital: Visual tracking via adversarial learning. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8990–8999 (2018)
41. Sun, C., Wang, D., Lu, H., Yang, M.: Correlation tracking via joint discrimination and reliability learning. In: CVPR (2018)
42. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: CVPR (2017)
43. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: CVPR (2017)
44. Wang, Q., Teng, Z., Xing, J., Gao, J., Hu, W., Maybank, S.J.: Learning attentions: Residual attentional siamese network for high performance online visual tracking. In: CVPR (2018)

45. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. TPAMI **37**(9), 1834–1848 (2015)
46. Xiao, J., Qiao, L., Stolkin, R., Leonardis, A.: Distractor-supported single target tracking in extremely cluttered scenes. In: ECCV (2016)
47. Xu, J., Ranftl, R., Koltun, V.: Accurate Optical Flow via Direct Cost Volume Processing. In: CVPR (2017)
48. Xu, T., Feng, Z., Wu, X., Kittler, J.: Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual tracking. CoRR **abs/1807.11348** (2018), http://arxiv.org/abs/1807.11348
49. Yang, T., Chan, A.B.: Recurrent filter learning for visual tracking. 2017 IEEE International Conference on Computer Vision Workshops (ICCVW) pp. 2010–2019 (2017)
50. Yang, T., Chan, A.B.: Learning dynamic memory networks for object tracking. In: ECCV (2018)
51. Zhang, L., Gonzalez-Garcia, A., Weijer, J.v.d., Danelljan, M., Khan, F.S.: Learning the model update for siamese trackers. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
52. Zhang, Y., Wang, L., Qi, J., Wang, D., Feng, M., Lu, H.: Structured siamese network for real-time visual tracking. In: ECCV (2018)
53. Zhu, Z., Wang, Q., Bo, L., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: ECCV (2018)
54. Zhu, Z., Wu, W., Zou, W., Yan, J.: End-to-end flow correlation tracking with spatial-temporal attention. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018 (2018)