IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 5, 2019, accepted May 2, 2019, date of publication May 7, 2019, date of current version May 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2915381

Innovative Target Tracking Method Combined Adaptive α-β Filter With Robust BPNN

BO LI
School of Electronics and Information Engineering, Liaoning University of Technology, Jinzhou 121001, China
e-mail: leeboo@yeah.net

This work was supported in part by the National Natural Science Foundation of China under Grant 51679116, in part by the Program for Liaoning Innovative Talents in University under Grant LR2017068, and in part by the Program for Liaoning Innovative Research Team in University under Grant LT2016006.

ABSTRACT Target tracking is popular in computer vision field. Although the classic BPNN completes targets tracking, its computation is complex and tracking accuracy is low when the tracking scene is uncertain or complex. To deal with the difficulties above, in this paper, we propose an innovative target tracking method combined adaptive α–β filter with robust BPNN. First, we utilize the adaptive α–β filter to compute the location region on optimal filtering parameters in the prediction stage. Of course, the novel filter reduces the region and gives effective image information to the robust BPNN that has the optimal number and weight of neurons as well as the improved learning rate. Subsequently, the network makes an accurate recognition and sends back the updated positions of targets to the filter for the next cycle. Employing the novel interactive mechanism, the numerical study and experiments indicate that the proposed method has remarkable improvement on average performance in the uncertain and complex environment.

INDEX TERMS Target tracking, $\alpha - \beta$ filter, BPNN, weight, target dynamics.

I. INTRODUCTION

As a popular research topic in computer vision field, the target tracking has significant applications on navigation, intelligent traffic, military and etc. In the past decades, many studies on this topic have been completed. First, the Kalman filter (KF) is used to track mobile targets. Aimed at the limitation on linear component, [1] proposed a new hierarchical model based on the contextual knowledge and multiple cues into the unscented KF (UKF). Of course, the particle filter (PF) is another method though the real-time is relatively poor. In [2], a robust and unconstrained tracking method was developed to overcome tracking failure issues with the PF. Nowadays, the kernelized correlation filter (KCF) is widely utilized. In [3], a shape-preserved KCF was proposed to accommodate target shape information for robust tracking. Subsequently, the historical template and the KCF were integrated to track the target and its position in [4]. In [5], a new visual tracking framework of discriminant KCF based on adaptive template update strategy was presented. Meanwhile, the methods cannot be well applied in some uncertain and complex scenes owing to technical restraint [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

Modeling on the architecture of human brain, the neural network (NN) is regarded as a set of connected neurons that can process signals through the network and generate the desired output. Especially, the back propagation NN (BPNN) computes the difference between the desired output and the actual output. When an input vector is presented to the network, it is propagated forward layer by layer, until it reaches the output layer. At this time, the actual output is compared with the desired output based on the loss function. The resulting error of BPNN is calculated for each neuron on the output layer and is propagated through the network, until the neuron has its error. With flexible property and learning ability, the BPNN is used to model the unknown and nonlinear relationship between multiple inputs and multiple outputs. Due to its advances, the BPNN has been used to track mobile targets. In [7], a new method of image recognition on feature extraction under the blurred backgrounds was proposed. Aiming at the problem that the existing target tracking framework based on deep learning is difficult to realize the real-time video target tracking on low-power mobile surveillance system, [8] put forward an improved multi-target tracking method. According to the Lyapunov function, a BPNN method was presented for nonlinear pure-feedback system,

where all closed-loop signals were uniformly bounded in [9]. Subsequently, [10] proposed a BPNN scheme for quarter-vehicle mode, and the network was used to approximate the unknown mass of vehicle-body. Although the BPNN completes target tracking, the computational cost is high in the uncertain and complex scene. We should reduce the cost of classic BPNN and improve its accuracy.

As for the classic BPNN, the whole image assigned to a given target brings about extra running time. Nevertheless, the filtering algorithm extrapolates target dynamics. It generates and updates the tracks based on the available measurement. In [11], an adaptive BPNN was proposed for suspension system, and then a dynamic surface control technique was developed to stabilize the attitude of vehicles by using a first-order filter. By comparison, the α-β filter, a second-order filter, presumes that the complex system is approximated under the dynamic model, which computes steady-state solutions with the exponentially reduced computation. However, it is difficult to adaptively define the filtering gains for maneuvering targets. To overcome it, a new α-β filter on the fuzzy method was discussed to compute filtering coefficients in [12]. In [13], an adaptive α -β filter was applied in the radar tracking system. Reference [14] designed a novel α -β filter and come to the satisfactory conclusion under the noisy condition. If possible, the target with the variable acceleration (VA) motion should be tracked by using it. Consequently, the filter-auxiliary tracking this kind of target with low computational load is our goal. In this work, we define the relationship between current measurement information and target dynamics. The adaptive coefficient is put forward to adapt to the target maneuverability in an optimal ellipse region. Subsequently, the BPNN utilizes the resulting errors to calculate gradient. It is fed to the optimization method, which in turn uses it to update the weight of each neuron, in an attempt to minimize the value of loss function. Inevitably, the weight output delta and input activation make the important role. Considering the tracking reliability of the classic BPNN, we introduce the inertia on initial weights to improve robustness, where the inertia makes the current weight change under the error function. Simultaneously, the problems of getting stuck are avoided, and the gradient on the error function becomes small in a flat plateau that immediately leads to a deceleration of gradient descent. The deceleration is delayed by the addition of inertia so that the plateau is escaped. Besides, the learning rate is improved based on the change of gradient direction.

Inspired by technical specifications for target tracking, the paper presents an innovative method combined adaptive α -β filter with robust BPNN. It mainly addresses the efficiency and reliability in some scenes, and distinctly exhibits a novel mechanism. The interactive tracking framework on the adaptive α -β filter and the robust BPNN are mutually adjusted to reduce tracking error and get optimal region when the motion state is maneuvering. For example, in the prediction stage, the adaptive α-β filter reduces the required location region by using the filtering parameters and target dimension. Within this region, the network makes an effective recognition and sents the updated positions of targets to the adaptive filter for the next calculation. As a result, the robustness and accuracy of the proposed method are improved inherently.

The remainder of this paper is assigned as follows: In Section 2, the principle of the adaptive α -β filter is formulated and the characteristics of two kinds of sensors are discussed to define the square location region. In Section 3, we explore the robust BPNN with the weight optimization, learning rate improvement and model construction. What's more, the process of the proposed method is presented at length. The numerical study and some scenarios are indicated with the promising results to verify the tracking performance of the proposed method in Section 4. Further, it achieves mobile targets tracking under the different environments. In Section 5, the conclusions are drawn by providing the next research plan.

II. FILTER MODEL

We have in hand the predicted vector of target in the state space $\mathcal{X}$ at time $k-1$ [9], [15], [16]:

$$\mathbf{X}_{k|k-1} = \mathbf{F}_{k|k-1}\mathbf{X}_{k-1|k-1} + \mathbf{\Gamma}_k\mathbf{U}_k \tag{1}$$

where $\mathbf{F}_{k|k-1}$ is the state transfer matrix of target center, $\mathbf{X}_{k-1|k-1}$ is the target state vector at time $k-1$, $\mathbf{\Gamma}_k$ is the gain matrix of state noise vector $\mathbf{U}_k$ that follows the Gaussian distribution $\mathcal{N}(0, \mathbf{Q}_k)$, and $\mathbf{Q}_k$ is the variance.

The measurement model in the measurement space $\mathcal{Z}$ at time k can be formulated as:

$$\mathbf{Z}_k = \mathbf{H}_k\mathbf{X}_{k|k-1} + \mathbf{V}_k \tag{2}$$

where $\mathbf{H}_k$ is the measurement transfer matrix, $\mathbf{V}_k$ is the measurement noise vector with the Gaussian distribution $\mathcal{N}(0, \mathbf{R}_k)$, and $\mathbf{R}_k$ is the variance.

A. PRINCIPLE OF α-β FILTER

In general, the classic α-β filter excludes measurement error as much as possible when gathering the target dynamics. Suppose that the scanning period is$T$, we have the predicted equation of target state as follows [17], [18]:

$$\begin{cases} \mathbf{x}_{k|k-1} = \mathbf{x}_{k-1|k-1} + \dot{\mathbf{x}}_{k-1|k-1}T \\ \dot{\mathbf{x}}_{k|k-1} = \dot{\mathbf{x}}_{k-1|k-1} \end{cases} \tag{3}$$

where $\mathbf{x}_{k|k-1}$ and $\dot{\mathbf{x}}_{k|k-1}$ are the predicted position and velocity vector respectively at time $k-1$.

Subsequently, the updated equation of target state at time k can be written as:

$$\begin{cases} \mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \alpha_k\left(\mathbf{Z}_k - \mathbf{H}_k\mathbf{x}_{k|k-1}\right) \\ \dot{\mathbf{x}}_{k|k} = \dot{\mathbf{x}}_{k|k-1} + \dfrac{\beta_k\left(\mathbf{Z}_k - \mathbf{H}_k\mathbf{x}_{k|k-1}\right)}{T} \end{cases} \tag{4}$$

where α_k and β_k are the filtering coefficients corresponding to the position and velocity. As for the target with linear

dynamics, they are often defined by [14], [19]:

$$\begin{cases} \alpha_k = \dfrac{2\,(2k-1)}{k\,(k+1)} \\ \beta_k = \dfrac{6}{k\,(k+1)} \end{cases} \tag{5}$$

Due to the target maneuverability and noise interference, the accuracy becomes poor.

B. ADAPTIVE α-β FILTER

To overcome the drawbacks of the classic α-β filter, an adaptive filter for general target has presented. As for the extended dynamics, the pseudo-acceleration is added to the prediction equation for tracking the non-maneuvering and maneuvering targets. In practice, the motion state of target is complex. It easily changes from the non-maneuvering state into the maneuvering state. The constant acceleration (CA) motion model should be constructed for maneuvering target. Therefore, we extend (3) as follows:

$$\begin{cases} \mathbf{x}_{k|k-1} = \mathbf{x}_{k-1|k-1} + \dot{\mathbf{x}}_{k-1|k-1}T + \dfrac{1}{2}aT^2 \\ \dot{\mathbf{x}}_{k|k-1} = \dot{\mathbf{x}}_{k-1|k-1} + aT \end{cases} \tag{6}$$

where a is the acceleration of target:

$$a = \frac{\dot{\mathbf{x}}_{k-1|k-1} - \dot{\mathbf{x}}_{k-2|k-2}}{T} \tag{7}$$

Substituting (7) into (6), we have:

$$\begin{cases} \mathbf{x}_{k|k-1} = \mathbf{x}_{k-1|k-1} + \dfrac{1}{2}\left(3\dot{\mathbf{x}}_{k-1|k-1} - \dot{\mathbf{x}}_{k-2|k-2}\right)T \\ \dot{\mathbf{x}}_{k|k-1} = \dot{\mathbf{x}}_{k-1|k-1} + \dot{\mathbf{x}}_{k-1|k-1} - \dot{\mathbf{x}}_{k-2|k-2} \end{cases} \tag{8}$$

When T is small, the process noise is regarded as a constant. At this time, we have the product of T^2 and the standard deviation ratio between $\mathbf{U}_k$ and $\mathbf{V}_k$:

$$\lambda_k = T^2\sqrt{\mathbf{Q}_k\mathbf{R}_k^{-1}} \tag{9}$$

where $\mathbf{Q}_k$ is defined by:

$$\mathbf{Q}_k = \sqrt{\frac{\sum_{k=1}^{K}\left(\mathbf{x}_{k|k} - \mathbf{x}_{k|k-1}\right)^2}{K}} \tag{10}$$

Note that the stronger the maneuverability of target, the greater the value of $\left(\mathbf{x}_{k|k} - \mathbf{x}_{k|k-1}\right)^2$. Then, we get the solutions in the limited time:

$$\begin{cases} \lambda_k = \dfrac{\beta_k}{\sqrt{1-\alpha_k}} \\ \beta_k = 2\,(2-\alpha_k) - 4\sqrt{1-\alpha_k} \end{cases} \tag{11}$$

After solving (11), we have:

$$\begin{cases} \alpha_k = \dfrac{\sqrt{\lambda_k\,(\lambda_k+8)}\left((\lambda_k+4) - \sqrt{\lambda_k\,(\lambda_k+8)}\right)}{8} \\ \beta_k = \dfrac{\lambda_k\left((\lambda_k+4) - \sqrt{\lambda_k\,(\lambda_k+8)}\right)}{4} \end{cases} \tag{12}$$

where$\alpha_k \in [0,1]$ and $\beta_k \in [0,2]$ under the condition of $\lambda_k \in [0,\infty)$, and α_k represents the trust level between the measurement position and the predicted position. Similarly, β_k describes the trust level on the between the measurement velocity and the predicted velocity.

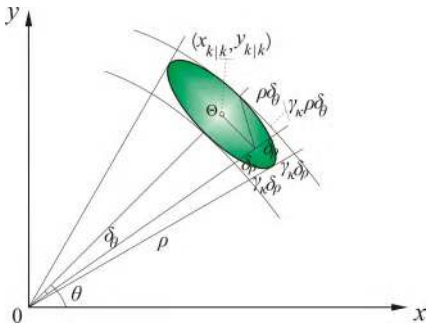


FIGURE 1. Ellipse estimation region.

With respect to the VA motion model, we introduce the adaptive adjustment coefficient:

$$\gamma_\kappa = \sqrt{\frac{2(2\kappa+1)}{\kappa(\kappa-1)}} \tag{13}$$

where the index κ is taken to an integer that is greater than 2. With the increase of κ, γ_κ becomes convergent and the target keeps stable dynamics. Therefore, we adjust κ with the increment $\Delta\kappa$ when the target dynamics is unstable:

$$\Delta\kappa = \text{round}\left(\kappa\left(\frac{\mathbf{x}_{k|k} - \mathbf{x}_{k|k-1}}{\gamma_\kappa\sqrt{\delta_\rho^2 + \rho^2\delta_\theta^2}} - \varepsilon\right)\right) \tag{14}$$

where round $(\cdot)$ denotes the nearest integer, ρ is the radius of the location region Θ, ε is the threshold of position change rate, δ_ρ and δ_θ are the errors of range and azimuth respectively. There is $\kappa \leftarrow \kappa + 1$ under the condition of $\frac{\mathbf{x}_{k|k} - \mathbf{x}_{k|k-1}}{\gamma_\kappa\sqrt{\delta_\rho^2+\rho^2\delta_\theta^2}} \geq \varepsilon$. Otherwise, we have $\kappa \leftarrow \kappa - \Delta\kappa$.

Let $\gamma_\kappa\delta_\rho$ and $\gamma_\kappa\rho\delta_\theta$ be the spans of range and azimuth. In Figure 1, an ellipse is applied to estimate target position in the Cartesian coordinate system $\mathbf{x}_{k|k} = \left(x_{k|k}, y_{k|k}\right)$:

$$\frac{\left(\left(y - y_{k|k}\right)\sin\theta + \left(x - x_{k|k}\right)\cos\theta\right)^2}{\gamma_\kappa^2\delta_\rho^2} + \frac{\left(\left(y - y_{k|k}\right)\cos\theta - \left(x - x_{k|k}\right)\sin\theta\right)^2}{\gamma_\kappa^2\rho^2\delta_\theta^2} = 1 \tag{15}$$

Note that the computational complexity is saved when estimating target position in Θ.

Proof 1: Recalling (15), we compute the area of Θ:

$$S_\Theta = \pi\gamma_\kappa^2\rho\delta_\rho\delta_\theta \tag{16}$$

The area ratio of total region to location region is:

$$\frac{S}{S_\Theta} = \frac{\left(\rho + 2\gamma_\kappa\delta_\rho\right)\gamma\rho\delta_\theta}{\pi\gamma_\kappa^2\rho\delta_\rho\delta_\theta} = \frac{1}{\pi}\left(\frac{\rho}{\gamma_\kappa\delta_\rho} + 2\right) \tag{17}$$

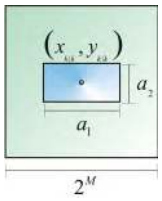


FIGURE 2. Optimal square region for actual target.

Due to $\rho \gg \gamma_\kappa \delta_\rho$, the actual estimation region is reduced and the computational cost is saved when using the target center (i.e., the geometrical center of target) in Θ.

C. OTHER PRELIMINARIES

Given that the actual target has $a_1 \times a_2$ pixel ($a_1 \geq a_2$) in the image, the required location region should be further optimized in a given square, where the minimal M meets the condition $a_1 \leq 2^M$ in Figure 2. Then, the target is fixed based on its size. Since the region is input to the BPNN, we use the square of $2^M \times 2^M$ pixel for convenience.

Remark 1: Although Figure 2 shows the appearance region of actual target, it is usually changed in the whole image. As a result, the actual target position should be further considered in double cases. First, the fixed sensor tracks multi-target in a certain region, that is, the targets are immediately found when they come into the region. It uses the fixed background and the absolute displacements of targets in the current image. On the other hand, the mobile sensor tracks multi-target with a certain velocity, that is, the targets cannot disappear when the velocity of mobile sensor is high. It applies the different background and relative displacements of targets between the current image and the previous images.

III. BP NEURAL NETWORK

A. PRINCIPLE OF BPNN

The BPNN with above one hidden layer has the faster training speed. We have in hand a general BPNN with C ($C \geq 3$) layers for p targets, which includes 1 input layer, $(C-2)$ hidden layers and 1 output layer. At time k, there are q_c ($q_c = 1, \cdots, Q_c, c = 1, \cdots, C$) neurons $\ell_{p,k}^{(c,q_c)}$ on each layer. Then, the neurons for the p^{th} target in this network are given by [20], [21]:

$$\begin{cases} \ell_{p,k}^{(c,q_c)} = \left(\ell_{p,k}^{(1,1)}, \ell_{p,k}^{(1,2)}, \cdots, \ell_{p,k}^{(1,Q_c)}\right)^{\mathrm{T}} \\ c = 1 \\ \ell_{p,k}^{(c,q_c)} = f\left(\sum\limits_{q_{c-1}=1}^{Q_{c-1}} w_{p,k}^{(c-1,q_{c-1})} \ell_{p,k}^{(c-1,q_{c-1})} + b_{p,k}^{(c-1)}\right)^{\mathrm{T}} \\ c = 2, \cdots, C \end{cases} \tag{18}$$

where $w_{p,k}^{(c-1,q_{c-1})}$ and $b_{p,k}^{(c-1)}$ are the weight and threshold between two adjacent layers, and $f(\cdot)$ is the excitation function. Usually, the Sigmoid function is applied, which can compress the input into an output in [0, 1].

In view of multi-class problem with p training targets, the squared error loss function on the output layer is:

$$E_k = \frac{1}{2} \sum_{p=1}^{P} \sum_{q_C=1}^{Q_C} \left(d_{p,k}^{(C,q_C)} - \ell_{p,k}^{(C,q_C)}\right)^2 \tag{19}$$

Note that the target is organized as a $1/p$ model, where the elements $d_{p,k}^{(C,q_C)}$ are positive when $\ell_{p,k}^{(C,q_C)}$ belongs to the p^{th} target, and the rest elements are zero or negative. Since E_k is the sum of errors of all targets, we only compute the back propagation on the single target p [22]–[24]:

$$E_{p,k} = \frac{1}{2} \sum_{q_C=1}^{Q_C} \left(d_{p,k}^{(C,q_C)} - \ell_{p,k}^{(C,q_C)}\right)^2 \tag{20}$$

Remark 2: The BPNN sends the error back to train all weights that are trained on the minimal sum of squared errors. Due to the dynamic change in the huge number of images and the parameters in the learning and training stages, the classic BPNN cannot keep rapid convergence when it is far away from the local minimum. Therefore, the weight assigned to a given neuron should be modified.

B. ROBUST BPNN

1) WEIGHT OPTIMIZATION

Suppose that ς is the inertia and η_{k-1} is the learning rate coefficient, the weight assigned to the q_c^{th} neuron on the c^{th} layer for the p^{th} target at time k is given by:

$$w_{p,k}^{(c,q_c)} = w_{p,k-1}^{(c,q_c)} + \varsigma\left(w_{p,k}^{(c,q_c)} - w_{p,k-1}^{(c,q_c)}\right) - \eta_{p,k-1} \frac{\partial E_{p,k-1}^{(C,q_c)}}{\partial w_{p,k-1}^{(c,q_c)}} \tag{21}$$

Note that the sum of them plays a positive role on $w_{p,k}^{(c,q_c)}$, and the weight is adjusted on the current gradient and inertia. We rewrite (21) into the recursion form:

$$\begin{cases} w_{p,k}^{(c,q_c)} - w_{p,k-1}^{(c,q_c)} = \varsigma\left(w_{p,k}^{(c,q_c)} - w_{p,k-1}^{(c,q_c)}\right) - \eta_{p,k-1} \dfrac{\partial E_{p,k-1}^{(C,q_c)}}{\partial w_{p,k-1}^{(c,q_c)}} \\ w_{p,k-1}^{(c,q_c)} - w_{p,k-2}^{(c,q_c)} = \varsigma\left(w_{p,k-1}^{(c,q_c)} - w_{p,k-2}^{(c,q_c)}\right) - \eta_{p,k-2} \dfrac{\partial E_{p,k-2}^{(C,q_c)}}{\partial w_{p,k-2}^{(c,q_c)}} \\ \vdots \\ w_{p,1}^{(c,q_c)} - w_{p,0}^{(c,q_c)} = -\eta_{p,0} \dfrac{\partial E_{p,0}^{(C,q_c)}}{\partial w_{p,0}^{(c,q_c)}} \end{cases} \tag{22}$$

After adding the equations above, we have:

$$\begin{aligned} w_{p,k}^{(c,q_c)} &= \varsigma w_{p,k-1}^{(c,q_c)} + (1-\varsigma) w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-1} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \\ &= \varsigma\left(\varsigma w_{p,k-2}^{(c,q_c)} + (1-\varsigma) w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-2} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}}\right) \end{aligned}$$

$$
\begin{aligned}
&+ (1-\varsigma)\, w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-1} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \\
&= \varsigma^2 w_{p,k-2}^{(c,q_c)} + \left(1-\varsigma^2\right) w_{p,0}^{(c,q_c)} \\
&\quad - \left(\sum_{i=0}^{k-1} \eta_i \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} + \varsigma \sum_{i=0}^{k-2} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \right) \\
&\quad \vdots \\
&= \varsigma^2 w_{p,0}^{(c,q_c)} + \left(1-\varsigma^2\right) w_{p,0}^{(c,q_c)} \\
&\quad - \left(\sum_{i=0}^{k-1} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} + \varsigma \sum_{i=0}^{k-2} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \right. \\
&\quad \left. + \cdots + \varsigma^{k-1} \sum_{i=0}^{0} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \right) \\
&= w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \qquad (23)
\end{aligned}
$$

where $w_{p,k}^{(c,q_c)}$ is determined by the initial weight and the sum of gradients at the previous time on the right-hand side. It reduces weight error and accelerates convergence rate.

Recalling the C-layer BPNN, we compute the weight assigned to the q_C^{th} neuron on the output layer:

$$
\begin{aligned}
&w_{p,k}^{(C,q_C)} \\
&= w_{p,0}^{(C,q_C)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial w_{p,i}^{(C,q_C)}} \\
&= w_{p,0}^{(C,q_C)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C,q_C)}} \frac{\partial \ell_{p,i}^{(C,q_C)}}{\partial \tilde{\ell}_{p,i}^{(C,q_C)}} \frac{\partial \tilde{\ell}_{p,i}^{(C,q_C)}}{\partial w_{p,i}^{(C,q_C)}} \\
&= w_{p,0}^{(C,q_C)} + \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \sum_{q_C=1}^{Q_C} \delta_{p,i}^{(C)} \qquad (24)
\end{aligned}
$$

where $\delta_{p,i}^{(C)}$ is defined as:

$$
\delta_{p,i}^{(C)} = \left(d_{p,i}^{(C,q_C)} - \ell_{p,i}^{(C,q_C)}\right) \left(1-\ell_{p,i}^{(C,q_C)}\right) \ell_{p,i}^{(C,q_C)} \ell_{p,i}^{(C-1,q_{C-1})} \qquad (25)
$$

As for the weight assigned to the q_{C-1}^{th} neuron on the last hidden layer, there is:

$$
\begin{aligned}
&w_{p,k}^{(C-1,q_{C-1})} \\
&= w_{p,0}^{(C-1,q_{C-1})} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial w_{p,i}^{(C-1,q_{C-1})}} \\
&= w_{p,0}^{(C-1,q_{C-1})} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C,q_C)}} \frac{\partial \ell_{p,i}^{(C,q_C)}}{\partial \tilde{\ell}_{p,i}^{(C,q_C)}} \\
&\quad \times \frac{\partial \tilde{\ell}_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C-1,q_{C-1})}} \frac{\partial \ell_{p,i}^{(C-1,q_{C-1})}}{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}} \frac{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}}{\partial w_{p,i}^{(C-1,q_{C-1})}} \\
&= w_{p,0}^{(C-1,q_{C-1})} + \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \sum_{q_{C-1}=1}^{Q_{C-1}} \delta_{p,i}^{(C-1)} \qquad (26)
\end{aligned}
$$

where $\delta_{p,i}^{(C-1)}$ is defined as:

$$
\begin{aligned}
&\delta_{p,i}^{(C-1)} \\
&= \left(\sum_{q_C=1}^{Q_C} \left(d_{p,i}^{(C,q_C)} - \ell_{p,i}^{(C,q_C)}\right) \left(1-\ell_{p,i}^{(C,q_C)}\right) \ell_{p,i}^{(C,q_C)} w_{p,0}^{(C,q_C)} \right) \\
&\quad \times \left(1-\ell_{p,i}^{(C-1,q_{C-1})}\right) \ell_{p,i}^{(C-1,q_{C-1})} \ell_{p,i}^{(C-2,q_{C-2})} \\
&= \left(1-\ell_{p,i}^{(C-1,q_{C-1})}\right) \ell_{p,i}^{(C-2,q_{C-2})} \sum_{q_C=1}^{Q_C} \delta_{p,i}^{(C)} w_{p,0}^{(C,q_C)} \qquad (27)
\end{aligned}
$$

Similarly, the weight assigned to the q_1^{th} neuron on the 1$^{\text{st}}$ hidden layer is:

$$
\begin{aligned}
w_{p,k}^{(2,q_2)} &= w_{p,0}^{(2,q_2)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial w_{p,i}^{(2,q_2)}} \\
&= w_{p,0}^{(2,q_2)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C,q_C)}} \frac{\partial \ell_{p,i}^{(C,q_C)}}{\partial \tilde{\ell}_{p,i}^{(C,q_C)}} \\
&\quad \times \frac{\partial \tilde{\ell}_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C-1,q_{C-1})}} \frac{\partial \ell_{p,i}^{(C-1,q_{C-1})}}{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}} \\
&\quad \times \frac{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}}{\partial \ell_{p,i}^{(C-2,q_{C-2})}} \cdots \frac{\partial \ell_{p,i}^{(2,q_2)}}{\partial \tilde{\ell}_{p,i}^{(2,q_2)}} \frac{\partial \tilde{\ell}_{p,i}^{(2,q_2)}}{\partial w_{p,i}^{(2,q_2)}} \\
&= w_{p,0}^{(2,q_2)} + \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \sum_{q_2=1}^{Q_2} \delta_{p,i}^{(2)} \qquad (28)
\end{aligned}
$$

where $\delta_{p,i}^{(2)}$ is defined as: From the recursion above, we can derive the presentation on each weight in the C-layer BPNN with Lemma 1.

Lemma 1: The weight $w_{p,k}^{(c,q_c)}$ assigned to the q_c^{th} neuron on the c^{th} $(c = 2, \cdots, C)$ layer at time k can be defined as:

$$
\begin{aligned}
w_{p,k}^{(c,q_c)} &= w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial w_{p,i}^{(c,q_c)}} \\
&= w_{p,0}^{(c,q_c)} - \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \frac{\partial E_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C,q_C)}} \frac{\partial \ell_{p,i}^{(C,q_C)}}{\partial \tilde{\ell}_{p,i}^{(C,q_C)}} \\
&\quad \times \frac{\partial \tilde{\ell}_{p,i}^{(C,q_C)}}{\partial \ell_{p,i}^{(C-1,q_{C-1})}} \frac{\partial \ell_{p,i}^{(C-1,q_{C-1})}}{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}} \\
&\quad \times \frac{\partial \tilde{\ell}_{p,i}^{(C-1,q_{C-1})}}{\partial \ell_{p,i}^{(C-2,q_{C-2})}} \cdots \frac{\partial \ell_{p,i}^{(c+1,q_{c+1})}}{\partial \tilde{\ell}_{p,i}^{(c,q_c)}} \frac{\partial \tilde{\ell}_{p,i}^{(c,q_c)}}{\partial w_{p,i}^{(c,q_c)}} \\
&= w_{p,0}^{(c,q_c)} + \sum_{i=0}^{k-1} \sum_{j=1}^{k-i} \varsigma^{k-i-j} \eta_{p,i} \sum_{q_c=1}^{Q_c} \delta_{p,i}^{(c)} \qquad (30)
\end{aligned}
$$

$$\begin{aligned}\delta_{p,i}^{(2)} &= \left(\sum_{q_3=1}^{Q_3} \cdots \left(\sum_{q_{C-1}=1}^{Q_{C-1}} \left(\left(\sum_{q_C=1}^{Q_C} \left(\begin{pmatrix} d_{p,i}^{(C,q_C)} - \ell_{p,i}^{(C,q_C)} \\ \times \left(1-\ell_{p,i}^{(C,q_C)}\right) \ell_{p,i}^{(C,q_C)} w_{p,0}^{(C,q_C)} \end{pmatrix} \right) \right) \right) \right) \\ &\quad \times \left(1-\ell_{p,i}^{(3,q_3)}\right) \ell_{p,i}^{(3,q_3)} w_{p,0}^{(3,q_3)} \\ &\quad \times \left(1-\ell_{p,i}^{(2,q_2)}\right) \ell_{p,i}^{(2,q_2)} \ell_{p,i}^{(1,q_1)} \\ &= \left(1-\ell_{p,i}^{(2,q_2)}\right) \ell_{p,i}^{(1,q_1)} \sum_{q_3=1}^{Q_3} \delta_{p,i}^{(3)} w_{p,0}^{(3,q_3)} \end{aligned} \tag{29}$$

where $\delta_{p,i}^{(c)}$ is given by:

$$\begin{aligned}\delta_{p,i}^{(c)} &= \left(1-\ell_{p,i}^{(c,q_c)}\right) \ell_{p,i}^{(c-1,q_{c-1})} \\ &\quad \times \begin{cases} \sum_{q_{c+1}=1}^{Q_c} \delta_{p,i}^{(c+1,)} w_{p,0}^{(c+1,q_{c+1})}, & c = 2, \cdots, C-1 \\ \left(d_{p,i}^{(c,q_c)} - \ell_{p,i}^{(c,q_c)}\right) \ell_{p,i}^{(c,q_c)}, & c = C \end{cases} \end{aligned} \tag{31}$$

Note that the weights on each hidden layer have the similar expression on initial weights of neurons. For comparison, the weight assigned to a given neuron on the output layer is only dependence of initial values on the layer. We can only modify them with the sum of gradients at previous times to compute the current weight.

2) LEANING RATE IMPROVEMENT

In view of the learning rate, the proposed BPNN should be adaptively adjusted based on the change of gradient direction. When the gradient directions of two successive iterations are the same, the learning rate $\eta_{p,k}$ should be increased, which indicates that the current descent is slow. When the gradient directions of successive iterations are opposite, $\eta_{p,k}$ should be reduced, which indicates that the current descent is fast. There is:

$$\eta_{p,k} = \begin{cases} \zeta_p \eta_{p,k-1}, & \text{opposite gradient directions } (\zeta_p \geq 1) \\ \xi_p \eta_{p,k-1}, & \text{same gradient directions } (0 < \xi_p < 1) \\ \eta_{p,k-1}, & \text{else} \end{cases} \tag{32}$$

where ζ_p and ξ_p are both the adjustment coefficients assigned to the previous learning rate $\eta_{p,k-1}$. Since $\eta_{p,k}$ is adjusted in the iteration process, the error function $E_{p,k}$ approximates the minimum in different directions on the hyper-surface.

3) MODEL CONSTRUCTION

Given that the size of required images are not the same during the training and testing stages and the number of neurons on the input layer is fixed, it is necessary to normalize the image size before grey processing. If the size is small, the image edges are filled with some zeros. In view of a 256-gray-scale image, we have the dynamic range [0,255] for input neurons. Since the original grey information of an image is used for recognition, the number of input neurons is as the number of input image pixels. Besides, the number of neurons on the output layer is:

$$Q_C = \text{round}\left(\log_2 (P+1)\right) \tag{33}$$

where round $(\cdot)$ denotes the nearest integer.

As for the classic BPNN, the robustness is poor when the number of neurons on each hidden layer is insufficient. It requires lots of computational complexity when the number of hidden neurons is extensive. Then, we define the number of neurons by using the geometric array:

$$Q_c = \text{round}\left(\sqrt{Q_{c-1} Q_{c+1}}\right) c = 2, \cdots, C-1 \tag{34}$$

Subsequently, the initial weights of neurons are taken to random values that are all less than 1, but the sum of weights is equal to 1. We put the concerned sub-image based on the coordinate (x, y) in the current frame into the network, and then get the output error E_k by using (20). The relative distance between the current output and the matching result in the previous frame is given by:

$$\Delta E_k = \frac{1}{2} \sum_{p=1}^{P} \sum_{q_C=1}^{Q_C} \left(o_{p,k-1}^{(C,q_C)} - \ell_{p,k}^{(C,q_C)} \right)^2 \tag{35}$$

where $o_{p,k-1}^{(C,q_C)}$ is the output with the same size of sub-image based on the center (x_0, y_0) in the previous frame. Let υ_1 and υ_2 be the coefficients corresponding to E_k and ΔE_k, the total errors can be obtained by the information entropy:

$$\log_2 \upsilon_1 E_k + \log_2 \upsilon_2 \Delta E_k \tag{36}$$

where the point with the minimum value in sub-image is regarded as the re-updated position $\left(x_{k|k}, y_{k|k}\right)$ for target center.

Remark 3: In the stage of BPNN construction, the feature extraction of targets in the sub-image is not operated. The reason is that the sub-image not only contains the given target, but also has the information entropy. The sub-image including the tracked target is input the network when the size of sub-image is small compared with the size of whole image. The efficiency of target tracking is improved.

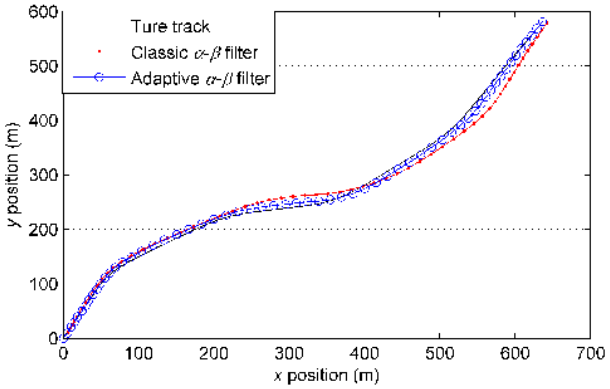


FIGURE 3. Target position estimates.

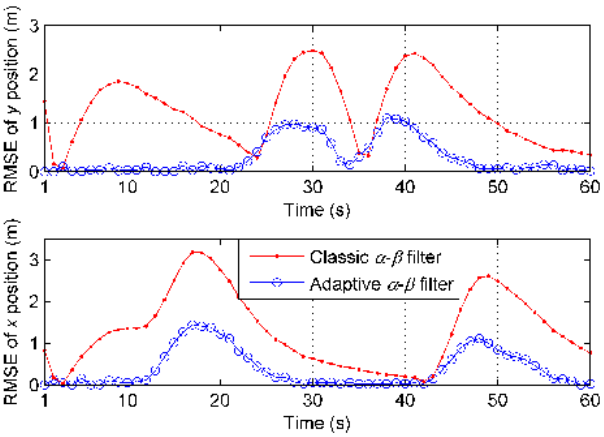


FIGURE 4. RMS in the x and y positions.

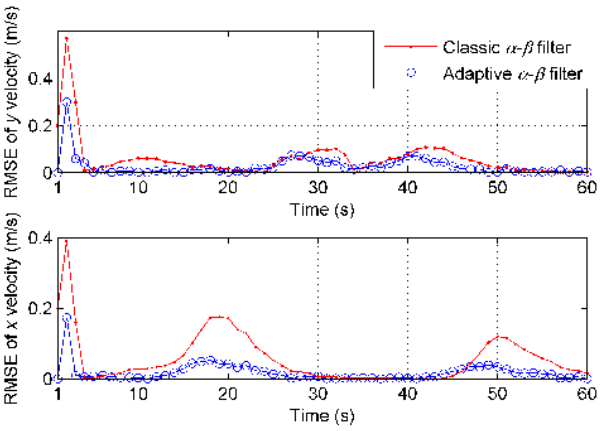


FIGURE 5. RMS in the x and y velocities.

4) PROCESS OF PROPOSED METHOD

① Optimize the α-β filtering parameters via (12)∼(14) and get the target location region Θ via (15);

② Compute the updated position of target center via (4);

③ Set the optimal square of $2a_1 \times 2a_1$ to locate target and get corresponding sub-image as $\Upsilon \times \Upsilon$ pixel before grey operation;

④ Optimize the BPNN parameters via (30), (32)∼(33) and input $\Upsilon \times \Upsilon$ pixel;

⑤ Compute the output $\ell_{p,i}^{(C,q_C)}$ and recognize target with optimal square. If there is no matched result in current image window, the whole image is recognized by blocking. If there is no matching result in the whole image, i.e., the target disappears, then exits the process.

⑥ Compute the error via (35) and get the re-updated position $\left(x_{k|k}, y_{k|k}\right)$ for target center;

⑦ Send the re-updated position into the α-βfilter for prediction in the next cycle.

IV. SIMULATION RESULTS AND DISCUSSIONS

A. NUMERIC STUDY

First, the numerical study is done to verify the adaptive α -β filter for maneuvering target tracking. The experimental environment was: Intel^{TM} Core^{TM} i5, Windows^{TM} 7, 4 GB DDR and MATLAB^{TM} R2018a.

During the tracking time of 60 s, the target has VA motion state. Its velocity is (5,10) m/s and initial position is (0,0) m. The accelerations are (2,0) m/s^2, (0,−1.5) m/s^2, (0,2) m/s^2 and (−1.5,0) m/s^2 during the $11^{th} \sim 15^{th}$s, $21^{st} \sim 25^{th}$s, $31^{st} \sim 35^{th}$s and $41^{st} \sim 45^{th}$s. Other parameters are set as: $T = 1$ s, $\gamma_\kappa = 2.23$, $\rho_0 = 100$ m, $\delta_\rho = 1$ m, $\delta_\theta = 0.1°$ and $\varepsilon = 0.01$. The root of mean squared error (RMSE) is used to compare the classic α-β filter.

Figure 3 shows the true and estimated tracks of the maneuvering target. Although the target has the VA motion states, the adaptive α-β filter effectively estimates it in the x and y coordinates. The estimated position still keeps close to the true track. For comparison, the classic filter gives some deviations because of the target maneuverability.

Figure 4 indicates the RMSE in the x and y positions. Note that the classic filter exaggerates the biased position during the period of motion state transition. Meanwhile, the adaptive filter achieves the prospective tracking whether the target is maneuvering or not. For example, it has 39% RMSE of 2-positions of classic filter at the peak moment.

Figure 5 depicts the RMSE of x and y velocities. Given that the target maneuverability determines the velocity components, the adaptive filter adjusts velocity estimates when the motion state changes abruptly. As a result, the velocity estimates are more accurate.

Subsequently, the tracking performance of the robust BPNN is evaluated. We consider a 4-layer BPNN for simplification. Its parameters are: $\eta_0 = 0.5$, $\varsigma = 1$ and $P = 1$. The size of all images is uniformed as 320×240 pixel, and 128 images are sampled in the training stage.

In Figure 6, the mean squared error (MSE) of the proposed BPNN is indicated. As seen, the values of MSE keep decreasing from $10^{-0.5}$ to 10^{-5}. At the 1834^{th} epoch, the MSE reaches the best value. By comparison, the classic BPNN needs 5778 epochs to achieve the preset value.

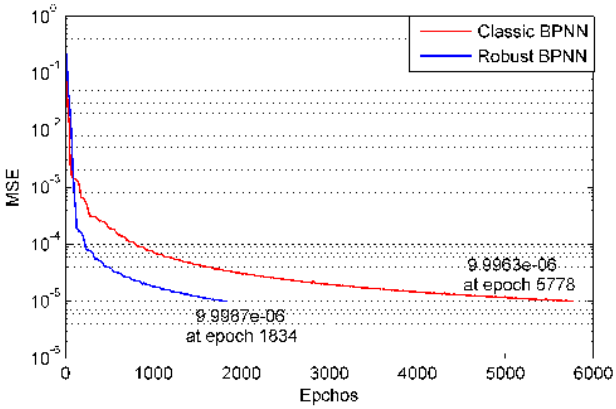


FIGURE 6. MSE and best values.

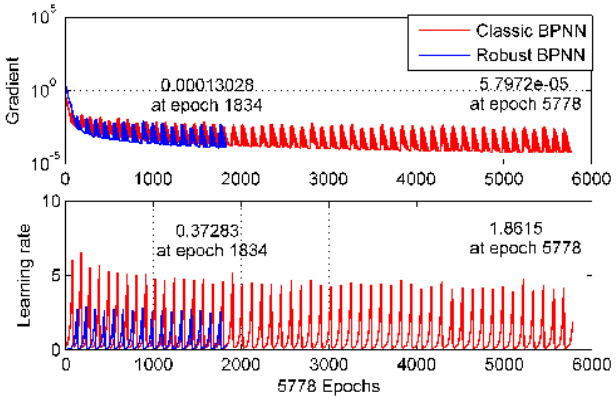


FIGURE 7. Gradient and learning rate.

Figure 7 shows the gradient and learning rate of the classic BPNN and robust BPNN corresponding to the best epoch. Note that the learning rate of the proposed BPNN is lower than that of the classic BPNN. According to the Figure 6, we conclude the proposed BPNN has prospective training performance because of the modified network structure, i.e., the optimal weight and number of the neuron on each layer and the improved learning rate.

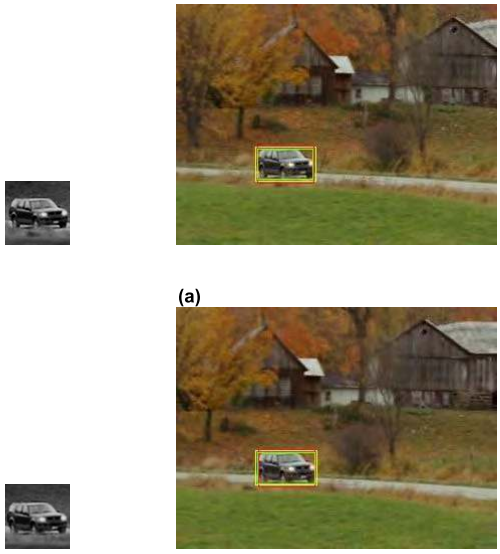
B. EXPERIMENTAL RESULTS AND DISCUSSIONS

We combine the adaptive α-β filter with the robust BPNN to track the vehicle-target under the different scenarios in traffic field for instance, which mainly includes the single target tracking, pair targets tracking, multiple targets tracking and occluded target tracking. Both the TB-100 sequences datasets and the actual traffic images are tested.

1) SINGLE TARGET TRACKING

As for the single target tracking, the numbers of neurons on each layer are $Q_1 = 4096$, $Q_2 = 256$, $Q_3 = 16$ and $Q_4 = 1$. We randomly extract 2 frames (Frames 0091 and 0094) form TB-100 Sequences Datasets.

Figure 8 shows the tracking results. Note that the vehicle can be recognized in the image, and the geometric center of vehicle is marked with a red point by using the proposed method. The detailed results are in Table 1, where the area of vehicle is 64×64 pixel in the adaptive α -β filter for making the robust BPNN convenient. The intermediate results indicate the interactive process in the proposed method. As seen, 2 256-grey-scale images are in the left column. The adaptive α-β filter gives the target center that is updated by using the BPNN for the next cycle. Of course, the classic BPNN can track the vehicle-target. Table 2 cites the average tracking performance of 3 methods. Note that the proposed method has higher tracking rate and accurate with small number of neurons. However, the classic BPNN needs plenty of neurons to get the similar performance. There is no neuron in the KCF that has satisfactory results owing to the kernelized correlation algorithm.



NOTE. Red rectangular: Classic BPNN; Green rectangular: KCF; Yellow rectangular: Proposed method.

FIGURE 8. Single target tracking. (a) Frame 0091. (b) Frame 0094.

TABLE 1. Center and area of single target.

Frame	Method	Center coordinate (pixel)	Area (pixels)
0091	Adaptive α-β filter	(108,161)	64×64
	Robust BPNN	(109,160)	59×34
0094	Adaptive α-β filter	(113,160)	64×64
	Robust BPNN	(112,160)	60×34

2) PAIR TARGETS TRACKING

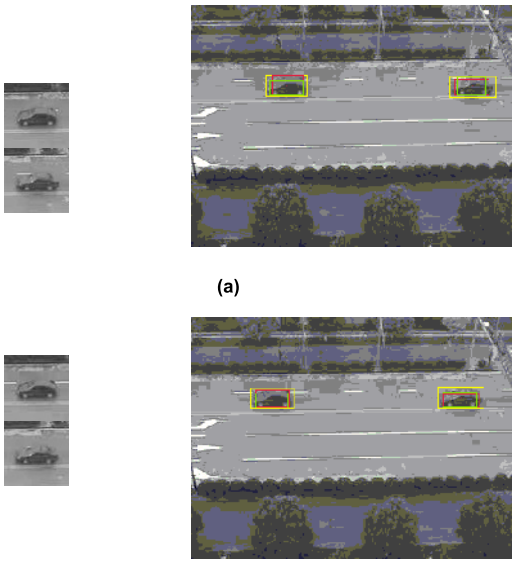
We randomly extract 2 frames (Frames 1186 and 1189) from actual Traffic Datasets.

TABLE 2. Average performance for singe target tracking.

Method	Neuron number	Tracking rate (%)	Accuracy (%)
Classic BPNN	78651	99.02	97.53
KCF	-	99.17	99.08
Proposed method	4369	99.85	99.67

TABLE 3. Center and area of multiple targets.

Frame	Method	Center coordinate (pixel)	Area (pixels)
1186	Adaptive α-β filter	L: (94,80)	64×64
		R: (281,80)	64×64
	Robust BPNN	L: (95,79)	42×22
		R: (280,80)	46×22
1189	Adaptive α-β filter	L: (82,79)	64×64
		R: (270,80)	64×64
	Robust BPNN	L: (81,79)	42×22
		R: (269,80)	46×22



(a)

(b)

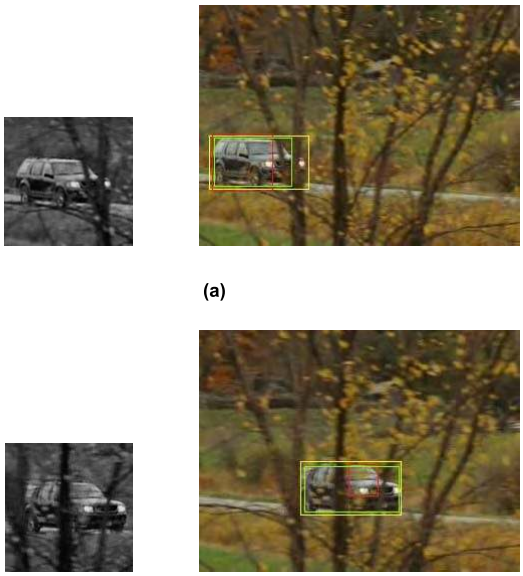
NOTE. Red rectangular: Classic BPNN; Green rectangular: KCF; Yellow rectangular: Proposed method.

FIGURE 9. Multiple targets tracking. (a) Frame 1186. (b) Frame 1189.

The tracking results are demonstrated in Figure 9. We can get the outlines and centers of 2 targets. Combined with Table 3, some information on the vehicles is obtained. We also set the required area of 2-vehicle as 64×64 pixel. Considering the vehicle outline is obscure in 2 images, we achieve the tracking region by using the filter in the left volume. Besides, the adaptive α- β filter finds 2-center coordinates. Within this region, the vehicle is basically located and is input into the robust BPNN. As a result, it reduces the accumulative errors. After the network testing, the updated center and size of targets are sent to the adaptive α -β filter. From Table 4, we can conclude that the proposed method has higher accurate owing to its inherent structure. However, the classic BPNN cannot get satisfactory tracking rate. Recalling Figure 9, we also find that the obscure outlines of 2 targets bring about low accuracy for KCF.

TABLE 4. Average performance for multiple targets tracking.

Method	Neuron number	Tracking rate (%)	Accuracy (%)
Classic BPNN	78651	88.43	95.34
KCF	-	91.76	81.07
Proposed method	4447	98.37	99.50



(a)

(b)

NOTE. Red rectangular: Classic BPNN; Green rectangular: KCF; Yellow rectangular: Proposed method.

FIGURE 10. Occluded target tracking. (a) Frame 0157. (b) Frame 0164.

3) OCCLUDED TARGET TRACKING

In general, there are some special cases should be considered, such as the occluded target tracking. As for the classic BPNN, it often has accumulative estimation errors owing to the uncertain environment. The weights of neurons cannot immediately adjust based on the actual situations, and then lead to the disappearing target or the unmatched target.

Figure 10 gives the occluded target tracking results. We randomly extract 2 frames (Frames 0157 and 0164) from TB-100 Sequences Datasets. Note that the vehicle is occluded by the tree. As we know, the proposed method has

TABLE 5. Center and area of occluded target.

Frame	Method	Center coordinate (pixel)	Area (pixels)
0157	Adaptive α-β filter	(59,158)	128×128
	Robust BPNN	(60,157)	100×54
0164	Adaptive α-β filter	(152,157)	128×128
	Robust BPNN	(150,157)	100×54

TABLE 6. Average performance for occluded target tracking.

Method	Neuron number	Tracking rate (%)	Accuracy (%)
Classic BPNN	78651	52.13	49.08
KCF	-	88.49	72.53
Proposed method	17055	94.75	93.81

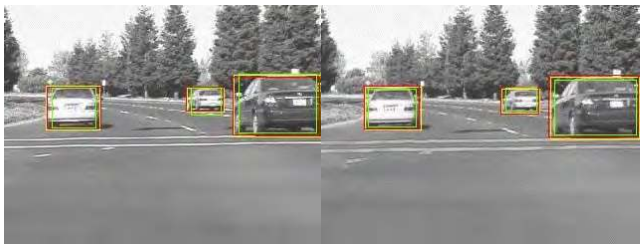
2 individual computing stages. In the adaptive α-β filtering stage, the center coordinate of target has been predicted. Combined with the size from the robust BPNN, we easily get the target location region in Table 5. Inevitably, the occlusion brings about some affection on the true target. According to the target location region, the network still makes full use of pixel information to adjust all weights to match image. Due to the previous information entropy, the accumulative errors are reduced for tracking the vehicle-target. Other methods can only give the partial outline, where the occlusion is also considered as noise so that the unmatched target is obtained. Subsequently, the target position is estimated in the following images. In Table 6, we find that the classic BPNN cannot make effectively tracking in this situation, and then distinguishes the true target with the random noise when the whole image is the input. Therefore, both the tracking rate and the accurate are lower. Although it tracks some occluded targets, the KCF is lack of target identification so that its accuracy is lower.

4) MULTIPLE TARGETS TRACKING

We evaluate the average tracking performance of proposed method by using TB-100 Sequences Datasets and actual Traffic Datasets.

For example, the 3-target and 5-target tracking results are indicated in Figures 11 and 12. We can distinguish the tracking performance of 3 methods. Obviously, the proposed method has prospective performance. The KCF has certain limitation on the fast mobile targets tracking.

Figure 13 shows the average accuracy (above 90%) under 100 trails. Note that the average accuracy decreases with the increasing target cardinality in both methods. The classic BPNN exhibits larger standard deviation (STD). The average accuracy of KCF declines when the target cardinality is more than 3. For comparison, the proposed method has its advance.



(a) (b)

NOTE. Red rectangular: Classic BPNN; Green rectangular: KCF; Yellow rectangular: Proposed method.

FIGURE 11. 3-targert tracking results. (a) Frame 0042. (b) Frame 0045.

(a) (b)

NOTE. Red rectangular: Classic BPNN; Green rectangular: KCF; Yellow rectangular: Proposed method.

FIGURE 12. 5-target tracking results. (a) Frame 1020. (b) Frame 1023.

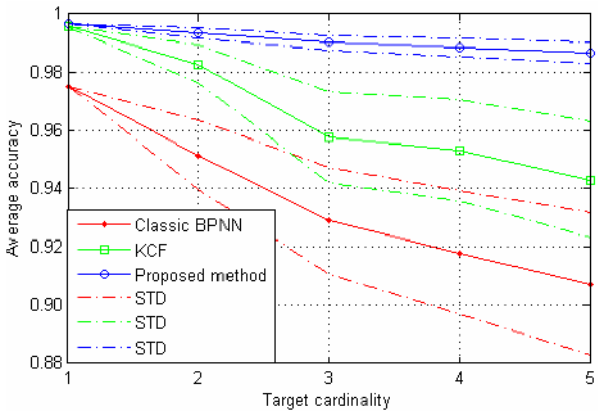


FIGURE 13. Average accuracy.

Employing the adaptive $\alpha-\beta$ filter and the robust BPNN, it overcomes the unstable decision. In view of the tracking reliability, its average accuracy is satisfactory.

Figure 14 demonstrates the number of the required neurons in the proposed method. Although the target cardinalities are different over time, the neurons are around the average value of 4486. It is easy to know there are 76800 neurons on the input layer reaches in the classic BPNN. With respect to the tracking efficiency, enormous neurons are saved when employing the proposed method.

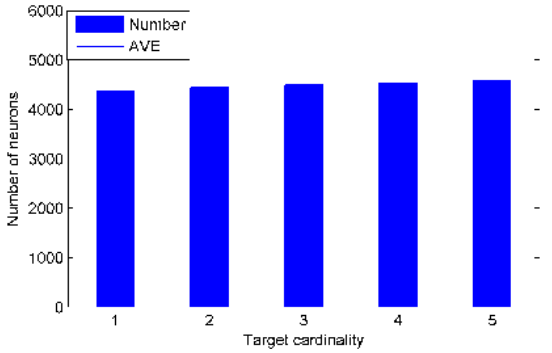


FIGURE 14. Number of required neurons.

V. CONCLUSIONS

This paper has presented an innovative target tracking method combined adaptive α-β filter with robust BPNN. It mainly addresses both efficiency and reliability in various scenes on the interactive tracking mechanism. We employ the adaptive α-β filter to get the required location region based on the optimal filtering parameters in prediction stage. After obtaining the center coordinate and the target area, the 256-grey-scale image information is sent to the BPNN that has optimal number and weight of neurons and the improved learning rate. The network makes an effective recognition and sents back the updated positions of targets to the adaptive α-β filter for the next cycle. Therefore, the robustness and accuracy of the proposed method are inherently boosted when the motion state of targets is maneuvering, even if its outline is obscure and occluded. The numerical study indicates that the proposed method has remarkable improvement. Especially, the actual experiment has verified the advances in various backgrounds. As the future developments of this research, we continue to improve tracking accuracy in the uncertain environment.

REFERENCES

[1] G. Liu, X. Tang, J. Huang, J. Liu, and D. Sun, "Hierarchical model-based human motion tracking via unscented Kalman filter," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[2] C. C. Lin and W. Wolf, "MCMC-based feature-guided particle filtering for tracking moving objects from a moving platform," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops*, Kyoyo, Japan, Sep. 2009, pp. 828–833.

[3] X. Sun, N. M. Cheung, H. Yao, and Y. Guo, "Non-rigid object tracking via deformable patches using shape-preserved KCF and level sets," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 5496–5504.

[4] X. Ning, W. J. Li, W. J. Tian, C. Xu, and J. Xu, "Adaptive template update of discriminant KCF for visual tracking," *CAAI Trans. Intell. Syst.*, vol. 14, no. 1, pp. 121–126, Jan. 2019.

[5] D. P. Wang and Y. Xie, "Target occlusion detection algorithm based on KCF," *Inf. Technol. Netw. Secur.*, vol. 37, no. 12, pp. 39–42, Dec. 2018.

[6] X. B. Lin, *The Particle Filtering Target Detection And Tracking Algorithm Based On Neural Network and Multi-Feature Fusion*. Hangzhou, China: Zhejiang Univ., 2017.

[7] C. T. Fang, *Study of Recognition and Tracking Based on BP Neural Network*. Xi'an, China: Xidian Univ., 2006.

[8] X. Hua, X. Q. Wang, Z. Y. Ma, D. Wang, and F. M. Shao, "Video multi-target detection technology based on recursive neural network," *Appl. Res. Comput.*. doi: 10.19734/j.issn.1001-3695.2018.05.0567.

[9] T. Gao, Y.-J. Liu, L. Liu, and D. Li, "Adaptive neural network-based control for a class of nonlinear pure-feedback systems with time-varying full state constraints," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 5, pp. 923–933, Sep. 2018.

[10] Y.-J. Liu, Q. Zeng, L. Liu, and S. Tong, "An adaptive neural network controller for active suspension systems with hydraulic actuator," *IEEE Trans. Syst., Man, Cybern. Syst.*, to be published. doi: 10.1109/TSMC.2018.2875187.

[11] Y.-J. Liu, Q. Zeng, S. Tong, C. L. P. Chen, and L. Liu, "Adaptive neural network control for active suspension systems with time-varying vertical displacement and speed constraints," *IEEE Trans. Ind. Electron.*, to be published. doi: 10.1109/TIE.2019.2893847.

[12] W. Zhu and C. H. Xia, "Adaptive α-β filter algorithm," *Comput. Appl.*, vol. 27, no. 8, pp. 2053–2055, Aug. 2007.

[13] T.-E. Lee, J.-P. Su, K.-H. Hsia, K.-W. Yu, and C.-C. Wang, "Design of an α-β filter by combining fuzzy logic with evolutionary methods," in *Proc. Int. Symp. Comput., Commun., Control Automat.*, Tainan, Taiwan, May 2010, pp. 270–273.

[14] Y. Ran, H. Li, and H. Li, "Study on adaptive α-β filtering used in ARPA system of the marine radar," *Fire Control Radar Technol.*, vol. 41, no. 4, pp. 39–42, Oct. 2012.

[15] B. Wang, W. Yi, R. Hoseinnezhad, S. Li, L. Kong, and X. Yang, "Distributed fusion with multi-Bernoulli filter based on generalized covariance intersection," *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 242–255, Jan. 2017.

[16] Y.-J. Liu, S. Lu, S. Tong, X. Chen, C. L. P. Chen, and D.-J. Li, "Adaptive control-based Barrier Lyapunov Functions for a class of stochastic nonlinear systems with full state constraints," *Automatica*, vol. 87, pp. 83–93, Jan. 2018.

[17] M. Vinaykumar and R. K. Jatoth, "Performance evaluation of α-β and Kalman filter for object tracking," in *Proc. IEEE Int. Conf. Adv. Commun. Control Comput. Technol.*, Ramanathapuram, India, May 2014, pp. 1369–1372.

[18] P. R. Kalata, "The tracking index: A generalized parameter for α-β and α-β-γ target trackers," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-20, no. 2, pp. 174–182, Mar. 1984.

[19] H. You, X. Jianjuan, and G. Xin, *Radar Data Processing With Applications*. Beijing, China: House Electron. Ind., 2009.

[20] J. T. Wu and J. H. Wang, *Technology and Applications of Neural Networks*. Harbin, China: Harbin Eng. Univ., 1996.

[21] Z. H. Zhou and C. G. Cao, *Applications of Neural Networks*. Beijing, China: Tsinghua Univ., 2004.

[22] J. Wang, J. Yang, and W. Wu, "Convergence of cyclic and almost-cyclic learning with momentum for feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1297–1306, Aug. 2011.

[23] L. C. Jiao, J. Zhao, S. Y. Yang, and F. Liu, *Deep Learning, Optimization and Recognition*. Beijing, China: Tsinghua Univ., 2018.

[24] H. Wang and H. Q. Cheng, "Modified blind equalization algorithm based on back-propagation neural networks with adaptive momentum factor," *Comput. Eng. Des.*, vol. 31, no. 6, pp. 1297–1300, Jun. 2010.

BO LI received the M.S. degree in communication and information system from the Liaoning University of Technology, China, in 2005, and the Ph.D. degree in communication and information system from Dalian Maritime University, China, in 2015. He is currently an Associate Professor with the Liaoning University of Technology, China. His research interests include information fusion, state estimate, target tracking, and digital signal processing.