2019 IEEE International Conference on Multimedia and Expo (ICME)

LEARNING MOTION-AWARE POLICIES FOR ROBUST VISUAL TRACKING

Qianqian Wang, Liansheng Zhuang, Ning Wang, Wengang Zhou, Houqiang Li

CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System
EEIS Department, University of Science and Technology of China
{wqian,wn6149}@mail.ustc.edu.cn {lszhuang,zhwg,lihq}@ustc.edu.cn

ABSTRACT

Visual object tracking aims to locate a moving target specified at the initial frame. Although this task is closely related to the temporal motion information, the motion model typically draws limited attention. In this paper, we propose a motion-aware multi-domain network for robust visual tracking. In our approach, a motion-aware agent is trained via reinforcement learning, which can infer the parameters of the particle filter in a continuous action space. Different from existing tracking-by-detection frameworks that the particle filter merely relies on the previous target state, our motion-aware agent, after receiving the current state, can adaptively change the parameters of the particle filter (*e.g.,* particle location and scale range). As a result, our approach samples high-quality candidates for further classification/tracking, thus can better handle challenges such as fast motion and scale variation. Extensive experiments on large-scale benchmarks verify the effectiveness of our method.

Index Terms— visual tracking, tracking-by-detection, particle filter, reinforcement learning

1. INTRODUCTION

Visual tracking is a fundamental task in computer vision with numerous applications such as video surveillance, human-computer interaction and augmented reality. In spite of considerable progress [1, 2, 3] in recent years, it still remains a challenging task due to factors like fast motion, occlusion, deformation, *etc*.

Many state-of-the-art tracking algorithms [1, 4, 5] follow the tracking-by-detection paradigm, which is typically based on a two-stage framework. In the first stage, a tracker samples plentiful candidates via a particle filter, and in the second stage, it selects the most reliable candidate as the tracking result. In recent years, the second stage has drawn considerable attention and many sophisticated classifiers have been proposed to effectively distinguish the target from background candidates. In contrast, the first stage, which mainly models the target motion information and plays a vital role in many challenging scenes such as fast motion and motion blur, receives limited attention. Under a classic particle filter framework, in each frame, most tracking-by-detection methods simply draw abundant candidates around the target position in the previous frame. This straightforward strategy is based on the assumption that the target object is unlikely to move beyond a large range in a short span of time. However, in case of target fast motion or camera shaking, the aforementioned general strategy may fail to capture enough candidate boxes near the current target location, which will further disturb the second-stage classification and finally lead to the unsatisfactory results.

In this paper, we propose a motion-aware agent to guide the particle sampling. This agent provides adaptive policies, which can estimate the suitable parameters of the particle filter in each frame by analyzing the current tracking environment as well as the predicted results. Compared with the traditional particle filter, the proposed agent has the advantage to estimate a better particle sampling location, scale range and aspect ratio. Accordingly, the high-quality candidates are provided to facilitate the second-stage classification, especially in the challenging scenes such as fast motion, motion blur or scale variation. As a result, our method shows superior results in the above scenes. Specifically, we train the agent in a deep reinforcement learning fashion during the training phase. After exploring plentiful videos offline, our agent directly provides the particle sampling policies during online tracking, which is extremely efficient with an ignorable additional computation complexity.

In summary, the main contribution of this work is the introduction of a motion-aware agent for adaptive particle sampling in the tracking-by-detection framework. Even though quite intuitive, our method focuses on a non-trivial component in the tracking system, *i.e.,* the motion model, which has been largely overlooked in recent years. Furthermore, we formulate this problem in the reinforcement learning framework to train a motion-aware agent. Finally, extensive experiments on large-scale datasets show that our method achieves state-of-the-art performance, and outperforms the tracking-

This work was supported in part to Dr. Houqiang Li by the 973 Program under Contract No. 2015CB351803 and NSFC under contract No. 61836011, in part to Dr. Liansheng Zhuang by NSCF under contract No.61472379, and in part to Dr. Wengang Zhou by NSFC under contract No. 61632019 and Young Elite Scientists Sponsorship Program By CAST (2016QNRC001).

978-1-5386-9552-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICME.2019.00307

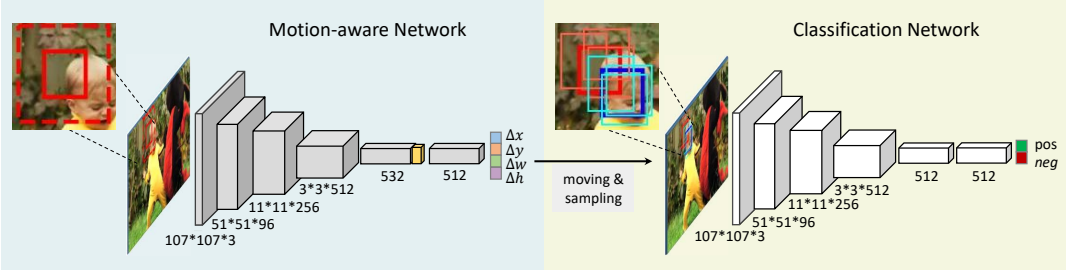


Fig. 1: The overall architecture of our method. In the left figure, the solid line is the target bounding box in the previous frame and the dotted line shows the twice target size with the same center. In the right figure, we show a more precise state after executing the action given by the motion-aware network and candidates drawn with our particle sampling strategy. Then the classification network gives positive and negative scores for every candidate to distinguish the target from the background.

by-detection based tracker using general particle sampling strategy (*e.g.,* MDNet [1]).

2. RELATED WORK

In this section, we briefly introduce the tracking-by-detection based methods and reinforcement learning.

Tracking-by-detection. The approach [1, 6] based on tracking-by-detection aims to build a discriminative classifier that distinguishes the target from the surrounding background. Deep correlation filter based tracker is trained through minimizing a least-squares loss for all circular shifts of a training sample, and is further enhanced by adopting an adaptive scale scheme [7] or a part-based framework [8], alleviating the boundary effects [9, 10], combining with re-detection modules [11] and so on. The MDNet method proposed in [1] trains a CNN in a multi-domain learning framework and shows outstanding performance among state-of-the-art methods. However, the above tracking-by-detection methods mostly ignore the motion information between the successive frames, and they simply draw a search window or plentiful candidates on the target position predicted in the previous frame. In contrast, we propose a motion-aware network to guide the candidate sampling for better classification.

Reinforcement Learning. The field of reinforcement learning (RL) has revived with the power of deep learning in recent years. RL is a machine learning technique considering how an agent learns to make decisions by interacting with the environment to achieve a given goal. Many algorithms have emerged such as DQN [12] and DPPG [13] to solve the sequential decision-making problems. This framework has recently been applied to video tracking tasks. ADNet [14] learns a policy to select an optimal discrete action to track the target, while ACT [2] exploits continuous actions to address this issue. EAST [15] learns an agent to adaptively select the cheap shallow feature or expensive deep feature. Deep RL is also used for hyperparameter optimization for tracking in [16]. Different from the previous methods, in this paper, we focus on learning a robust motion model using deep RL.

3. METHOD

Figure 1 shows an overview of our tracking framework, which consists of a motion-aware network and a classification network. In the rest of this section, we first revisit the MDNet method in Section 3.1. Then we introduce the details of our deep reinforcement learning-based motion-aware network in Section 3.2. Online tracking and discussion are elaborated in Section 3.3 and Section 3.4.

3.1. Revisiting MDNet Tracker

The MDNet method [1] not only learns the shared representation of targets but also the domain-specific information where each video is regarded as a separate domain. To this end, the network has K branches for output domain-specific layers corresponding to K sequences, while all the preceding layers are shared to capture the general feature representations. The model receives a 107×107 RGB input and contains five hidden layers including three convolutional layers followed by two fully connected layers. In the offline training phase, each domain is processed individually. Namely, only one single branch corresponding to the video sequences is enabled in each iteration. During online tracking, the multiple branches of output layers are replaced with a new domain-specific layer for each test sequence. During online tracking, the tracking-by-detection method mainly consists of two steps as follows.

Particle Sampling. To estimate the target state in the t-th frame, $N(= 256)$ candidates, $\mathbf{x}_t^i = (x_t^i, y_t^i, s_t^i)$, $i = 1, \cdots, N$, are drawn from a Gaussian distribution based on the previous target box $b_{t-1}^* = \{x_{t-1}^*, y_{t-1}^*, w_{t-1}^*, h_{t-1}^*\}$, and the mean as well as the covariance of the Gaussian distribution are defined as follows,

$$\begin{aligned} \text{mean} &= \{x_{t-1}^*, y_{t-1}^*, 0\}, \\ \text{covariance} &= diag(0.09r^2, 0.09r^2, 0.25), \end{aligned} \tag{1}$$

where

$$r = (w_{t-1}^* + h_{t-1}^*)/2,$$

and $diag(\cdot)$ means the diagonal matrix. Besides, the scale of each sample is computed by multiplying $1.05^{s_t^i}$. In this case, width and height can only be enlarged or reduced in the same proportion. Note that the above strategy simply relies on the previous target state and does not exploit the motion information between successive frames.

Binary Classification. In the second stage, the above samples are further evaluated by the classification network. After computing their positive scores $f^+(\mathbf{x}_t^i)$ and negative scores $f^-(\mathbf{x}_t^i)$, the optimal target state b_t^* is estimated by searching the sample with the maximum positive score.

3.2. Learning Motion-aware Network

Our Particle Sampling Strategy. Traditional tracking-by-detection methods typically take a Gaussian sampling strategy as described in Section 3.1. Admittedly, the location in the previous frame provides a strong clue to guide the target searching in the current frame. However, the rich information in the current frame is ignored. To this end, we propose a motion-aware network to estimate the movement of the target object to direct where and how to draw the candidates.

Specifically, the motion-aware network outputs a continuous action $a = [\Delta x, \Delta y, \Delta w, \Delta h]$ to predict the particle sampling, where Δx and Δy denote the horizontal and the vertical scale-invariant shift, respectively, and $\Delta w, \Delta h$ represent the width and height variation range of the particles in the current frame. Then the mean and covariance of our Gaussian distribution can be formulated as follows,

$$\begin{aligned} \text{mean} &= \{x_{t-1}^*(1+\Delta x),\ y_{t-1}^*(1+\Delta y), 0\}, \\ \text{covariance} &= diag(0.09r^2, 0.09r^2, 0.25), \end{aligned} \tag{2}$$

where

$$r = [w_{t-1}^*(1+\Delta w) + h_{t-1}^*(1+\Delta h)]/2.$$

Then the width and height of each sample are computed by multiplying $(1+\Delta w)\cdot 1.05^{s_t^i}$ and $(1+\Delta h)\cdot 1.05^{s_t^i}$, respectively. Obviously, variations in width and height are independent of each other, so as to obtain candidates with different aspect ratios.

By performing action $a = [\Delta x, \Delta y, \Delta w, \Delta h]$, samples are more likely to overlap or even coincide with the ground truth than the usual sampling method. Thus we will get a more accurate location estimation, which is exactly what we expect.

Problem Settings. We formulate the motion prediction problem as a Markov Decision Process (MDP) since this setting provides a formal framework to model an agent that makes a sequence of decisions to optimize a goal by interacting with the environment. In this paper, we consider a single image as the environment, where the agent infers the motion of the target. Formally, the MDP has a set of actions A, a set of states S, and a reward function R. In each frame of the training sequences, the agent gives an action $a \in A$ according to the current state $s \in S$, and then receives a positive or negative reward after executing this action.

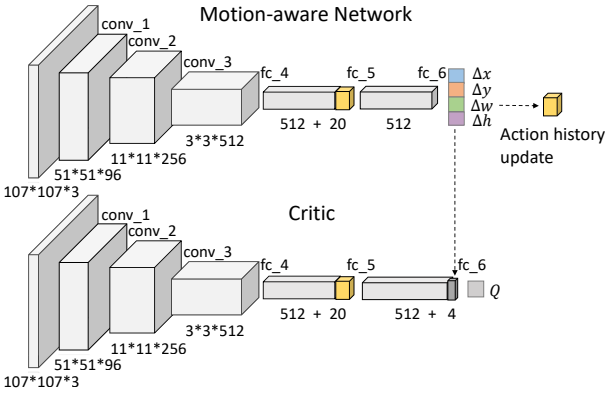


Fig. 2: An overview of the motion-aware network and critic structure.

The state representation in our work is a tuple (o, d), where o is the current observed image patch and d is a vector of the action history. Specifically, given a video sequence, in the t-th frame, the observed image patch o is cropped from the image within bounding box b_{t-1}^* and the twice target size with the same center (the crop operation below is the same as here). The vector d is concatenated by the last D actions. Despite that d is low-dimensional compared to o, it provides rich information of the past actions which is helpful for motion smoothness and motion modeling. Given the current state s, our motion-aware network gives a continuous action a to predict the motion and scale transform of the target to get a new bounding box b_t. Afterward, the next state s' can be obtained by cropping the image within b_t and updating the action history according to a_t.

After executing the action a, we give a $+1$ reward signal to encourage our agent if the Intersection-over-Union (IoU) between the ground truth g_t and the predicted bounding box b_t is larger than 0.7, where $IoU(b_t, g_t) = area(b_t \cap g_t)/area(b_t \cup g_t)$. Otherwise, a -1 reward signal will be given to penalize the agent.

Network Architecture. We adopt DDPG algorithm [13] based on the Actor-Critic framework to train our motion-aware network. In the DDPG method, there is a critic network to suggest the learning direction for actor network. As illustrated in Figure 2, the motion-aware network, *i.e.,* the actor in our work, uses three convolutional layers to extract the image patch feature. The following layer fc_4 is a fully connected layer with 512 units and combined with ReLUs. The feature after fc_4 and the action history are concatenated and fed into fc_5 with 532 units and $tanh$. We keep the recent 5 actions in our setup. The output layer predicts a four-dimensional action predicting the target's motion. The critic network has the same structure with the motion-aware network, except that the feature vector after fc_5 should be concatenated with the ac-

tion vector before the output layer, meanwhile its output layer only has one unit which represents the Q value for the action executed on the corresponding state.

Network Training. We use the pre-trained VGG-M [17] to initialize our motion-aware network and critic network. We denote the motion-aware network as $\mu(s|\theta)$. The output of the critic, expressed as $Q(s, a|\omega)$ is learned using the recursive relationship known as the Bellman equation. We create a copy of the actor and critic networks, $\mu'(s|\theta')$ and $Q'(s, a|\omega')$, respectively, which are used for calculating the target values. Given some pairs of (s_t, a_t, r_t, s'_t), we can optimize the motion-aware network by minimizing the loss:

$$L(\omega) = \mathbb{E}_{s_t,a_t}[(Q(s_t, a_t|\omega) - y_t)^2], \tag{3}$$

where

$$y_t = r_t + \gamma Q'(s'_t, \mu'(s'_t|\theta')|\omega'). \tag{4}$$

The motion-aware network is updated by the chain rule to the expected return from the start distribution J with respect to θ:

$$\nabla_\theta J = \mathbb{E}_{s_t}[\nabla_a Q(s, a|\omega)|_{s=s_t,a=\mu(s_t)} \nabla_\theta \mu(s|\theta)|_{s=s_t}]. \tag{5}$$

The weights of these target networks are then updated by having them slowly track the learned networks:

$$\begin{aligned} \theta' &\leftarrow \tau\theta + (1-\tau)\theta', \\ \omega' &\leftarrow \tau\omega + (1-\tau)\omega', \end{aligned} \tag{6}$$

where $\tau = 0.001$ to constrain the target network to change slowly.

During training, we randomly select a video sequence from the training set. Following previous methods, in the first frame, we use supervised learning with L_2 loss to train our model, due to the huge state and action space for our agent to randomly explore. That is, the ground truth actions of the training samples will be given, and then the parameter can be updated by minimizing the following loss:

$$\mathbb{E}_k \|\mu(s_k|\theta) - a_k\|_2^2, \tag{7}$$

where s_k is the k-th sample and a_k is the ground truth action. In the following frames, for instance, in frame t, our motion-aware network interacts with the image to get the transition (s_t, a_t, r_t, s'_t) which will be stored in a reply buffer for training. We also conduct the ε-greedy method to alleviate the problem that the positive reward signals are seriously less than negative rewards. Namely, an expert policy is adopted with the probability ε, ε equals 0.5 at first and anneals along with the training by multiplying 0.05 every 10k iterations.

3.3. Online Tracking

Once we complete the motion-aware network training described in Section 3.2, we remove the critic network and only use the motion-aware network to guide where to sample during the online tracking process. Although the network has learned the motion of the objects based on the current frame only, the target object of each video sequences has its own specific attributes. Hence we fine-tune the motion-aware network in the first frame following the L_2 loss described above.

To estimate the target bounding box b_t^* in current frame t, we first crop the image patch within the previous target bounding box b_{t-1}^* and read the action history from the cache as the input to the motion-aware network, then we get a closer location b_t to the target object by performing the output action a_t. $N = 128$ samples are drawn using Eq. 2. And another 128 samples are drawn following Eq. 1.

In the second stage, all the samples are fed into the classification network. The sample with the maximum positive score is our final target object status in the current frame. For more details of the classification network, please refer to [1].

3.4. Discussion

How does the motion-aware agent work? Our motion-aware network performs as a regression model and can be regarded as a local object detection network. The network extracts the rich motion information in the current frame to direct particle sampling, thus boosting the tracking-by-detection framework.

The role of reinforcement learning. The previous art [16] proves that deep reinforcement learning is an effective way for hyperparameter learning. Different from it, our method focuses on motion information modeling and estimates the parameters for particle filter, and solves this continuous problem in the DDPG framework.

The extra computational burden. After offline reinforcement learning, our motion-aware network merely involves one feed-forward computation for each frame. Besides, our network is relative lightweight with only three convolutional layers, which also guarantees the efficiency.

Finally, our motion-aware network is quite general and can be combined with other tracking-by-detection methods.

4. EXPERIMENTS

4.1. Experimental Details

In our experiments, the training sets are 768 video sequences from ILSVRC [18] trainval set. We randomly select $20 \sim 40$ consecutive frames from a video to train our motion-aware network and the critic. In the first frame, we take 32 samples whose IoU scores with the ground truth are larger than 0.7 to train our network by supervised learning, and the learning rate here is 1e-4. In the following frames, the learning rate of motion-aware network and the critic is 1e-6 and 1e-5, respectively. The batch size is 128. In the online tracking phase, we take 500 positive samples to fine-tune our motion-aware network with learning rates 1e-4. We follow the parameters

Table 1: Experimental results of MDNet and our approach on OTB-2015 [19] and Temple-Color [20] benchmark. The distance precision (DP) at 20 pixels threshold and overlap precision (OP) at an overlap threshold 0.5 are reported.

	DP		OP	
	MDNet	Ours	MDNet	Ours
OTB-2015	87.9%	89.8% (**1.9%** ↑)	82.0%	83.8% (**1.8%** ↑)
Temple-Color	80.5%	82.0% (**1.5%** ↑)	72.5%	73.9% (**1.4%** ↑)

in the MDNet method for classification network. The experiment was conducted on a computer with an Intel Core i9-7900X 3.30GHz CPU and a NVIDIA GeForce GTX 1080 Ti GPU.

We evaluate the proposed method on two standard benchmarks: OTB-2015 [19] and Temple-Color [20]. All the tracking methods are evaluated by conducting a one-pass evaluation (OPE) based on two metrics: center location error and bounding box overlap ratio.

4.2. Ablation Study

In Table 1, we exhibit the experimental results on the OTB-2015 [19] and Temple-Color [20] benchmark to verify the effectiveness of our proposed framework. Following our method, the candidates tend to be denser around the ground truth. Besides, $\Delta w, \Delta h$ changes the width and height of the bounding box, which means we are more likely to get the candidates having the same scale and aspect ratio with the ground truth than the original MDNet method. For detailed performance analysis, we also report the results on the common challenging factors on the OTB-2015 in Table 2. The results show that our method outperforms the baseline by a large margin almost on all attributes, especially on fast motion, scale variation, deformation and motion blur.

It should be noted that our baseline, *e.g.,* MDNet, has already achieved an excellent performance level, but our method still obviously boosts its performance. Since the only difference between our method and the MDNet tracker is the addition of motion-aware network, we can attribute the performance gain to our motion-aware agent without doubt.

The MDNet in our experiments is the Python version[1] and all of our implementations are based on Python.

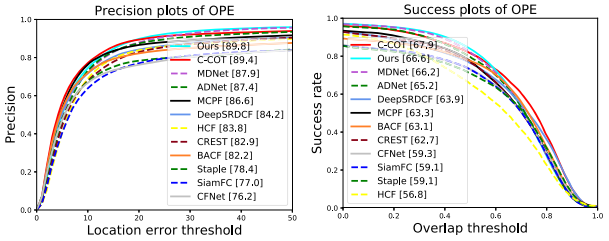
4.3. State-of-the-art Comparison

We present the evaluation results of our method and the other 11 state-of-the-art trackers including MDNet [1], ADNet [14], DeepSRDCF [21], HCF [22], MCPF [7], C-COT [9], CREST [23], BACF [10], Staple [24], SiamFC [25], and CFNet [26]. Figure 3 illustrates the precision and success plots of OPE based on distance precisoin and overlap ratio. Our tracker provides the state-of-the-art performance on the OTB-2015 [19] against other trackers. Compared to the

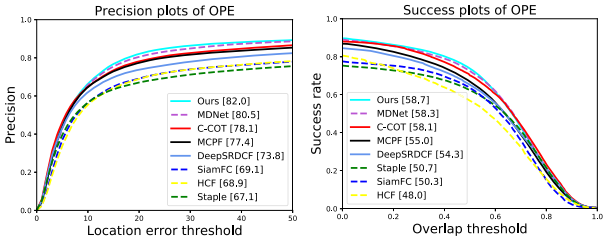
[1]Python version of MDNet open source code is available at https://github.com/HyeonseobNam/py-MDNet

Table 2: Comparisons with baseline tracker under 11 challenging factors, including Illumination Variation (IV), Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Motion Blur (MB), Fast Motion (FM), In-Plane Rotation (IPR), Out-of-Plane Rotation (OPR), Out-of-View (OV), Background Clutters (BC) and Low Resolution (LR). The DP (%) (@20px) and OP (%) at an overlap threshold 0.5 are reported. The best performances are highlighted in bold.

	Method	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR
DP	MDNet	**91.1**	85.1	84.7	83.9	82.1	81.9	88.8	87.1	79.9	88.6	94.2
	Ours	90.8	**88.6**	**85.7**	**88.9**	**84.4**	**85.4**	**90.6**	**89.1**	**80.8**	**90.7**	**94.7**
		-0.3	**+3.5**	**+1.0**	**+5.0**	**+2.3**	**+3.5**	**+1.8**	**+2.0**	**+0.9**	**+2.1**	**+0.5**
OP	MDNet	85.8	76.4	79.1	75.9	82.4	77.6	79.4	79.8	**77.6**	**84.3**	76.0
	Ours	**86.6**	**79.4**	**80.5**	**79.7**	**83.6**	**80.5**	**80.4**	**81.3**	77.1	84.0	**76.2**
		+0.8	**+3.0**	**+1.4**	**+3.8**	**+1.2**	**+2.9**	**+1.0**	**+1.5**	-0.5	-0.3	**+0.2**



(a) OTB-2015



(b) Temple-Color

Fig. 3: Precision and success plots with Area Under the Curve (AUC) for OPE on OTB-2015 [19] and Temple-Color [20]. In the legend, the DP at a threshold of 20 pixels and AUC scores are reported.

ADNet method, which is also trained based on reinforcement learning, our method achieves a notable absolute gain of 2.4% and 1.4% in distance precision and AUC score on the OTB-2015, respectively. On a more challenging dataset, Temple-Color [20], the proposed method still achieves the best results on both metrics. As shown in Fig. 3(b), our proposed tracker provides the distance precision score of 82.0% on Temple-Color, which outperforms the C-COT method by a significant margin of 3.9%.

Figure 4 presents the results comparing with MDNet [1], MCPF [7], SiamFC [25] and C-COT [9] qualitatively in several challenging sequences. It shows that in the case of short time occlusion, fast motion or motion blur, our tracker shows superior results thanks to the motion information used in the current frame.

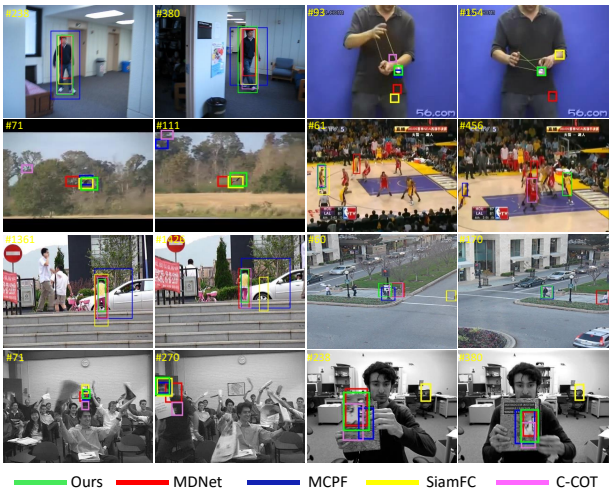


Fig. 4: Qualitative evaluation of our tracker, MDNet [1], MCPF [7], SiamFC [25] and C-COT [9] on several challenging video sequences (*BlurBody, Yo-yos_ce2, Eagle_ce, Kobe_ce, Girl2, Human3, Freeman4, ClifBar*).

5. CONCLUSION

In this paper, we proposed a motion-aware network to direct the particle sampling in the tracking-by-detection framework. Taking advantage of the motion information, we predict a more precise location of the target object. Experimental results prove that our tracker achieves the state-of-the-art performance on the challenging datasets.

6. REFERENCES

[1] Hyeonseob Nam and Bohyung Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, 2016.

[2] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu, "Real-time 'actor-critic' tracking," in *ECCV*, 2018.

[3] Jianglei Huang and Wengang Zhou, "Re2ema: Regularized and reinitialized exponential moving average for target model update in object tracking," in *AAAI*, 2019.

[4] Yibing Song, Chao Ma, Xiaohe Wu, et al., "Vital: Visual tracking via adversarial learning," in *CVPR*, 2018.

[5] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li, "Multi-cue correlation filters for robust visual tracking," in *CVPR*, 2018.

[6] Zdenek Kalal, Krystian Mikolajczyk, Jiri Matas, et al., "Tracking-learning-detection," *IEEE TPAMI*, vol. 34, no. 7, pp. 1409–1422, 2012.

[7] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang, "Multi-task correlation particle filter for robust object tracking," in *CVPR*, 2017.

[8] Ning Wang, Wengang Zhou, and Houqiang Li, "Robust object tracking via part-based correlation particle filter," in *ICME*, 2018.

[9] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *ECCV*, 2016.

[10] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey, "Learning background-aware correlation filters for visual tracking.," in *ICCV*, 2017.

[11] Ning Wang, Wengang Zhou, and Houqiang Li, "Reliable re-detection for long-term tracking," *IEEE TCSVT*, vol. 29, no. 3, pp. 730–743, 2018.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, et al., "Continuous control with deep reinforcement learning," in *ICLR*, 2016.

[14] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *CVPR*, 2017.

[15] Chen Huang, Simon Lucey, and Deva Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *ICCV*, 2017.

[16] Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli, "Hyperparameter optimization for tracking with continuous deep q-learning," in *CVPR*, 2018.

[17] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2014.

[18] Olga Russakovsky, Jia Deng, Su, et al., "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[19] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Object tracking benchmark," *IEEE TPAMI*, vol. 37, no. 9, pp. 1834–1848, 2015.

[20] Pengpeng Liang, Erik Blasch, and Haibin Ling, "Encoding color information for visual tracking: Algorithms and benchmark," *IEEE TIP*, vol. 24, no. 12, pp. 5630–5644, 2015.

[21] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg, "Convolutional features for correlation filter based visual tracking," in *ICCV Workshops*, 2015.

[22] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang, "Hierarchical convolutional features for visual tracking," in *ICCV*, 2015.

[23] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang, "Crest: Convolutional residual learning for visual tracking," in *ICCV*, 2017.

[24] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr, "Staple: Complementary learners for real-time tracking," in *CVPR*, 2016.

[25] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV*, 2016.

[26] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr, "End-to-end representation learning for correlation filter based tracking," in *CVPR*, 2017.