STMTrack: Template-free Visual Tracking with Space-time Memory Networks

Zhihong Fu, Qingjie Liu[*], Zehua Fu, Yunhong Wang
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
Hangzhou Innovation Institute, Beihang University
{fuzhihong, qingjie.liu, yhwang}@buaa.edu.cn, zehua_fu@163.com

Abstract

Boosting performance of the offline trained siamese trackers is getting harder nowadays since the fixed information of the template cropped from the first frame has been almost thoroughly mined, but they are poorly capable of resisting target appearance changes. Existing trackers with template updating mechanisms rely on time-consuming numerical optimization and complex hand-designed strategies to achieve competitive performance, hindering them from real-time tracking and practical applications. In this paper, we propose a novel tracking framework built on top of a space-time memory network that is competent to make full use of historical information related to the target for better adapting to appearance variations during tracking. Specifically, a novel memory mechanism is introduced, which stores the historical information of the target to guide the tracker to focus on the most informative regions in the current frame. Furthermore, the pixel-level similarity computation of the memory network enables our tracker to generate much more accurate bounding boxes of the target. Extensive experiments and comparisons with many competitive trackers on challenging large-scale benchmarks, OTB-2015, TrackingNet, GOT-10k, LaSOT, UAV123, and VOT2018, show that, without bells and whistles, our tracker outperforms all previous state-of-the-art real-time methods while running at 37 FPS. The code is available at https://github.com/fzh0917/STMTrack.

1. Introduction

Visual object tracking is an essential task in computer vision with applications in various fields such as human-computer interactions [29], video surveillance [54], and autonomous driving [22]. Significant efforts have been devoted to address this problem, yet there is still a great gap to the practical applications due to the challenging factors such as occlusions, fast motions, and non-rigid deformations [13, 53, 20], which urge us to develop trackers with strong adaptiveness and robustness.

[*]Corresponding author.

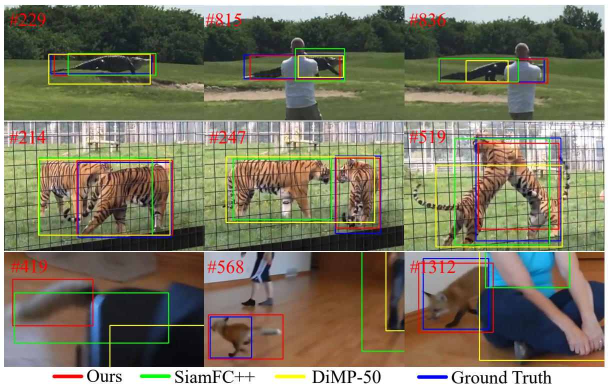


Figure 1: Visualized comparisons of our method with representative trackers SiamFC++ [56] and DiMP-50 [2]. Our method can estimate more accurate target state when targets suffer from partial occlusions and non-rigid deformations.

The goal of visual tracking is to locate an object in the subsequent frames of a video given its initial annotation in the first frame. In recent years, with the advancements of deep learning techniques, deep trackers have dominated the tracking field, among which two methodologies are widely studied, and one popular methodology addresses object tracking as a similarity matching problem between the target template and the search frames in an embedding space offline trained. The representative template-matching methods are siamese trackers [1, 65, 61, 14, 50, 6, 24, 23, 56, 15]. These methods usually do not update the template and thus are hard to adapt to appearance changes caused by occlusions, non-rigid deformations, etc.

To solve this problem, some trackers [2, 11] are equipped with sophisticated template updating mechanisms and thus show stronger robustness than siamese trackers. However, online template updating requires much more computational resources, which impends trackers from real-time tracking. Furthermore, these customized updating strategies [16, 58, 65, 25, 59, 7] introduce hyper-parameters that require tricky tuning.

Note that when tracking moving objects humans remember their identities in visual working memory to maintain temporal continuity in a constantly changing environment [33]. This inspires us to develop a memory-based tracking model to take advantage of rich historical information of the object. In contrast to previous works that strive to design template updating mechanisms to capture appearance variations of the object, our model predicts the state of the object from its historical information stored in the memory network, thus avoiding of using the template and updating it. Thus we call our model a template-free method. Furthermore, our tracker computes pixel-level similarities to locate the target, making it more robust to partial occlusions and non-rigid deformations than those using feature-map-level cross correlation. Fig. 1 depicts this advantage of our tracker (Refer to the section 1.1 of the supplemental material for quantitative comparisons).

The proposed method is evaluated on six benchmarks: OTB-2015, TrackingNet, LaSOT, GOT-10k, UAV123, and VOT2018 and surpasses all state-of-the-art real-time approaches while running at 37 FPS. Notably, it achieves 80.3 success (AUC) on the challenging TrackingNet dataset, outperforming the previous best real-time method by 4.5%. It also sets new state-of-the-art performance on the performance-saturated OTB-2015 dataset.

Summarily, the main contributions of this work are fourfold.

- We propose a novel end-to-end memory-based tracking framework, which not only is as simple and efficient as the offline trained siamese networks, but also has strong adaptiveness ability as the sophisticated template updating strategies.
- The proposed tracking framework deviates from the original evolving path of template-based tracking, and it could inspire more space-time-memory-based trackers to be developed in the future.
- A novel memory mechanism based on pixel-level similarity computation is introduced for visual tracking, which enables our tracker to have stronger robustness and to generate much more accurate target bounding boxes than many previous high-performance methods that use feature-map-level cross correlation.
- Our proposed tracker outperforms all state-of-the-art real-time approaches on OTB-2015, TrackingNet, LaSOT, and GOT-10k, while running in real-time at 37 FPS, which demonstrates the superiority of the framework.

2. Related Work

2.1. Siamese Trackers

Siamese networks [5, 60, 8] have attracted significant attention in recent years and been very popular in the tracking community [1, 65, 61, 14, 50, 6, 24, 23, 56, 15] . Siamese trackers treat the visual object tracking task as a matching problem. During inference, a template is cropped from the first frame and matched to the search regions in the current frame to achieve tracking. They have excellent performance and real-time tracking speed when running in many routine tracking scenarios. However, their vulnerability becomes evident when the targets suffer from drastic appearance changes, non-rigid deformations, and partial occlusions. Unlike these approaches, our proposed work makes full use of historical multiple-frame information in the tracking process, which can greatly improve the robustness of the model in those challenging scenarios.

2.2. Template Updates

To cope with challenging factors in tracking processing, updating the template is critical to adapt the tracker to target variations.

DSiam [16] proposes a dynamic siamese network with a fast transformation learning model to enable effective template updating and cluttered background suppression. In [65], a distractor-aware module is designed to transfer the general embedding in siamese networks to the specific target domain of the current video. Observing that the absolute values of gradients at locations where objects are occluded or distracted from similar objects are prone to be higher, GradNet [25] integrates backward gradients into the initial template for augmenting the discriminative ability of the template. These methods explicitly select informative frames and use customized strategies to update the template. Different from these explicit template updating strategies, we propose to store the historical information of the target in memory networks and retrieve them as needed.

2.3. Memory Networks

Memory networks were first proposed to solve document Q&A [52, 40, 34, 21] in the Natural Language Processing field. They are common neural networks equipped with external memory components to read and write historical information. Recently, memory networks have shown significant performance improvement in few shot learning [39, 30], video object segmentation [37, 30], *etc.*

Memory networks have been also introduced into visual tracking, a typical one is MemTrack [58]. It uses a memory network to read a residual template during tracking and then combines it with the initial template to yield a synthetic template that is used as an updated representation of the target. Although MemTrack [58] uses a large amount of historical information in the tracking process, the memory reading operation controlled by a LSTM may lose useful information. The overall performance of the tracker is greatly affected by the learning quality of the LSTM controller.

In our work, the retrieval of historical information is de-

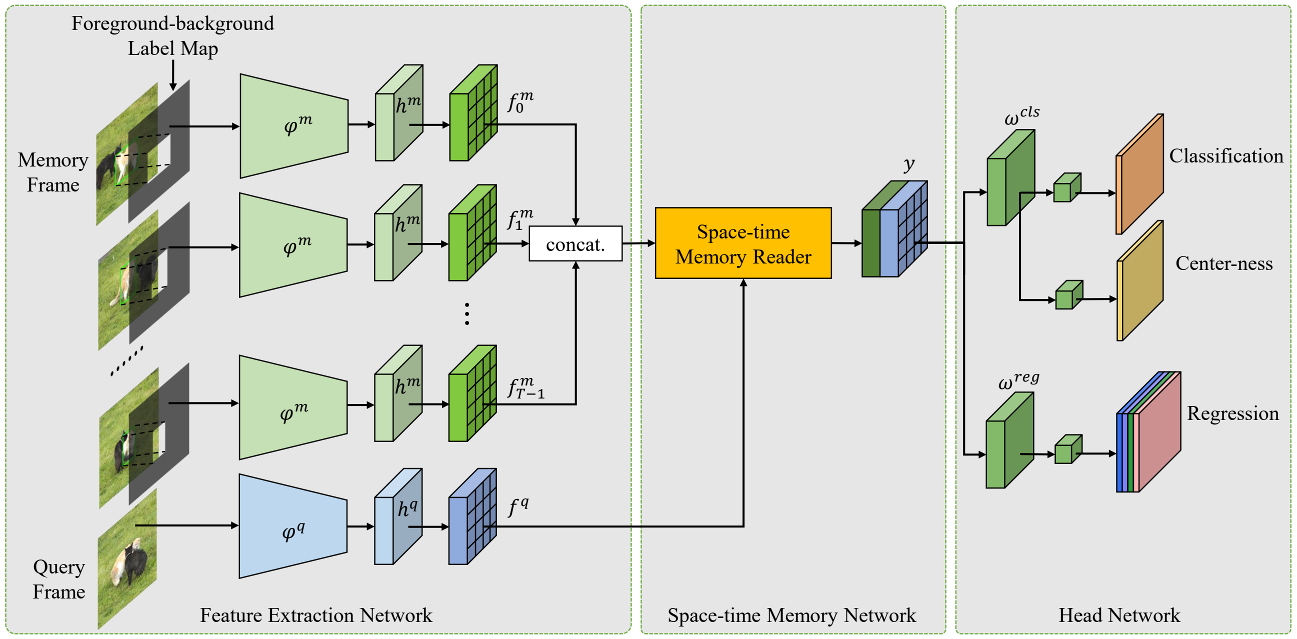


Figure 2: The architecture of our proposed method. The left part is the feature extraction network that consists of a memory branch (displayed in light green) and a query branch (displayed in light blue). The memory branch takes both memory frames and corresponding foreground-background label maps as inputs. "concat." denotes the concatenation operation along the temporal dimension. The middle part is the space-time memory network that retrieves the target information from multiple memory frames for the target localization in the query frame. The right side is the head network for the foreground-background classification and the target bounding box regression of the query frame.

termined by the current frame itself, therefore it can obtain all it needs, adaptively.

3. Proposed Method

In this section, we will describe the proposed tracking framework in detail. First, we will introduce an overview of the framework in Sec. 3.1. Then, we will give an account of each module of the whole framework one by one from Sec. 3.2 to Sec. 3.4. Last, we will present the online tracking process of the framework in Sec. 3.5.

3.1. Architecture

As shown in Fig. 2, the framework can be divided into three parts, a feature extraction network, a space-time memory network, and a head network. The feature extraction network consists of a memory branch (in light green) and a query branch (in light blue). The memory branch takes both memory frames and corresponding foreground-background label maps (will be explained in the next section) as inputs, while the input of the query branch is only a single query frame. In this work, the memory frames are multiple historical frames, and the query frame is the current frame in a tracking sequence. After the feature extraction, the space-time memory network retrieves information related to the target from features of all memory frames, generating a synthetic feature map to classify the target from backgrounds and to predict the target bounding box for the query frame.

3.2. Feature Extraction Network

Here we describe the feature extraction procedures of the memory branch and the query branch, respectively.

Memory Feature Extraction. The inputs of the memory branch are T memory frames m (each frame is m_i) and T foreground-background label maps c (each label map is c_i), where c is to ensure that the memory backbone φ^m learns the consistency of the real target characteristics rather than distractors and cluttered background information. Specifically, we label each pixel with 1 within the corresponding ground truth bounding box and 0 elsewhere for each memory frame m_i. Then, we adopt the first convolutional layer of φ^m (represented by φ_0^m) and an extra convolutional layer g to map m and c to a same embedding space, respectively. After that, we add $\varphi_0^m(m)$ and $g(c)$ element-wise, then input the sum to the latter layers of φ^m to generate T memory feature maps (denoted as f^m, each memory feature map is f_i^m). The feature dimensionality of f^m is then reduced to 512 by a non-linear convolutional layer (denoted as h^m):

$$f_i^m = h^m(\varphi_\gamma^m(\varphi_0^m(m_i) \oplus g(c_i))) \tag{1}$$

where $f_i^m \in \mathbb{R}^{C\times H\times W}$, φ_γ^m represents all layers of φ^m except the first layer, and "$\oplus$" is element-wise addition.

Query Feature Extraction. Different from the memory branch, the query branch takes a query frame q as input and produces a feature map $\varphi^q(q)$. Similar to the memory branch, the feature dimensionality of $\varphi^q(q)$ is also reduced to 512 by a non-linear convolutional layer (denoted as h^q):

$$f^q = h^q(\varphi^q(q)) \tag{2}$$

where $f^q \in \mathbb{R}^{C\times H\times W}$.

Note that the two backbones φ^m and φ^q share the same network architecture but have different parameters. An ablation study on whether sharing one backbone can be seen in Sec. 4.3.

3.3. Space-time Memory Network

As illustrated in Fig. 3, we first compute the similarities between every pixel of f^m and every pixel of f^q to obtain a similarity matrix $w \in \mathbb{R}^{THW\times HW}$. Inspired by [51], we expect the similarity computation to apply the gaussian function. Thus, we normalize w with a `softmax` function. Taking one element w_{ij} for example, we can formally denote w_{ij} as:

$$w_{ij} = \frac{\exp\left[\left(f^m_{i\cdot} \odot f^q_{\cdot j}\right)/s\right]}{\sum\limits_{\forall k} \exp\left[\left(f^m_{k\cdot} \odot f^q_{\cdot j}\right)/s\right]} \tag{3}$$

where i is the index of each pixel on $f^m \in \mathbb{R}^{THW\times C}$, j is the index of each pixel on $f^q \in \mathbb{R}^{C\times HW}$, and the binary operator $\odot$ denotes vector dot-product. Here s is a scaling factor to prevent the `exp` function from overflowing numerically. Following [45], we set s to $\sqrt{C}$, where C is the feature dimensionality of f^m.

Then, treating w as a soft weight map, we multiply f^m by w. Because f^m stores all historical memory information related to the target, according to the needs of the query frame itself, the target information stored in f^m is adaptively retrieved. Obviously, the the readout information is a feature map as the same size as f^q. Therefore, we concatenate the readout information and the query feature map f^q along the channel dimension to generate the final synthetic feature map y. Formally, for the i-th element of y, the space-time memory read operation can be denoted as:

$$y_i = \text{concat}\left(f^q_i, (f^m)^T_i \otimes w\right) \tag{4}$$

where $(f^m)^T \in \mathbb{R}^{C\times THW}$ is the transpose of f^m, and the `concat`$(\cdot,\cdot)$ function represents the concatenation operation.

At a first glance, the working mechanism of the memory read operation is similar to the non-local self-attention [51]. A representative example of deploying the non-local self-attention [51] in visual tracking is AlphaRefine [57], the winner of the real-time tracking challenge VOT-RT2020 [19], that uses a non-local block to augment the response map generated by a pixel-wise correlation since the longer-range dependencies can produce more precise target boundary decision information. Differently, the purpose of designing the space-time memory reader in our proposed framework, however, is to retrieve the target information from multiple memory frames by taking the similarity matrix as soft weights, instead of computing the non-local self-attention for each pixel pair in a feature map.

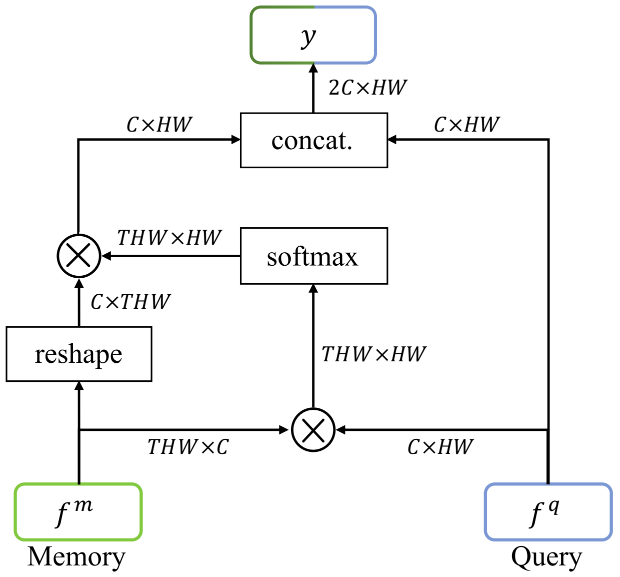


Figure 3: The space-time memory reader. Here $f^m \in \mathbb{R}^{T\times C\times H\times W}$ and $f^q \in \mathbb{R}^{C\times H\times W}$, where T is the number of memory frames, C, H and W represent the feature dimensionality, the height, and the width of the feature map, respectively. For the convenience of matrix multiplication in math, we reshape f^m from $T\times C\times H\times W$ to $THW\times C$, and reshape f^q from $C\times H\times W$ to $C\times HW$, thus here $THW = T\times H\times W$ and $HW = H\times W$. the operator "$\otimes$" denotes matrix multiplication, and "concat." denotes the concatenation operation along the channel dimension.

Particularly, different from STMVOS [34] and Gragh-MemVOS [37] in video object segmentation, our method does not divide the features extracted by φ^m and φ^q into keys and values, but directly uses f^m and f^q to locate the target. The motivation is that, f^m itself happens to provide adequate target information to find the exposed parts of the target when it suffers from partial occlusions in the query frame. This difference makes the space-time memory network more suitable for the single object tracking task.

3.4. Head Network

Inspired by the phenomenon that the one-stage anchor-free detector [44] has achieved better performance and has fewer parameters than the one-stage anchor-based method [27] in object detection, we design an anchor-free head network that contains a classification branch to classify the target from backgrounds and an anchor-free regression branch to directly estimate the target bounding box.

To be specific, first, we encode y with a lightweight classification convolutional network ω^{cls} to integrate f^q and the retrieved information from f^m to adapt to the classification task. Then, a linear convolutional layer with 1×1 kernel is used to reduce the dimensionality of the output of ω^{cls} to 1, producing the final classification response map $R^{cls} \in \mathbb{R}^{1 \times H \times W}$.

Moreover, we observe that the positive samples near the target boundary tend to predict low-quality target bounding boxes. Therefore, a sub-branch is forked after ω^{cls} to generate a center-ness response map $R^{ctr} \in \mathbb{R}^{1 \times H \times W}$, as illustrated in the right part of Fig. 2. During inference, R^{cls} is multiplied by R^{ctr} to suppress the classification confidence scores of pixels away from the target center.

In the regression branch, we pass y to another lightweight regression convolutional network ω^{reg} and then reduce the dimensionality of the outputted features to 4 to generate a regression response map $R^{reg} \in \mathbb{R}^{4 \times H \times W}$ for the target bounding box estimation.

We recommend readers to refer to [56] for more details about the encodings and the training objectives of R^{cls}, R^{ctr}, and R^{reg}.

3.5. Inference Phase

Our space-time memory network is flexible so that the number of used memory frames (*i.e.* the memory size) during inference is independent of the number of memory frames during training (See Sec. 4.3 for the impact of different number of memory frames on performance in the two phases). In this work, for the current frame F_t, we select N memory frames from all historical frames (*i.e.* frame F_1 to frame F_{t-1}) as memory frames for rich appearance information and strong generalization ability. From the perspective of existing works [46, 37], experiences, and intuitions, target information from the first frame and the previous frame plays an important role for the target localization in the current frame. Specifically, the target from the first frame provides the most reliable information, while the tracked target from the previous frame has the most similar appearance to the target in the current frame. Therefore, for the current frame F_t, the memory frames hold the first frame F_1, the previous frame F_{t-1} and other $N-2$ frames $F_{\tau_1}, F_{\tau_2}, \cdots, F_{\tau_{(N-2)}}$ sampled following the methodology: splitting all historical frames into $N-2$ segments, and choosing one representative frame from each segment for the best balance between the target domain adaptation, underfitting, and the time cost. Formally, the sampling method can de described as:

$$\tau_i = \left\lfloor \left\lfloor \frac{t-1}{N-2} \right\rfloor \times (i + \Delta_i) \right\rfloor \tag{5}$$

Here, $i \in \{1, 2, \cdots, N-2\}$, and $\Delta_i \in [0, 1)$ is the offset of the representative frame in the i-th segment. For the first N frames, we set all historical frames (*i.e.* $F_1, F_2, \cdots, F_{N-1}$) as memory frames. In our experiments, N is set to 6, and we simply set $\left\{\Delta_i = \frac{1}{2} | 1 \le i \le N-2\right\}$.

For each frame in the whole tracking process, after obtaining R^{cls}, R^{ctr}, and R^{reg}, the postprocessing is the same as [56].

4. Experiments

Our tracker is implemented in Python using *PyTorch* framework, which runs at 37[1] FPS on an NVIDIA Tesla V100 GPU. We evaluate our tracker on benchmarks: OTB-2015 [53], TrackingNet [36], GOT-10k [17], LaSOT [13], UAV123 [35], and VOT2018 [20].

4.1. Training Dataset

We adopt TrackingNet [36], the *training* set of LaSOT [13] and GOT-10k [17], ILSVRC VID [38], ILSVRC DET [38], and COCO [28] as our training dataset except the GOT-10k benchmark. We sample T ($T = 3$ in this paper) frames within the maximum frame index gap 100 and adopt random affine transformations to increase data diversity for training video sequences. To be specific, the translation is randomly performed from $-0.2S$ to $0.2S$ and the resizing scale varies between $\frac{1}{1+r}$ and $1+r$ with $r = 0.3$. Here S is the cropping size, and we set S to the scale of 4 times the target bounding box for mining as much contextual information as possible and enhancing the discriminative ability of the model. We apply the above data augmentation strategy with different parameter settings to generate a synthetic video sample for ILSVRC DET [38] and COCO [28]. Note that, different from siamese trackers, our method does not have to keep the same target scale for a training sample. For each frame in a training sample, taking the target center as the center, we crop a square image patch with side length S from the original frame and resize the cropped image patch to 289×289 to be the input of the model.

4.2. Implementation Details

Model Settings. We adopt GoogLeNet [43] as our backbone φ^m and φ^q. The classification convolutional network ω^{cls} and the regression convolutional network ω^{reg} are both comprised of seven convolutional layers. Each convolutional layer in ω^{cls} and ω^{reg} is followed by a `ReLU` activation function.

Optimization and Training Strategies. Our tracker is trained with SGD optimizer for 20 epochs. There are 300,000 samples per epoch, and the mini-batch size is set to 64. The whole training phase takes about 27 hours to converge with four NVIDIA Tesla V100 GPUs. For the GOT-

[1]The running speed is calculated by the GOT-10k evaluation server.

10k benchmark, we set the number of samples per epoch to 150,000 and the mini-batch size to 32. The momentum and the weight decay rate are set to 0.9 and 1×10^{-4}, respectively. We freeze all convolutional layers of φ^m and φ^q in the first 10 epochs and unfreeze all convolutional layers of stage 3 and 4 [43] of φ^m and φ^q for the other epochs. The learning rate increases from 1×10^{-2} to 8×10^{-2} with a warmup technology at the first epoch and decreases from 8×10^{-2} to 1×10^{-6} with a cosine annealing learning rate schedule for the other epochs.

4.3. Ablation Study

In this section, we conduct a comprehensive ablation study for the proposed tracker.

Should the Tracker Share one Backbone? First, we analyze whether the tracker should share one backbone between the memory branch and the query branch like most siamese networks. As shown in Tab. 1, from the top two rows, we observe that sharing one backbone can improve the average overlap (AO) by 2.3% if we do not use foreground-background label maps. However, the comparison of the bottom two rows in Tab. 1 and comparisons on other benchmarks in Tab. 2 show that the performance of using different backbones is more superior than sharing one.

Should the Tracker Use Foreground-background Label Maps in the Input of the Memory Branch? Second, we discuss the necessity to use foreground-background label maps in the input of the memory branch. By comparing the first row with the third row and comparing the second row with the last row in Tab. 1, we notice that the performance will be improved by 2.9% and 7.4% with the same backbone and different backbones, respectively. It indicates that foreground-background label maps are crucial to the memory mechanism in this framework.

What is the Best Number of Reference Frames in the Memory? Last but not least, the number of reference frames in the memory (*i.e.* the memory size) is a key factor for memory networks. During training, the number of reference frames affects the learning qualities of two backbones. The more reference frames, the more target patterns can be trained, but the more frames are similar to the current frame. In that case, the network tends to compare the most similar image pairs, rather than learning to compute the similarities between the current frame and frames with clutter backgrounds or partially occluded targets. We verify the impact in training with different memory size settings on the GOT-10k benchmark. Tab. 3 shows that using 3 reference frames in a training sample brings the best performance in terms of average overlap (AO) metric.

During inference, the memory size not only affects the performance, but also significantly determines the running speed. We conduct a group of experiments on TrackingNet to analyze the impact of the memory size on the performance and the speed. As listed in Tab. 4, the most suitable memory size is 6 for this work. Besides, the more reference frames does not produce the better performance. We speculate that there are two main reasons: the one is overfitting, and the another one is too many low-quality memory frames affect tracking results that further products more inferior memory frames.

Table 1: Ablation study on the GOT-10k benchmark. Here the variable "share" denotes whether the network should share the backbone between the memory and the query branch, and the variable "fb_label" represents whether the network should use foreground-background label maps in the memory branch input.

share	fb_label	**AO**	**$\text{SR}_{0.5}$**	**$\text{SR}_{0.75}$**
✓	-	0.591	0.662	0.507
-	-	0.568	0.638	0.480
✓	✓	0.620	0.713	0.538
-	✓	**0.642**	**0.737**	**0.579**

Table 2: The same ablation study as Tab. 1 on OTB-2015, TrackingNet, and LaSOT. The tracker is evaluated by success (AUC) metric.

share	fb_label	**OTB-2015**	**TrackingNet**	**LaSOT**
✓	✓	0.702	79.7	0.593
-	✓	**0.719**	**80.3**	**0.606**

Table 3: The performance on GOT-10k with different number of reference frames in training. Here **AO** is the average overlap metric.

#	**1**	**2**	**3**	**4**
AO	0.629	0.624	**0.642**	0.627

Table 4: The performance in terms of success (AUC) metric on TrackingNet with different number of reference frames in the inference phase.

#	**1**	**2**	**4**	**6**	**8**	**ALL**
Success	79.1	79.3	80.2	**80.3**	80.2	79.8
FPS	43.0	26.6	29.3	28.6	22.7	6.5

4.4. Comparison with the state-of-the-art

OTB-2015. OTB-2015 [53] is a classical benchmark in visual object tracking, containing 100 short-term videos with 590 frames per video on average. We report the results on OTB-2015. It is known to have tended to saturation over recent years. Still, as shown in Tab. 5, our approach surpasses the previous best performance trackers by 0.4% in terms of success (AUC) metric, setting a new state-of-the-art performance on this dataset.

TrackingNet. TrackingNet [36] is a large-scale short-

Table 5: A success (AUC) performance list on OTB-2015 for a comprehensive comparison of our tracker with competitive trackers published in recent years. The best three results are highlighted in **red**, **blue**, and **green**, respectively. Trackers are ranked from top to bottom and left to right according the **Success** values.

Tracker	**Success**	**Tracker**	**Success**
Ours	**0.719**	SiamRPN++ [23]	0.696
DROL [64]	**0.715**	KYS [3]	0.695
RPT [32]	**0.715**	MCCT [49]	0.695
CGACD [12]	**0.713**	GFS-DCF [55]	0.693
SiamAttn [59]	0.712	ASRCF [9]	0.692
DCFST [63]	0.709	PGNet [26]	0.691
UPDT [4]	0.702	RPCF [42]	0.690
DRT [41]	0.699	SPM [48]	0.687
SiamCAR [15]	0.697	DiMP-50 [2]	0.684
PrDiMP-50 [11]	0.696	Ocean [62]	0.684
SiamBAN [6]	0.696	SiamFC++ [56]	0.683

Table 7: A performance comparison of our tracker with other competitive approaches on the test split of GOT-10k in terms of average overlap (AO) and success rates (SR) at threshold 0.5 and 0.75. The best three results are highlighted in **red**, **blue**, and **green**, respectively. Trackers are ranked from top to bottom according the **AO** values.

Tracker	**AO**	**SR$_{0.5}$**	**SR$_{0.75}$**	**FPS**
Ours	**0.642**	**0.737**	**0.575**	37
KYS [3]	**0.636**	**0.751**	**0.515**	20
PrDiMP-50 [11]	**0.634**	**0.738**	**0.543**	30
RPT [32]	0.624	0.730	0.504	20
Ocean [62]	0.611	0.721	-	58
DiMP-50 [2]	0.611	0.717	0.492	43
D3S [31]	0.597	0.676	0.462	25
SiamFC++ [56]	0.595	0.695	0.479	90
SiamCAR [15]	0.569	0.670	0.415	52
ATOM [10]	0.556	0.634	0.402	30
SiamRPN++ [23]	0.517	0.616	0.325	35

Table 6: A performance comparison of our tracker with other competitive approaches on the test split of TrackingNet. Trackers are ranked from top to bottom according the "**Suc.**" values. "**Suc.**", "**Prec.**", and "**Norm. Prec.**" are abbreviations for success (AUC), precision, and normalized precision, respectively. The best three results are highlighted in **red**, **blue**, and **green**, respectively.

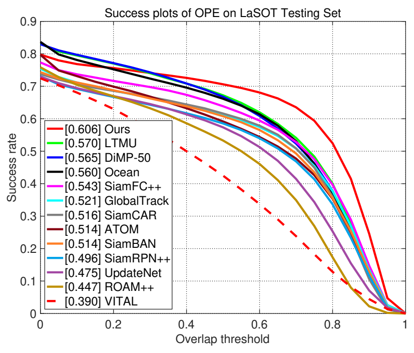
Tracker	**Suc.**	**Prec.**	**Norm. Prec.**
Ours	**80.3**	**76.7**	**85.1**
PrDiMP-50 [11]	**75.8**	70.4	81.6
FCOS-MAML [47]	**75.7**	-	**82.2**
SiamFC++ [56]	75.4	**70.5**	80.0
SiamAttn [59]	75.2	-	**81.7**
DCFST-50 [63]	75.2	70.0	80.9
DROL [64]	74.6	**70.8**	**81.7**
KYS [3]	74.0	68.8	80.0
DiMP-50 [2]	74.0	68.7	80.1
SiamRPN++ [23]	73.3	69.4	80.0
D3S [31]	72.8	66.4	76.8
CGACD [12]	71.1	69.3	80.0
GlobalTrack [18]	70.4	65.6	75.4
ATOM [10]	70.3	64.8	77.1

term tracking dataset that provides a large amount of videos in the wild for training and testing. The *testing* set contains 511 videos without publicly released ground truths. We evaluate our tracker on the *testing* set and obtain results from the dedicated evaluation server. As shown in Tab. 6, our tracker outperforms all previous state-of-the-art real-time approaches by a large margin and strongly sets leading performance. It is noteworthy that this dataset has a wide variety in terms of classes and scenarios in the wild; therefore, the significant performance improvement of our approach illustrates its strong generalization ability to real-world tracking videos.

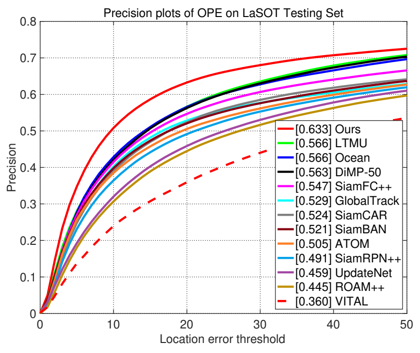
GOT-10k. GOT-10k [17] is a recently released large-scale generic object tracking benchmark, containing 10,000 videos totally, in which the *testing* set has 180 videos. Similar to TrackingNet, the ground truths of the *testing* set are also withheld so that all tracking results must be evaluated in a specific evaluation server. Different from others, GOT-10k benchmark restricts trackers to use only the *training* set for training. In this work, we follow this protocol for training our tracker and testing it on the *testing* set. All settings are unchanged except for the training data. Tab. 7 lists a comparison of our tracker with other state-of-the-art trackers in terms of average overlap (AO) and success rates (SR) at threshold 0.5 and 0.75. Benefit from the pixel-level similarity computation in the space-time memory reader, our method outperforms the second place tracker PrDiMP-50 [11] by 3.2% for the SR$_{0.75}$ metric (The percentage of successfully tracked frames where the overlaps exceed 0.75).

LaSOT. LaSOT [13] is also a large-scale single object tracking dataset with high-quality annotations. Its *testing* set consists of 280 long videos, with an average of 2500 frames per video. Thus, the robustness of trackers is crucial against complicated scenarios, such as occlusions, out-of-view, *etc*. We report the results on the *testing* set. As shown in Fig. 4, comparing with twelve comparable trackers, our tracker sets top performance in terms of success, precision, and normalized precision.

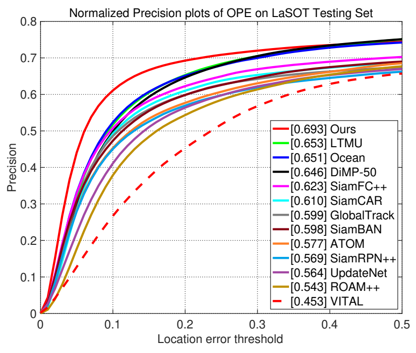
UAV123. UAV123 [35] is designed to evaluate trackers in UAV applications, including 123 low altitude aerial videos, with an average of 915 frames per video. Due to the



(a) Success Plot



(b) Precision Plot



(c) Normalized Precision Plot

Figure 4: Plots show comparisons of our tracker with other competitive trackers on the *testing* set of LaSOT. Trackers are evaluated by the success, precision, and normalized precision metrics.

Table 8: A comparison of our tracker with other competitive approaches on UAV123 in terms of success (AUC) metric. The best three results are highlighted in **red**, **blue**, and **green**, respectively.

	Ours	**DiMP-50** [2]	**ATOM** [10]	**SiamBAN** [6]	**SiamCAR** [15]	**SiamRPN++** [23]	**UPDT** [4]
Success	**0.647**	**0.654**	**0.643**	0.631	0.614	0.613	0.545

Table 9: A comparison of our tracker with state-of-the-art trackers on VOT2018. The best results are highlighted in **red**, **blue**, and **green**, respectively. Trackers are ranked from top to bottom according the **EAO** scores. The arrows after the metrics mean that the bigger(↑) or the smaller(↓) is the better.

Tracker	**EAO**↑	**A**↑	**R**↓
Ours	0.447	0.590	0.159
D3S [31]	**0.489**	**0.640**	**0.150**
Ocean [62]	**0.489**	0.592	**0.117**
SiamAttn [59]	**0.470**	**0.630**	0.160
KYS [3]	**0.462**	0.609	**0.143**
SiamBAN [6]	0.452	0.597	0.178
PGNet [26]	0.447	0.618	0.192
PrDiMP-50 [11]	0.442	0.618	0.165
DiMP-50 [2]	0.440	0.597	0.153
Siam R-CNN [46]	0.408	0.609	0.220
SiamFC++ [56]	0.426	0.587	0.183
SiamRPN++ [23]	0.414	0.600	0.234
FCOS-MAML [47]	0.392	**0.635**	0.220

characteristics of UAV, this dataset has numerous scenarios with partial and full occlusions, out-of-view, and small objects. Thus, many objects have quite low resolutions. As shown in Tab. 8, however, our tracker obtains a success (AUC) score of 0.647, which still significantly outperforms recent competitive siamese trackers SiamBAN [6], SiamCAR [15], and SiamRPN++ [23], while running at a real-time speed.

VOT2018. The 2018 version of the visual object tracking (VOT) challenge [20] contains 60 videos. Following the evaluation protocol of the VOT2018 dataset, we report the results of our tracker in terms of expected average overlap (EAO), accuracy (A), and robustness (R) and compare it with state-of-the-art trackers. As shown in Tab. 9, the robustness of our tracker is similar to D3S [31] and SiamAttn [59]. However, the accuracy is worse than them since the ground truths in the VOT evaluation system are rotated bounding boxes, the estimated bounding boxes in our tracker are axis-aligned instead.

5. Conclusions

This work proposes a novel tracking framework based on space-time memory networks. The framework abandons the traditional template-based tracking mechanism, using multiple memory frames and foreground-background label maps to locate the target in the query frame. In the space-time memory networks, the target information stored in multiple memory frames is adaptively retrieved by the query frame, so that the tracker has a strong adaptive ability to the target variations. Extensive experiments demonstrates that, without bells and whistles, the proposed tracker achieves better performance than current state-of-the-art real-time methods, while running at 37 FPS. The experiments also shows its generalizability, extendibility, and applicability.

Acknowledgement. The work was supported by the National Key Research and Development Program of China under Grant 2018YFB1701600, National Natural Science Foundation of China under Grants U20B2069 and 61573045, and the 13th Five-Year Civil Space Key Project.

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016. 1, 2

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, pages 6182–6191, 2019. 1, 7, 8

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *ECCV*, pages 205–221, 2020. 7, 8

[4] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, pages 483–498, 2018. 7, 8

[5] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *TPAMI*, 7(04):669–688, 1993. 2

[6] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *CVPR*, pages 6668–6677, 2020. 1, 2, 7, 8

[7] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time target-aware visual tracking. In *ICCV*, pages 911–920, 2019. 1

[8] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546, 2005. 2

[9] Kenan Dai, Dong Wang, Huchuan Lu, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *CVPR*, pages 4670–4679, 2019. 7

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019. 7, 8

[11] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, pages 7183–7192, 2020. 1, 7, 8

[12] Fei Du, Peng Liu, Wei Zhao, and Xianglong Tang. Correlation-guided attention for corner detection based visual tracking. In *CVPR*, pages 6836–6845, 2020. 7

[13] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, pages 5374–5383, 2019. 1, 5, 7

[14] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, pages 7952–7961, 2019. 1, 2

[15] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *CVPR*, pages 6269–6277, 2020. 1, 2, 7, 8

[16] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, pages 1763–1771, 2017. 1, 2

[17] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2019. 5, 7

[18] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Globaltrack: A simple and strong baseline for long-term tracking. In *AAAI*, volume 34, pages 11037–11044, 2020. 7

[19] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Martin Danelljan, Alan Lukezic, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernandez, and et al. The eighth visual object tracking vot2020 challenge results, 2020. 4

[20] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pfugfelder, Luka Čehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al. The sixth visual object tracking vot2018 challenge results, 2018. 1, 5, 8

[21] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, pages 1378–1387, 2016. 2

[22] Kuan-Hui Lee and Jenq-Neng Hwang. On-road pedestrian tracking across multiple driving recorders. *TMM*, 17(9):1429–1438, 2015. 1

[23] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, pages 4282–4291, 2019. 1, 2, 7, 8

[24] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018. 1, 2

[25] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Gradnet: Gradient-guided network for visual object tracking. In *ICCV*, pages 6162–6171, 2019. 1, 2

[26] Bingyan Liao, Chenye Wang, Yayun Wang, Yaonong Wang, and Jun Yin. Pg-net: Pixel to global matching network for visual tracking. In *ECCV*, 2020. 7, 8

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 4

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 5

[29] Liwei Liu, Junliang Xing, Haizhou Ai, and Xiang Ruan. Hand posture recognition using finger geometric feature. In *ICPR*, pages 565–568, 2012. 1

[30] Xiankai Lu, Wenguan Wang, Danelljan Martin, Tianfei Zhou, Jianbing Shen, and Van Gool Luc. Video object segmentation with episodic graph memory networks. In *ECCV*, 2020. 2

[31] Alan Lukezic, Jiri Matas, and Matej Kristan. D3s-a discriminative single shot segmentation tracker. In *CVPR*, pages 7133–7142, 2020. 7, 8

[32] Ziang Ma, Linyuan Wang, Haitao Zhang, Wei Lu, and Jun Yin. Rpt: Learning point set representation for siamese visual tracking. *arXiv preprint arXiv:2008.03467*, 2020. 7

[33] Tal Makovski and Yuhong V Jiang. The role of visual working memory in attentive tracking of unique objects. *Journal of Experimental Psychology: Human Perception and Performance*, 35(6):1687, 2009. 2

[34] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016. 2, 4

[35] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, pages 445–461, 2016. 5, 7

[36] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, pages 300–317, 2018. 5, 6

[37] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, pages 9226–9235, 2019. 2, 4, 5

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5

[39] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016. 2

[40] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015. 2

[41] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, pages 489–497, 2018. 7

[42] Yuxuan Sun, Chong Sun, Dong Wang, You He, and Huchuan Lu. Roi pooled correlation filters for visual tracking. In *CVPR*, pages 5783–5791, 2019. 7

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 5, 6

[44] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, pages 9627–9636, 2019. 4

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 4

[46] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *CVPR*, pages 6578–6588, 2020. 5, 8

[47] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *CVPR*, pages 6288–6297, 2020. 7, 8

[48] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. In *CVPR*, pages 3643–3652, 2019. 7

[49] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, pages 4844–4853, 2018. 7

[50] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, pages 1328–1338, 2019. 1, 2

[51] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018. 4

[52] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. 2

[53] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 1, 5, 6

[54] Junliang Xing, Haizhou Ai, and Shihong Lao. Multiple human tracking based on multi-view upper-body detection and discriminative learning. In *ICPR*, pages 1698–1701, 2010. 1

[55] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Joint group feature selection and discriminative filter learning for robust visual object tracking. In *ICCV*, pages 7950–7960, 2019. 7

[56] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020. 1, 2, 5, 7, 8

[57] Bin Yan, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. *arXiv preprint arXiv:2007.02024*, 2020. 4

[58] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *ECCV*, pages 152–167, 2018. 1, 2

[59] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *CVPR*, pages 6728–6737, 2020. 1, 7, 8

[60] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, pages 4353–4361, 2015. 2

[61] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, pages 4591–4600, 2019. 1, 2

[62] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020. 7, 8

[63] Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, and Hanqing Lu. Learning feature embeddings for discriminant model based tracking. In *ECCV*, 2020. 7

[64] Jinghao Zhou, Peng Wang, and Haoyang Sun. Discriminative and robust online learning for siamese visual tracking. In *AAAI*, pages 13017–13024, 2020. 7

[65] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, pages 101–117, 2018. 1, 2