Convolutional Adaptive Particle Filter with Multiple Models for Visual Tracking

Reza Jalil Mozhdehi, Yevgeniy Reznichenko, Abubakar Siddique, and Henry Medeiros

Electrical and Computer Engineering Department, Marquette University, Milwaukee, WI, USA
{reza.jalilmozhdehi, yevgeniy.reznichenko, abubakar.siddique and henry.medeiros}@marquette.edu

Abstract. Although particle filters improve the performance of convolutional-correlation trackers, especially in challenging scenarios such as occlusion and deformation, they considerably increase the computational cost. We present an adaptive particle filter to decrease the number of particles in simple frames in which there is no challenging scenario and the target model closely reflects the current appearance of the target. In this method, we consider the estimated position of each particle in the current frame as a particle in the next frame. These refined particles are more reliable than sampling new particles in every frame. In simple frames, target estimation is easier, therefore many particles may converge together. Consequently, the number of particles decreases in these frames. We implement resampling when the number of particles or the weight of the selected particle is too small. We use the weight computed in the first frame as a threshold for resampling because that weight is calculated by the ground truth model. Another contribution of this article is the generation of several target models by applying different adjusting rates to each of the high-likelihood particles. Thus, we create multiple models; some are useful in challenging frames because they are more influenced by the previous model, while other models are suitable for simple frames because they are less affected by the previous model. Experimental results on the Visual Tracker Benchmark v1.1 beta (OTB100) demonstrate that our proposed framework significantly outperforms state-of-the-art methods.

Keywords: Adaptive Particle Filter · Deep Convolutional Neural Network · Correlation Models · Adjusting Rate · Visual Tracking.

1 Introduction

Tracking a specific target in challenging scenarios such as occlusion and deformation has attracted the attention of computer vision researchers for decades. Recently, deep convolutional neural networks (CNN) have been used to extract the target's features. One particularly effective mechanism to determine the similarity between an image patch and the target is the correlation filter tracking framework [1,2,3]. Trackers based on correlation filters measure the correlation between the target model and an image patch in the frequency domain and are agnostic to the features used to represent the targets. As a consequence, most state-of-the art CNN-based trackers integrate convolutional features and correlation filters [4]. The Hierarchical Convolutional Feature Tracker (HCFT) proposed by Ma et al. [5] uses the hierarchical convolutional features

generated by multiple layers of a deep CNN in a coarse-to-fine manner in conjunction with the correlation filter proposed in [6]. HCFT shows substantial performance improvement in comparison with other visual trackers such as MEEM [7], Struck [8], SCM [9] and TLD [10]. Thus, the idea of employing features generated using CNNs with different filtering mechanisms is becoming increasingly more popular. However, the main limitation of these correlation filters is to generate only one model in each frame, which leads to high dependency on the estimated target size and position. Incorrectly updating the model causes inaccuracies in the determination of the target size and position in subsequent frames.

In this article, we extend our previous visual tracker named Deep Convolutional Particle Filter with Adaptive Correlation Maps (DCPF2) [11]. Similar to DCPF [12], DCPF2 employs a particle filter in conjunction with a CNN and an adaptive correlation filter. However, DCPF considers the target size to be fixed while DCFP2 also estimates the target size in each frame. In our new tracker named Deep Convolutional Adaptive Particle Filter with Multiple Correlation Models (CAP-mc), we replace the particle filter proposed in DCPF2 with an adaptive particle filter. Adaptive particle filters can improve the results of object tracking [13]. However, they have not been used in conjunction with CNNs and correlation filters yet. Our adaptive particle filter decreases the number of particles and the computation cost in simple frames in which there is no challenging scenario and the target model closely reflects the current appearance of the target. Additionally, our adaptive particle filter can refine the particles' locations to be used in the next frame. This method is more reliable than sampling new particles in every frame which was employed in DCPF2. We use the weight calculated in the first frame as one of the resampling thresholds because it is based on the ground truth target model, and hence serve as a reference for the quality of subsequent models. Another threshold for resampling is the number of particles.

Additionally, we determined that the adjusting rate is a critical parameter in correlation-based trackers. The adaptive correlation filter proposed in DCPF2 generates several target models based on all the high-likelihood particles to cover probable errors instead of generating one model based on the selected particle. In our new tracker, we apply different adjusting rates to generate several target models for each high-likelihood particle. Thus, we create multiple models; some are less affected by the previous model, such models are useful in simple frames, while other models, more affected by the previous model, are suitable for challenging frames. We tested our tracker on The Visual Tracker Benchmark v1.1 beta (OTB100) [14], and the results show outstanding performance of our tracker against state-of-the-art methods.

2 Deep Convolutional Adaptive Particle Filter with Multiple Correlation Models

In this section, we present our adaptive particle filter illustrated in Fig. 1. We then discuss our new adaptive correlation filter based on employing different adjusting rates.

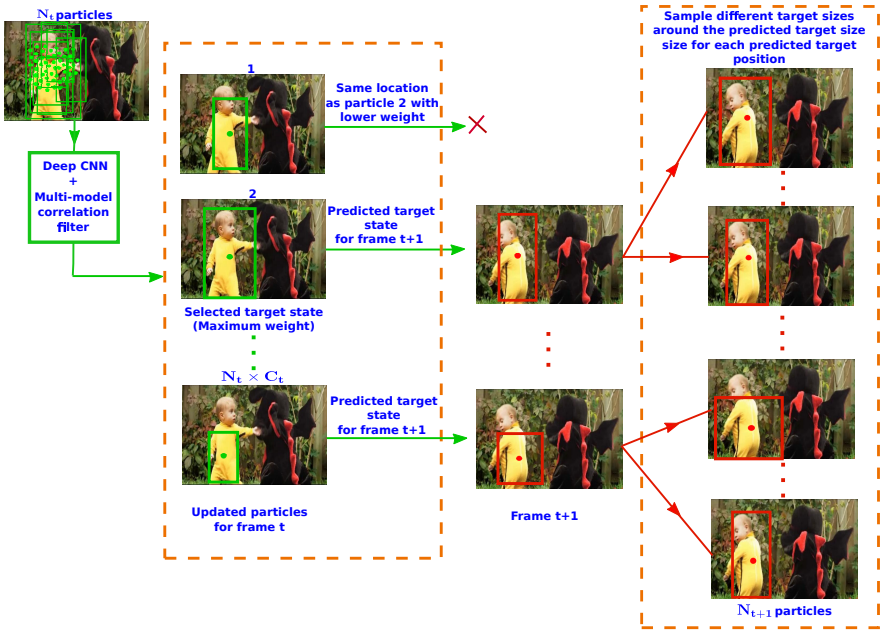

Fig. 1. Our proposed adaptive particle filter (when resampling is not needed).

2.1 Adaptive Particle Filter

Algorithm 1 explains our adaptive particle filter. In visual tracking, the ground truth target position and size are used for initialization. Let z_1 be the ground truth target position and size in the first frame

$$z_1 = \left[u_1, v_1, h_1, w_1\right]^T, \tag{1}$$

where u_1 and v_1 are the ground truth locations of the target and h_1 and w_1 are its ground truth width and height. The target state is defined by

$$x_1 = \left[z_1, \dot{z}_1\right]^T, \tag{2}$$

where $\dot{z}_1$ is the velocity of z_1 and is assumed to be zero in the first frame. After extracting a patch from the first frame based on based on z_1, we feed this patch to a CNN [15] to calculate its convolutional features. the ground truth target model is then generated by applying the Fourier transform to the convolutional features as explained in [5]. The ground truth target model is used in calculating a threshold for the resampling process as explained later in this section. Additionally, this model is updated during the next frames as discussed in the next section.

In the next step, we generate and evaluate the initial particles as explained in Algorithm 2. Considering x_{t-1} as the previous target state, the predicted target state is calculated by [11]

$$\hat{x}_t = Ax_{t-1}, \tag{3}$$

where t is the frame number and A is a standard constant velocity process matrix defined by

$$A = \begin{bmatrix} I_4 & \mid I_4 \\ 0_{(4,4)} & \mid I_4 \end{bmatrix}, \tag{4}$$

where I_y is an $y \times y$ identity matrix and $0_{(u,z)}$ is a $u \times v$ zero matrix. It is clear that $\hat{x}_2 = x_1$ because of $\dot{z}_1 = 0$. As discussed in [11], by sampling from a zero-mean normal distribution and adding those samples $\zeta_t \in \mathbb{R}^8$ to $\hat{x}_t$, the particles are generated according to

$$x_t^{(i)} = \hat{x}_t + \zeta_t^{(i)}, \tag{5}$$

where

$$x_t^{(i)} = \left[z_t^{(i)}, \dot{z}_t^{(i)} \right]^T, \tag{6}$$

where $i = 1, ..., N_t$ and N_t is the number of initial particles.

Algorithm 1 Adaptive particle filter

Input: Current frame, previous target state x_{t-1} and C_{t-1} target models $\mho_{t-1}^{(j)}$
Output: Current target state x_t, particles $x_{t+1}^{(i)}$ for the next frame

```
Generate initial particles to determine the target state x_t according to Algorithm 2
if t = 1 then
    Calculate T_w
end if
Update particles and remove redundant ones using Algorithm 3
Examine the resampling conditions according to Eq. 15 and Eq. 16
if resampling is needed then
    Generate particles x_{t+1}^{(i)} for the next frame based on Algorithm 2
else
    Calculate the predicted particles x̂_{(t+1)}^{(p)} for frame t + 1 using Eq. 17
    for each x̂_{(t+1)}^{(p)} do
        Generate β samples of the target size
        Calculate the particles for the next frame t + 1 according to Eq. 18
    end for
end if
```

In the next step, different patches from frame t are generated based on $z_t^{(i)}$. For each patch, a convolutional feature map is calculated using the CNN. Let $R^{(i)(j)} \in \mathbb{R}^{M \times Q}$ be the final correlation response map for particle i and target model $j, j = 1, ..., C_{t-1}$ (the generation of different target models in the previous frame is explained in the next section). M and Q are the length and width of the final correlation response map. These correlation response maps are computed by comparing the target models and the convolutional feature maps [5]. For each correlation response map, the likelihood or weight is calculated by [11]

$$\omega^{(i)(j)} = \sum_{m=1}^{M} \sum_{q=1}^{Q} R_{(m,q)}^{(i)(j)}, \tag{7}$$

Algorithm 2 Generate and evaluate initial particles

Input: Current frame, previous target state x_{t-1} and C_{t-1} target models $\mho_{t-1}^{(j)}$
Output: Current target state x_t, N_t particles $x_t^{(i)}$, their correlation response maps $R^{(i)(j)}$, their weights $\omega^{(i)(j)}$, maximum weight $\omega^{(i^*)(j^*)}$ and the best target model $\mho_{t-1}^*$

```
Calculate the predicted target state x̂_t according to Eq. 3 and Eq. 4
Generate N_t particles x_t^(i) around the predicted target state according to Eq. 5 and Eq. 6
for each particle x_t^(i) do
    for each of the C_{t-1} target models ℧_{t-1}^(j) do
        Generate the correlation response map R^(i)(j)
        Calculate its weight ω^(i)(j) according to Eq. 7
    end for
end for
Find the maximum weight ω^(i*)(j*) based on Eq. 8
Consider the particle corresponding to ω^(i*)(j*) as the final target state x_t
Consider the target model corresponding to ω^(i*)(j*) as the best model ℧*_{t-1}
```

In the first frame, after comparing the convolutional features with the ground truth

Algorithm 3 Update particles and remove redundant ones

Input: N_t particles $x_t^{(i)}$, their correlation response maps $R^{(i)(j)}$, their weights $\omega^{(i)(j)}$
Output: Remaining updated particles $\bar{x}_t^{(i)(j)}$

```
for each R^(i)(j) do
    Calculate its peak according to Eq. 9
    Update its state x_t^(i) to find x̄_t^(i)(j) using Eq. 10 to Eq. 12
end for
for every pair of particles do
    if Eq. 13 is satisfied then
        Remove the particle with lower weight
    end if
end for
```

model , we save the weight calculated from the correlation response map as a threshold T_w which is a reliable representative of the target because it is calculated based on the ground truth model. The location of the particle with the maximum weight is [11]

$$[i^*, j^*] = \arg\max_{i,j} \omega^{(i)(j)}. \tag{8}$$

The maximum weight is therfore $\omega^{(i^*)(j^*)}$, i.e., the weight corresponding to $[i^*, j^*]$. The target size corresponding to the maximum weight is then selected as the size of the bounding box for the current frame [11]. For the target position, the peak of the correlation response map with maximum weight is added to the corresponding particle location as discussed in [11]. Thus, the final target state x_t is calculated by comparing

Algorithm 4 Generate multiple target models

Input: Current frame, maximum weight $\omega^{(i^*)(j^*)}$, updated states $\bar{x}_t^{(i)(j)}$, their weights $\omega^{(i)(j)}$ and the best target model $\mho_{t-1}^*$

Output: C_t target models $\mho_t^{(j)}$ for the next frame $t+1$

Examine Eq. 20 to determine the high-likelihood states
Generate K_t current target models $\breve{\mho}_t^{(j)}$ based on the high-likelihood states
for Each $\breve{\mho}_t^{(j)}$ **do**
    if Eq. 15 is correct **then**
        Select the set with higher adjusting rates S_1
    else
        Select the set with lower adjusting rates S_2
    end if
    Generate Γ final target models $\mho_t^{(j)}$ for the next frame based on Eq. 21
end for

the convolutional features of particle i^* (the best particle) and the j^*th target model (the best target model). We define the best model as $\mho_{t-1}^*$, which is one of the C_{t-1} target models generated in frame $t-1$.

We then use the positions estimated in frame t as the locations of the new particles for frame $t+1$ as explained in Algorithm 3. The peak of the correlation response map of particle i compared with target model j is given by [11]

$$[\delta_u^{(i)(j)}, \delta_v^{(i)(j)}] = \underset{m,q}{\operatorname{argmax}}\, R_{(m,q)}^{(i)(j)}. \tag{9}$$

The target position corresponding to that particle and target model is then given by [11]

$$[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}] = [u_t^{(i)} + \delta_u^{(i)(j)}, v_t^{(i)} + \delta_v^{(i)(j)}], \tag{10}$$

where $[u_t^{(i)}, v_t^{(i)}]$ corresponds to the location of particle i according to Eq. 5. As seen in Eq. 9, for the location of each particle $x_t^{(i)}$, we estimate C_{t-1} target positions. The updated particle i compared with target model j in frame t is

$$\bar{x}_t^{(i)(j)} = \left[\bar{z}_t^{(i)(j)}, \dot{\bar{z}}_t^{(i)(j)}\right]^T, \tag{11}$$

where $\dot{\bar{z}}_t^{(i)(j)}$ is the updated version of $\dot{z}_t^{(i)}$ based on $[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}]$ and

$$\bar{z}_t^{(i)(j)} = \left[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}, h_t^{(i)}, w_t^{(i)}\right]^T. \tag{12}$$

$\bar{z}_t^{(i)(j)}$ is rounded because target positions and sizes are discrete quantities measured in pixels. After rounding, several $\bar{z}_t^{(i)(j)}$ may map to the same location. Since the initial particles can refine their locations for subsequent frames, these refined particles perform better than newly sampled particles in every frame. Their locations can also merge especially in simple frames to decrease the number of particles. Thus, the number of particles in simple frames is lower. For the target size, our tracker samples around each

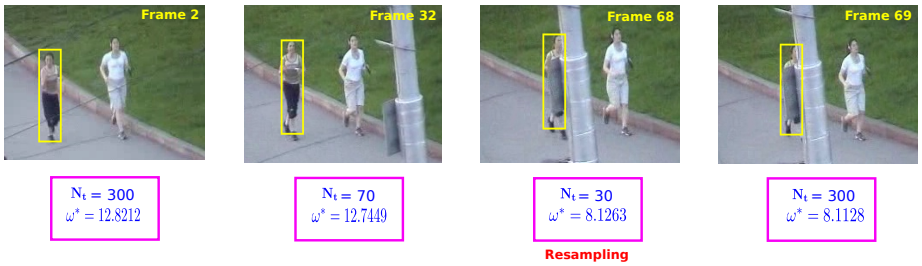

Fig. 2. Decreasing the number of the particles in simple frames and implementing resampling in difficult frames.

remaining particle based on its velocity. Consider two particles $\bar{x}_t^{(i')(j')}$ and $\bar{x}_t^{(i'')(j'')}$, if

$$[\tilde{u}_t^{(i')(j')}, \tilde{v}_t^{(i')(j')}] = [\tilde{u}_t^{(i'')(j'')}, \tilde{v}_t^{(i'')(j'')}], \tag{13}$$

and

$$\omega^{(i')(j')} > \omega^{(i'')(j'')}, \tag{14}$$

$\bar{x}_t^{(i')(j')}$ is selected and $\bar{x}_t^{(i'')(j'')}$ is removed, that is, if two particles converge to the same target position, the one with highest weight is selected and the other is removed. In the next step, the C_t target models are generated as discussed in the following section.

Two resampling conditions are then examined for frame $t+1$. The first condition is

$$\omega^{(i^*)(j^*)} > \varphi \times T_w. \tag{15}$$

As explained earlier, T_w is the maximum particle weight in the first frame. This maximum weight is calculated based on the comparison with the model generated by the ground truth in the same frame. When the maximum weight in frame t is less than T_w, it means our tracker could not produce a reliable correlation response map because of challenging scenarios such as occlusion. Therefore, the particles cannot properly refine their locations based on these weak correlation response maps. In these scenarios, we resample new particles. The second resampling condition is

$$N_t \cdot C_{t-1} - Z > T_t, \tag{16}$$

where T_t is the minimum number of particles to transfer to the next frame, and Z is the number of $\bar{x}_t$ which are removed. When too many particles converge to the same location and Z is too high, we increase the number of particles by resampling.

If resampling is not needed, we should predict new particles for the next frame based on the remaining particles in the current frame. Let $P = N_t \cdot C_{t-1} - Z$ be the number of remaining particles. We then predict the new particles for frame $t+1$ according to

$$\hat{x}_{t+1}^{(p)} = A\bar{x}_t^{(p)}, \tag{17}$$

where $p = 1, ..., (N_t \cdot C_{t-1} - Z)$ is the index of the remaining particles. We generate β samples to be added to the target size of each $\hat{x}_{(t+1)}^{(p)}$ according to

$$x_{t+1}^{(f)} = \hat{x}_{t+1}^{(p)} + [0_{(1,4)}, \zeta_{t+1}^{(f)}], \tag{18}$$

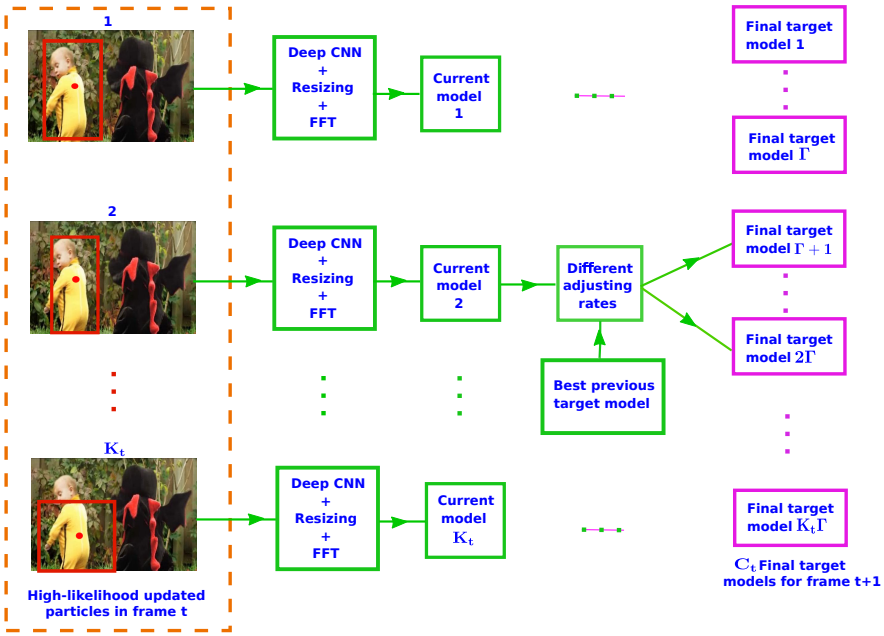

Fig. 3. Our proposed multiple models with different adjusting rates.

where $\zeta_{t+1}^{(f)} \in \mathbb{R}^4$ is drawn from a zero-mean normal distribution and $f = 1, ..., \beta$. Thus, the number of particles for frame $t+1$ is

$$N_{t+1} = \beta \cdot (N_t \cdot C_{t-1} - Z). \tag{19}$$

If the resampling conditions are not met, the selected target state x_t for frame t is applied in Eq. 3 to Eq. 6 to generate the new particles. Fig. 2 illustrates how the number of the particles decrease in simple frames. Additionally, the figure shows when the maximum weight significantly decreases comparing with T_w, resampling is implemented.

2.2 Multiple Correlation Models

In this section, we describe the process to generate C_t target models for frame t to be employed in frame $t+1$. A target model is generated by computing the Fourier transform of the convolutional features corresponding to an image patch. Fig. 3 illustrates our method for generating several target models. As discussed in [11], a comparison between the best model and the most accurate target size and position results in higher weights. Similar to [11], we select all high-likelihood target states by examining the following relation over all $N_t \times C_{t-1}$ weights

$$\omega^{(i)(j)} > \alpha\omega^*, \tag{20}$$

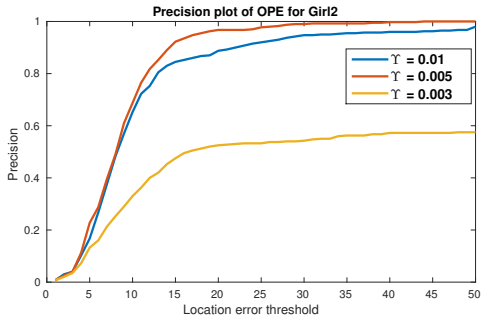

Fig. 4. The impact of different adjusting rates on the performance of DCPF2 [11] illustrates the benefit of generating multiple target models based on different adjusting rates

where we set α to 0.8. The particles that satisfy Eq. 20 are considered high-likelihood candidates [11]. A target model is generated based on each of the K_t selected high-likelihood particles in frame t, as discussed in [11]. Let $\breve{\mho}_t^{(j)}$ represent one of the K_t target models generated from the current frame. The final target models to be used in frame $t+1$ are a combination of the current target models and the previous selected target model according to [5]

$$\mho_t^{(j)} = (1-\Upsilon)\mho_{t-1}^{*} + \Upsilon\breve{\mho}_t^{(j)}, \tag{21}$$

where Υ is the adjusting rate. As seen in Fig. 4, the adjusting rate has a significant influence on the performance of correlation trackers. In our tracker, we consider different adjusting rates and then apply them to Eq. 21. Thus, the number of target models becomes

$$C_t = \Gamma K_t, \tag{22}$$

where Γ is the number of adjusting rates. We define two sets of adjusting rates S_1 and S_2. Eq. 15, determines which set should be used at each frame. When the tracker does not satisfy Eq. 15, the correlation response maps are not reliable because of challenging scenarios, and we should use lower adjusting rates to increase the effect of the previous target model and decrease the current one. When Eq. 15 is satisfied, we can use higher adjusting rates. Algorithm 4 explains the method of generating multiple target models.

3 Results and Discussion

We used the Visual Tracker Benchmark v1.1 beta (OTB100) to test our tracker performance. This benchmark contains 100 data sequences and considers 11 challenging scenarios. Results are based on a one-pass evaluation, which means the ground truth target size and position are used in the first frame to initialize the tracker. We select $N_t = 300$, $\varphi = 0.7$, $Tr2 = 4$, $\beta = 5$, $\Gamma = 3$, $S_1 = [0.0075, 0.01, 0.015]$ and $S_2 = [0.0075, 0.005, 0.001]$. The number of adjusting rates in each set is limited to three because of the computation costs. However, these sets are defined based on experiments. The precision and success criteria are explained in [14].

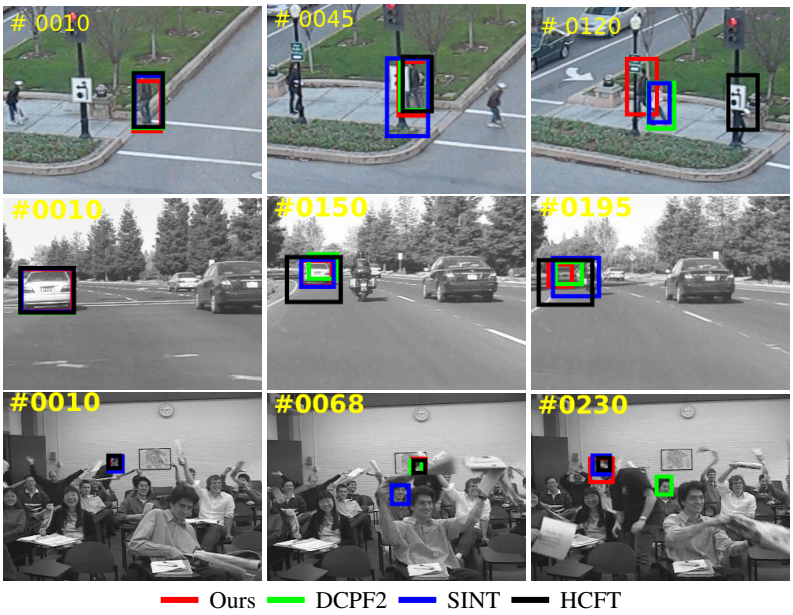

Fig. 5. Qualitative evaluation of our tracker, *DCPF2*, *SINT* and *HCFT* on three challenging sequences (from top to bottom *Human3*, *Car1* and *Freeman4*, respectively).

Fig. 5 illustrates the qualitative evaluation of our tracker compared to DCPF2, SINT [16] and HCFT. In the first data sequence *Human3*, the lower adjusting rate helps our tracker to rely more on the previous target model. In the second data sequence *Car1*, Our tracker improves the estimated target position. Additionally, it is able to better handle the occlusion in the third data sequence *Freeman4*.

We compared our tracker with eight state of the art trackers including: CFNet-conv3 [17], SiameseFC [18], SINT, LCT [19] and CNN-SVM [20], HDT, HCFT and DCPF2. As illustrated in Fig. 6, our overall performance in terms of precision and success are improved by 3.5% and 5.5% in comparison with DCPF2, which is the second and fourth-best tracker in the precision and success plots, respectively. Our tracker outperforms SINT, the second-best tracker in the success plot, by around 4.5%. On deformation and occlusion, our performance is better because we employ different adjusting rates and decrease the updating rate of the model in challenging frames. As seen in Fig. 6, for deformation and occlusion, our performance is improved around 4.5% in precision and 6% in success in comparison with the second-best tracker. For other challenging scenarios such as motion blur, background clutter and out of plane rotation, our tracker shows improvements of approximately 5%, 3.5% and 3%, respectively, in comparison to the second-best tracker.

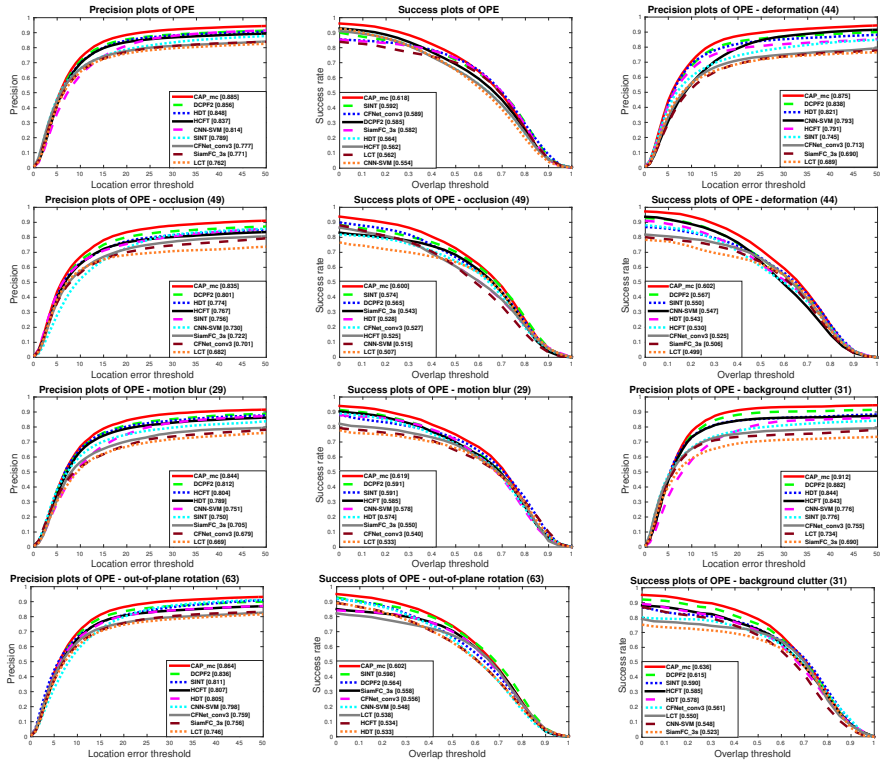

Fig. 6. Quantitative evaluation of our tracker in comparison with state-of-the-art trackers on OPE.

4 Conclusion

This article proposes a novel framework for visual tracking based on the integration of an adaptive particle filter, a deep CNN, and a correlation filter. In our adaptive particle filter, the locations of the updated particle in the current frame are used as particles for the next frame, which provides more reliable particles than sampling around the final estimated target state in every frame. Our adaptive particle filter can decrease the number of particles especially in simple frames because the particles can converge together. If the number of particles is too small or the maximum weight in the current frame is significantly lower than the weight in the first frame, resampling is performed. The reason for using the weight in the first frame as a threshold for resampling is that it is computed based on the ground truth target model, and hence serves as an upper bound on the weight of subsequent particles. Additionally, we generate multiple target models in each frame by applying different adjusting rates to the models created by the high-likelihood particles. For challenging frames, we use lower adjusting rates, which means we rely more on previous target models. The Visual Tracker Benchmark v1.1 beta (OTB100) is used for evaluating the proposed tracker's performance. The results show that our tracker outperforms several state-of-the-art methods.

References

1. J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, "Visual tracking using attention-modulated disintegration and integration," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
2. M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3038–3046.
3. M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, December 2015.
4. M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.
5. C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, December 2015.
6. J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.
7. J. Zhang, S. Ma, and S. Sclaroff, "Robust tracking via multiple experts," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 188–203.
8. S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2011, pp. 263–270.
9. W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.
10. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.
11. R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Deep convolutional particle filter with adaptive correlation maps for visual tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2018.
12. R. J. Mozhdehi and H. Medeiros, "Deep convolutional particle filter for visual tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2017.
13. H. Y. Cheng and J. N. Hwang, "Adaptive particle sampling and adaptive appearance for multiple video object tracking," *Signal Processing*, vol. 89, no. 9, pp. 1844–1849, 2009.
14. Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
15. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
16. R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
17. J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
18. L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV 2016 Workshops*, 2016, pp. 850–865.
19. C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5388–5396.
20. S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *32nd International Conference on Machine Learning*, 2015.