ADAPTIVE FILTERING ALGORITHMS IN TARGET TRACKING APPLICATIONS

Jovandić I., Đurović Ž., Kovačević B.
Faculty of Electrical Engineering, Belgrade

I INTRODUCTION
Accurate estimation of target position and velocity is the problem of significant importance in a multiple target tracking (MTT) system. Since target dynamics is subject to various changes, solutions based on only one filter with no adaptation at all rarely give satisfactory results [1,2,3].

A number of adaptive algorithms utilizing Kalman filter approach have been proposed [1,2], three of which are to be discussed here. Namely, discrete noise level adjustment (DNLA), variable state dimension (VSD) and interacting multiple model (IMM) algorithms are presented and compared. The DNLA algorithm uses only one Kalman filter with adjustable process noise variance, while VSD and IMM algorithms require multiple filters. The VSD algorithm uses two filters. Only one of these two is used for estimation purposes at a given time, while switching between two filters is done based on quality of the estimate that filter currently in use produces. The IMM algorithm uses both filters simultaneously, combining estimates produced by each of them.

Problem discussed is estimation of position and velocity of a single target in planar motion, based on a sequence of radar measurements. It is assumed that only position measurements are available. Discrete state vectors for constant velocity and constant acceleration (CA) models [1]. Discrete state equation (1) describes these models in $(\xi,\eta)$ plane:

$$x_{k+1} = F \cdot x_k + G \cdot w_k,$$

where state vectors for constant velocity and constant acceleration model are given as $x_k = \left[ \begin{array}{c} \xi_k \\ \xi_k \\ \eta_k \\ \eta_k \end{array} \right]$ and $x_k = \left[ \begin{array}{c} \xi_k \\ \xi_k \\ \eta_k \\ \eta_k \end{array} \right]$, respectively. Process noise vector is given as $w_k = \left[ \begin{array}{c} w_x^k \\ w_y^k \end{array} \right]$. Process noise components are Gaussian zero-mean white noise sequences and are assumed to be independent. In the case of CV (CA) model, they represent velocity (acceleration) increments during $k$-th sampling period in $\xi$ and $\eta$ coordinates.

Variance of both process noise components are assumed to be equal and are marked as $\sigma^2_x$. Therefore process noise covariance matrix can be written as $Q_k = \left[ \begin{array}{c} 1 & 0 \\ 0 & 1 \end{array} \right] \cdot \sigma^2_x$.

Matrices $F$ and $G$ for constant velocity model are given by equation (2):

$$F = \left[ \begin{array}{c} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{array} \right], \quad G = \left[ \begin{array}{c} T/2 & 0 \\ 0 & 1 \\ 0 & T/2 \\ 0 \end{array} \right].$$

Matrices $F$ and $G$ for constant acceleration model are given by equation (3):

$$F = \left[ \begin{array}{c} 1 & T^2/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{array} \right], \quad G = \left[ \begin{array}{c} T^2/6 & 0 \\ 0 & 1 \\ 0 & T^2/6 \\ 0 \end{array} \right].$$

Since only position measurements are available, discrete measurement equation is:

$$z_k = H \cdot x_k + v_k,$$

where

$$H = \left[ \begin{array}{c} 0 \\ 0 \end{array} \right].$$

for constant velocity model and

$$H = \left[ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array} \right]$$

for constant acceleration model. Therefore dimension of measurement vector $z_k$ is $n_z = 2$. Measurement noise vector consists of two components and is given as: $v_k = \left[ \begin{array}{c} v_x^k \\ v_y^k \end{array} \right]$. Measurement noise components are Gaussian zero-mean white noise sequences and are assumed to be independent. Variances of both measurement noise components are assumed to be equal and constant, and are marked as $\sigma^2_v$.

Therefore measurement noise covariance matrix can be written as $R_k = \left[ \begin{array}{c} 1 & 0 \\ 0 & 1 \end{array} \right] \cdot \sigma^2_v$.

II TARGET MODELS
Target models used here are simple constant velocity (CV) and constant acceleration (CA) models [1]. Discrete state equation (1) describes these models in $(\xi,\eta)$ plane:

$$x_{k+1} = F \cdot x_k + G \cdot w_k,$$

where state vectors for constant velocity and constant acceleration model are given as $x_k = \left[ \begin{array}{c} \xi_k \\ \xi_k \\ \eta_k \\ \eta_k \end{array} \right]$ and $x_k = \left[ \begin{array}{c} \xi_k \\ \xi_k \\ \eta_k \\ \eta_k \end{array} \right]$, respectively. Process noise vector is given as $w_k = \left[ \begin{array}{c} w_x^k \\ w_y^k \end{array} \right]$. Process noise components are Gaussian zero-mean white noise sequences and are assumed to be independent. In the case of CV (CA) model, they represent velocity (acceleration) increments during $k$-th sampling period in $\xi$ and $\eta$ coordinates.

Variance of both process noise components are assumed to be equal and are marked as $\sigma^2_x$. Therefore process noise covariance matrix can be written as $Q_k = \left[ \begin{array}{c} 1 & 0 \\ 0 & 1 \end{array} \right] \cdot \sigma^2_x$.

Matrices $F$ and $G$ for constant velocity model are given by equation (2):

$$F = \left[ \begin{array}{c} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{array} \right], \quad G = \left[ \begin{array}{c} T/2 & 0 \\ 0 & 1 \\ 0 & T/2 \\ 0 \end{array} \right].$$

Matrices $F$ and $G$ for constant acceleration model are given by equation (3):

$$F = \left[ \begin{array}{c} 1 & T^2/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{array} \right], \quad G = \left[ \begin{array}{c} T^2/6 & 0 \\ 0 & 1 \\ 0 & T^2/6 \\ 0 \end{array} \right].$$

Since only position measurements are available, discrete measurement equation is:

$$z_k = H \cdot x_k + v_k,$$

where

$$H = \left[ \begin{array}{c} 0 \\ 0 \end{array} \right].$$

for constant velocity model and

$$H = \left[ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array} \right]$$

for constant acceleration model. Therefore dimension of measurement vector $z_k$ is $n_z = 2$. Measurement noise vector consists of two components and is given as: $v_k = \left[ \begin{array}{c} v_x^k \\ v_y^k \end{array} \right]$. Measurement noise components are Gaussian zero-mean white noise sequences and are assumed to be independent. Variances of both measurement noise components are assumed to be equal and constant, and are marked as $\sigma^2_v$.

Therefore measurement noise covariance matrix can be written as $R_k = \left[ \begin{array}{c} 1 & 0 \\ 0 & 1 \end{array} \right] \cdot \sigma^2_v$.

III ADAPTIVE FILTERING ALGORITHMS
An overview of algorithms discussed is presented.

DNLA

The DNLA algorithm is based on a single, constant velocity model [1]. Kalman filter corresponding to this model is initialized with some low-level process noise variance, marked as $\sigma^2_x$. Residual (innovation) sequence is constantly monitored. If 'large' innovation is detected it is assumed that target is maneuvering, so Kalman filter is reinitialized with new, larger process noise variance (marked as $\sigma^2_x'$) in order to 'cover' the maneuver. Detection of large innovation can be performed in several ways; normalized innovation squared (NIS) approach is applied here. Normalized innovation squared is defined as:
where \( r_k \) is innovation and \( S_k \) is innovation covariance matrix, both in \( t = kT \). NIS is chi-square distributed, with \( n_i \) degrees of freedom (\( n_i \) is dimension of measurement vector; here \( n_i = 2 \)). A threshold \( e_{TR}^a \) can be defined based on the chosen probability that maneuver takes place when \( e_k < e_{TR}^a \). That probability should be low, say 0.01. Process noise switching from low to high value is done when \( e_k > e_{TR}^a \) is detected, and return from high to low process noise value when \( e_k < e_{TR}^a \).

**VSD**

The VSD algorithm utilizes both constant velocity and constant acceleration model [1,4]. Nominal (nonmaneuvering) model is the CV model. Fading memory average (FMA) of nominal model innovations is constantly monitored in order to detect a maneuver. If FMA exceeds certain threshold it is declared that maneuver has started. When choosing a value of FMA threshold one has to have in mind the definition of fading memory average given by equation (7):

\[
e_k^a = a \cdot e_{k-1}^a + e_k, \quad 0 < a < 1
\]

where \( a \) is fading parameter and \( e_k \) is normalized innovation squared as defined in (6). FMA can be regarded as chi-square distributed with \( s \cdot n_i \) degrees of freedom, where \( s = \frac{1}{1-a} \) is effective window length. FMA threshold value \( e_{TR}^a \) then corresponds to chosen probability that maneuver has taken place when \( e_k^a < e_{TR}^a \). That probability should be low, say 0.01.

When maneuver is detected in \( t = kT \) one should initialize Kalman filter based on CA (maneuvering) model, so future measurements (\( z_{k+1}, z_{k+2}, \ldots \)) could be processed with it. That is, state estimate for maneuvering model \( \hat{x}_k^m \) and its covariance matrix \( P_k^m \) are needed. It is assumed that maneuver has started \( s \) periods before detection, so values \( \hat{x}_{k-s}^m, P_{k-s}^m \) are calculated based on nonmaneuvering state estimate \( \hat{x}_{k-s-1} \), its covariance matrix \( P_{k-s-1} \) and measurement \( z_{k-s} \). Relations defining these calculations (in \( \xi \) coordinate only) are given by following equations:

\[
\hat{x}_{k-s}^m = \frac{2}{T^2} \hat{x}_{k-s-1} + \frac{2}{T^2} T \hat{x}_{k-s-1}^T
\]

\[
P_{k-s}^m = \frac{4}{T^2} R_{k} + \frac{4}{T^2} P_{k-s-1} + \frac{6}{T} P_{k-s} + \frac{2}{T} P_{k-s-1}^T
\]

where \( P_{\mu_{11}}, P_{\mu_{12}}, P_{\mu_{13}}, P_{\mu_{22}}, P_{\mu_{33}} \) and \( P_{\mu_{33}}^m \) are elements of \( P_{\mu_{33}} \), corresponding to \( \xi \) coordinate, \( P_{\mu_{11}}, P_{\mu_{12}} \) and \( P_{\mu_{22}} \) are elements of \( P_{\mu_{22}} \), \( R_{\mu_{1}} \) and \( R_{\mu_{2}} \) are elements of \( R_{\mu_{22}} \) are elements of \( R = cov(\dot{v}) \).

Measurements \( z_{k-s-1}, ..., z_{k-s} \) are reprocessed with filter based on CA model, so values \( \hat{x}_k^m, P_k^m \) are obtained. Measurements arriving after \( t = kT \) are processed with filter based on CA model.

Moving sum of normalized estimates of acceleration, given by (8), is monitored in order to detect maneuver termination:

\[
MS_k = \sum_{j=k-p+1}^{k} \delta_r^j,
\]

where \( \delta_r^j = \begin{bmatrix} \hat{x}_k^m \\ \hat{\dot{x}}_k^m \\ P_{\mu_{33}}^m(k) \\ P_{\mu_{33}}^m(k^-1)^{-1} \end{bmatrix} \) and \( p \) is sliding window length.

Moving sum (MS) is chi-square distributed with \( p \cdot n_i \) degrees of freedom, so threshold for MS can be set so probability that maneuver has ended when \( MS_k > MS_{TR} \) is low. If MS falls below this threshold it is assumed that maneuver is terminated and measurements to come should be processed with filter based on CV model. Initialization of CV filter is done by simple truncation of acceleration states in state vector and corresponding elements in state covariance matrix.

**IMM**

One cycle of the IMM algorithm is depicted in figure 1. Short description of algorithm will be given [1,5]. Two Kalman filters (filter M1 set according to CV model and filter M2 set according to CA model) operate simultaneously producing state estimates \( \hat{x}^1 \) and \( \hat{x}^2 \) with corresponding covariance matrices \( P^1 \) and \( P^2 \). In addition, both filters produce their likelihood function values defined as:

\[
\Lambda(k) = \frac{1}{(2\pi)^{\frac{N}{2}} |S_k|} \exp\left( -\frac{1}{2} r_k^T \cdot S_k^{-1} \cdot r_k \right),
\]

where \( r_k \) is innovation and \( S_k \) is innovation covariance matrix, both in \( t = kT \). Estimates \( \hat{x}^1 \) and \( \hat{x}^2 \), together with corresponding covariance matrices \( P^1 \) and \( P^2 \) (all in \( t = (k-1)T \), that is resulting from previous cycle) are mixed to form initial conditions \( \hat{x}^{01} \) and \( \hat{x}^{02} \) and corresponding covariance matrices \( P^{01} \) and \( P^{02} \) for processing to come. Variables marked as \( \mu \) are called mixing probabilities.

Results of processing (state estimates and corresponding covariance matrices) are combined to form final (output) estimate. Combining is done according to mode probabilities \( \mu \). Algorithm is adaptive since mode probabilities \( \mu \) and \( \mu_2 \) are changed based on quality of estimates produced; here, likelihood functions are taken as measures of estimate quality. The IMM algorithm assumes that model jump process is a Markov process with known transition probabilities:
\begin{align*}
p_{ij} &= P[M(k) = M_j \mid M(k-1) = M_i], \quad i, j \in \{1,2\}, \\
&\text{where } M(k) \text{ is mode (model) at time } t = kT. \text{ These mode transition probabilities are assumed to be time invariant and independent of the base state. Since only two models are used, transition probability matrix } P_{\text{TRAN}} \text{ is given by (9):}
\end{align*}

\begin{equation}
P_{\text{TRAN}} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}.
\end{equation}

One cycle of the algorithm consists of the following five steps (number of filters is marked as \( r \)):

1. **Calculation of the mixing probabilities** (\( i, j = 1, \ldots, r \)):

\[
\mu_{ij}(k-1 \mid k-1) = \frac{1}{\mathcal{C}_j} \cdot p_{ij} \cdot \mu_i(k-1),
\]

where normalizing constants are:

\[
\mathcal{C}_j = \sum_{i=1}^r p_{ij} \cdot \mu_i(k-1).
\]

2. **Mixing** (\( j = 1, \ldots, r \)):

\[
\dot{x}^{0j}(k-1 \mid k-1) = \sum_{i=1}^r \dot{x}^i(k-1 \mid k-1) \cdot \mu_{ij}(k-1 \mid k-1),
\]

\[
P^{0j}(k-1 \mid k-1) = \sum_{i=1}^r \mu_{ij}(k-1 \mid k-1) \cdot \left\{ P^i(k-1 \mid k-1) + \right. \\
&\left. + \text{err}^0_j(k-1 \mid k-1) \cdot \text{err}_0^j(k-1 \mid k-1)^T \right\}
\]

where

\[
\text{err}^0_j(k-1 \mid k-1) = \dot{x}^i(k-1 \mid k-1) - \dot{x}^{0j}(k-1 \mid k-1).
\]

3. **Mode matched filtering** (\( j = 1, \ldots, r \)): starting with \( \dot{x}^{0j}(k-1 \mid k-1), P^{0j}(k-1 \mid k-1) \) and \( z(k) \), Kalman filter matched to model \( M_j \) calculates the estimate \( \hat{x}^j(k \mid k) \), its covariance matrix \( P^j(k \mid k) \) and likelihood function \( \Lambda_j(k) \).

4. **Mode probability update** (\( j = 1, \ldots, r \)):

\[
\mu_j(k) = \frac{1}{c} \Lambda_j(k) \cdot \mathcal{C}_j,
\]

where

\[
c = \sum_{j=1}^r \Lambda_j(k) \cdot \mathcal{C}_j.
\]

5. **Estimate and covariance combination**: this step is used for output purposes only; it is not a part of algorithm recursions:

\[
\hat{x}(k \mid k) = \sum_{j=1}^r \hat{x}^j(k \mid k) \cdot \mu_j(k),
\]

\[
P(k \mid k) = \sum_{j=1}^r \mu_j(k) \cdot \left\{ P^j(k \mid k) + \text{err}_j(k \mid k) \cdot \text{err}_j(k \mid k)^T \right\},
\]

where \( \text{err}_j(k \mid k) = \hat{x}^j(k \mid k) - \hat{x}(k \mid k) \).

**IV EXPERIMENTAL RESULTS**

Algorithms described are compared using computer simulation. Target trajectory in \( (\xi, \eta) \) plane is defined. Initial target state is described by following equations:

\[
\xi_0 = -10 \cdot 10^4[m], \quad \eta_0 = -15 \cdot 10^4[m],
\]

\[
\dot{\xi}_0 = 100[m/s], \quad \dot{\eta}_0 = 120[m/s],
\]

\[
\ddot{\xi}_0 = 0[m/s^2], \quad \ddot{\eta}_0 = 0[m/s^2].
\]

During first 50 scans target exhibits constant velocity motion. First maneuver (constant acceleration motion) starts in 51st scan and lasts 10 scans. Constant velocity motion follows and lasts 50 scans. Second maneuver is of the same type as the first one, starts in 111th scan and lasts only 5 scans. During the rest of 55 scans target has constant velocity motion.

Accelerations in \( m/s^2 \) during first and second maneuver are \( [a_{\xi}, a_{\eta}] = [6, 0] \) and \( [a_{\xi}, a_{\eta}] = [0, 12] \), respectively.

Sampling period is \( T = 1[s] \). Target trajectory is shown in figure 2.
Measurement noise standard deviation is $\sigma_n = 100 [m]$ in both coordinates. Same random numbers (representing measurement noise samples) are used for the simulation of all three algorithms. Parameters used in each algorithm are given. Initial state estimate is chosen based on position measurements in all three cases (initial velocity and acceleration estimates are zero). NIS threshold in the DNLA algorithm is $e_{TR} = 5$, while low and high process noise variances are $\sigma_L^2 = 10^{-3}$ and $\sigma_H^2 = 20$, respectively. Parameters used in the VSD algorithm are $a = 0.89$, $p = 2$, $e_{TR}^a = 30$ and $MS_{TR} = 0.1$. Transition probability matrix in the IMM algorithm is

$$P_{\text{TRAN}} = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix},$$

while initial mode probabilities are $\mu_1 = 0.9$ and $\mu_2 = 0.1$; process noise variances are $\sigma_{CV}^2 = \sigma_{CL}^2 = 0.1$. Criterion for comparison of algorithms is the value of cumulative estimation error (CEE) defined as follows:

$$CEE(k) = \frac{1}{k} \sum_{i=1}^{k} \frac{(x_i - \hat{x}_i)^T \cdot W \cdot (x_i - \hat{x}_i)}{x_i^T \cdot W \cdot x_i},$$

where $W$ is diagonal weighting matrix whose values are chosen in order to equalize the influence of position and velocity estimation errors; here $W = \text{diag}(1 \ 1 \ 1)$.

Figure 3 gives CEE plots for each of described algorithms; logarithmic scale is used.

**V CONCLUSION**

By inspecting CEE plots one concludes that the IMM algorithm is superior compared to the other two, both during maneuver and in steady state. The VSD algorithm is slightly better then the DNLA algorithm in a view of overall performance. All algorithms have some maneuver detection delay (a few scans only). The estimation error during maneuver is largest in the DNLA case; this result shows that an algorithm utilizing one filter only hardly covers the maneuver in spite of adaptation. Nevertheless, rather good results were obtained in the case of the second (fast and short) maneuver compared to results of the VSD algorithm in the same case. That reveals a drawback of the VSD algorithm: the mere necessity to initialize the maneuvering filter. The IMM algorithm does not encounter this problem since both filters operate simultaneously. However, one should account for the price of such estimation quality. The price is here expressed in terms of computational complexity of an algorithm. The subject of computational complexity is not discussed here, but one could conclude (based on description of algorithms) that the DNLA algorithm is the simplest one.

One should note that the IMM algorithm is not limited on two models only; equations given describe algorithm in general case where $r$ models are used.

**LITERATURE**


